Universidad de Cuenca

Facultad de Ingeniería

Carrera de Ingeniería en Telecomunicaciones

Diseño e Implementación de VLANs en Entornos Universitarios Basados en Redes Definidas por Software (SDN)

Trabajo de titulación previo a la obtención del título de Ingeniero en Telecomunicaciones

Autores:

Erick Patricio Pérez Peralta

Kevin Mateo Molina Yunga

Director:

Andrés Marcelo Vázquez Rodas

ORCID: 0000-0002-6114-1179

Cuenca, Ecuador

2023-08-22

Resumen

Una Campus Area Network (CAN) es una red de infraestructura computacional que interconecta varias Local Area Network (LAN) dentro de un campus, abarcando múltiples ubicaciones y proporcionando conectividad a dispositivos en toda la CAN. A medida que crece en tamaño y complejidad, se fragmenta en Virtual Local Area Networks (VLANs) para optimizar eficiencia y seguridad. Las redes definidas por software (SDN) brindan ventajas al permitir la asignación dinámica de VLANs. La separación de control y datos en una SDN facilita la flexibilidad en la gestión de VLANs. Este trabajo propone diseñar y poner en práctica un entorno de pruebas y análisis de VLANs basadas en SDN para contextos universitarios. Utiliza herramientas de código abierto para configurar una CAN a pequeña escala. Primero, se emula una red clásica con Raspberry Pi actuando como conmutadores de red y Open vSwitch (OvS), asignando VLANs mediante comandos de terminal. Luego, se implementa el controlador OpenDaylight (ODL) para establecer una SDN y asignar VLANs de manera estática. La tercera etapa implica la creación de un servidor de acceso a la red (NAC) en Python, utilizando la capa de aplicación de la SDN para obtener información topológica y VLANs dinámicamente. Los resultados obtenidos se presentan junto con conclusiones relevantes. Además, se proporciona una guía general para la asignación dinámica de VLANs en una CAN con SDN. Este estudio explora la implementación y beneficios de VLANs en una CAN, destacando cómo las SDN ofrecen adaptabilidad y agilidad en la administración de redes, especialmente en una CAN.

Palabras clave: red definida por software, vlan, conmutador ovs, nac

Abstract

A Campus Area Network (CAN) is a computational infrastructure network that interconnects multiple Local Area Networks (LANs) within a campus, spanning multiple locations and providing connectivity to devices across the entire CAN. As it grows in size and complexity, it is divided into Virtual Local Area Networks (VLANs) to optimize efficiency and security. Software-Defined Networks (SDNs) offer advantages by enabling dynamic assignment of VLANs. The separation of control and data in an SDN facilitates flexibility in VLAN management. This work proposes designing and implementing a testing and analysis environment for SDN-based VLANs in university settings. It utilizes open-source tools to set up a small-scale CAN. Initially, a classic network is emulated using Raspberry Pi as network switches and Open vSwitch (OvS), with VLANs assigned through terminal commands. Subsequently, the OpenDaylight (ODL) controller is deployed to establish an SDN and statically allocate VLANs. The third phase involves creating a Network Access Control (NAC) server in Python, utilizing the SDN's application layer to dynamically obtain topological information and VLANs. The obtained results are presented alongside relevant conclusions. Additionally, a comprehensive guide is provided for dynamically assigning VLANs in a CAN with SDN. This study explores the implementation and benefits of VLANs in a CAN, highlighting how SDNs offer adaptability and agility in network administration, particularly within a CAN context.

Keywords: software-defined networking, vlan, switch ovs, nac

Índice de contenido

Índice de figuras

Índice de tablas

Agradecimientos

Queremos expresar nuestro profundo agradecimiento al Ing. Andrés Vázquez por su inestimable orientación y apoyo en la realización de este proyecto de titulación. Asimismo, deseamos extender este agradecimiento a todos nuestros estimados profesores universitarios, quienesgenerosamente han compartido sus conocimientos y experiencia, contribuyendo así a nuestro desarrollo académico integral.

Erick y Mateo

Dedicatoria

A mis padres, Ana y Patricio, quienes han sido mi guía constante y apoyo incondicional en cada paso que he dado. Les estoy profundamente agradecido por ser los pilares que me han permitido alcanzar cada una de mis metas. Les debo mi formación como profesional y mi desarrollo personal, ya que su amor y dedicación han sido fundamentales en mi crecimiento.

A mi hermana, Johanna, quien ha sido un apoyo incondicional y una guía invaluable en mi camino hacia la superación personal y profesional.

Erick

A mis padres, Gerardo y Johanna, quienes han sido un pilar fundamental en este camino deformación académica y profesional. Gracias por su amor incondicional, apoyo constante y porcreer en mis capacidades. Su confianza y sacrificio ha sido la fuerza impulsora detrás de cadalogro alcanzado. Gracias por ser mis ejemplos de perseverancia, humildad y generosidad.

A mi hermano, Tomás, quien ha sido testigo de mis largas horas de estudio, mis momentos de frustración y mis incontables sacrificios. Tu presencia, aunque a veces sin darte cuenta, ha sido una luz que me ha impulsado a seguir adelante, incluso cuando las cosas parecían difíciles.

A mis abuelos, Ricardo y Melchora, esta meta que hoy alcanzo no podría estar completo sinreconocer el inmenso papel que han desempeñado en mi vida. A lo largo de los años, han sidomis guías, mis modelos a seguir y mis fuentes de sabiduría. Este logro es su logro también, yquiero que se sientan orgullosos de la persona en la que me he convertido gracias a su amory valores que me han inculcado.

A mis tíos y primos, este logro no habría sido posible sin el respaldo y confianza de cada uno de ustedes. A todos ellos, le dedico este trabajo, como un pequeño reconocimiento a suimportancia en mi formación académica y como un gesto de gratitud eterna.

Mateo

Abreviaciones y Acrónimos

API Application Programming Interfaces. 34

ARP Address Resolution Protocol. 35

CAN Campus Area Network. 2, 15–19, 49, 70, 71

DHCP Dynamic Host Configuration Protocol. 53

ICMP Internet Control Message Protocol. 82

IP Internet Protocol. 92

LAN Local Area Network. 2, 15

MAC Media Access Control. 37

MAN Metropolitan Area Network. 15

NAC Network Access Control. 40, 70

NOS Network Operating System. 33

ODL OpenDayLight. 8, 56–60, 62, 63, 65–67, 83, 95, 96

OFM OpenFlow Manager. 8, 59, 60, 66, 67, 98, 99

ONF Open Networking Foundation. 28

OvS Open vSwitch. 2, 8, 23, 53–55, 58, 59, 69, 83, 85, 93–95

PAN Personal Area Network. 15

PIF Physical Interface. 41

QoS Quality of Service. 42

RPC Remote Procedure Call. 33

RPi Raspberry Pi. 9, 53, 55, 89, 90, 94

RTP Real Time Protocol. 72, 84

SDN Software-Defined Networking. 8, 21, 23, 33, 56, 59, 62, 63, 66, 70–72, 82, 83, 87

VIF Virtual Interface. 41

VLAN Virtual Local Area Network. 8, 9, 37–39, 42, 50, 53, 54, 62, 63, 65–67, 72, 82, 115

VLANs Virtual Local Area Networks. 2, 8, 22, 39, 55, 62, 66, 70, 71, 83, 87

WAN Wide Area Network. 15

1. Introducción

Este capítulo tiene como objetivo presentar una introducción al funcionamiento de la arquitectura Campus Area Network (CAN) en el ámbito del control y la administración de una red distribuida en varios departamentos, incluyendo administrativos, docentes y alumnado. Se destacan tanto las ventajas como las limitaciones de esta arquitectura, las cuales impulsan el desarrollode nuevas arquitecturas y paradigmas para mejorar y mantener redes de escala media. En la primera parte (Sección 1.1), se aborda la problemática identificada en una CAN, la cual hallevado a proponer un nuevo paradigma de arquitectura de red. Posteriormente, en la Sección1.2, se presenta una contribución a la arquitectura de red que busca solucionar dicha problemática y se establecen los objetivos del presente trabajo. A continuación, en la Sección 1.3, se expone el alcance de este proyecto y se plantean los objetivos específicos (Sección 1.4).

1.1. Identificación del problema

Las redes tradicionales han evolucionado para adaptarse a los distintos requerimientos de tráfico y aplicaciones de red. Se dividen según su tamaño y alcance, abarcando inicialmente cuatro tipos: Personal Area Network (PAN), Local Area Network (LAN), Metropolitan Area Network (MAN) y Wide Area Network (WAN). Sin embargo, con la constante evolución tecnológica, la conexión a Internet se ha vuelto esencial, lo que ha llevado a la necesidad de subdividir las clases originales de redes para permitir la sectorización y distribución de las redes desplegadas. En el ámbito de la educación, se han desarrollado las Redes de Campus o CAN, queconsisten en una colección de redes LAN desplegadas en áreas geográficas concentradas, como los campus universitarios. Estas redes están compuestas por dispositivos de red como conmutadores y enrutadores, siendo administradas por los departamentos de tecnologías de la información de las instituciones educativas [2]. Las CAN brindan conectividad a los diferentesactores de la institución, incluyendo áreas administrativas, docentes, estudiantiles, etc. Una ventaja significativa de las CAN es su capacidad para distribuir eficientemente a los usuariosconectados dentro de la red, lo que reduce la latencia en comparación con las redes MAN o WAN, ya que los datos se distribuyen localmente.

Un componente esencial de las Redes de Campus (CAN) son las Redes de Área Local Virtual(VLAN). Las VLAN son una herramienta muy útil para dividir y separar una red en

diferentessegmentos, reduciendo los dominios de difusión de manera independiente a la ubicación física de los dispositivos y terminales. Además, brindan la oportunidad de controlar el acceso de los clientes dentro de la red. Normalmente, en empresas o universidades, el Centro de Operaciones de Red (NOC) utiliza VLAN para separar los diferentes departamentos de la organización[6]. Debido a los beneficios y ventajas que ofrece el uso de VLAN, los operadores y administradores de red han adoptado ampliamente esta herramienta en la construcción de sus redes.Sin embargo, en redes de gran escala, la configuración de VLAN puede volverse un procesocomplejo, tedioso, lento y propenso a errores. En las redes tradicionales, el plano de control se basa en un enfoque distribuido, donde los protocolos de red funcionan de forma independiente en cada dispositivo [7]. Estos dispositivos de red se conectan entre sí, pero no hay una máquina centralizada que administre toda la red. Asimismo, en las redes tradicionales se utilizan conmutadores, enrutadores y otros dispositivos de *hardware* físico para establecer conexiones y operar la red [8].

A raíz de esto, ha surgido un nuevo paradigma conocido como redes definidas por software (SDN). En las SDN, se separa el plano de control del plano de datos, lo que permite que las redes sean fácilmente monitoreadas y administradas por un controlador central. Este controlador toma decisiones precisas para lograr una configuración óptima de la red. Esta separaciónde funciones y la existencia de un servidor dedicado son posibles gracias a las SDN [9, 10]. En contraste con las arquitecturas de red tradicionales, las SDN ofrecen un control centralizado que brinda una visión completa de toda la red. En este caso, todas las configuraciones y programaciones de la red se realizan a través del controlador, lo que acelera los procesos deaprovisionamiento y la prestación de servicios para los dispositivos conectados a la red. Debido a su capacidad de administración centralizada y programabilidad de la red, las SDN se presentan como una solución prometedora para abordar los desafíos mencionados anteriormente en la gestión de las VLAN.

1.2. Justificación

Las SDN son un paradigma emergente que se está implementando gradualmente en redes de mediana y gran escala, como las CAN. Los departamentos de tecnologías de la información ycodificación tienen la necesidad de investigar e implementar VLANs para la administración ysegmentación de la red, permitiendo la identificación de usuarios y

dispositivos de diferentesdepartamentos. Esto implica la identificación de escenarios, configuraciones de red y el desarrollo de aplicaciones SDN específicas para la CAN. Por lo tanto, es necesario contar con entornos de pruebas que permitan el desarrollo y la aplicación de pruebas de concepto, con el objetivo de proponer soluciones para problemas específicos y evaluar de manera objetiva y técnica el control y la gestión de la red. En este sentido, se propone el diseño e implementación de VLANs con asignación dinámica en entornos universitarios basados en SDN, utilizando unentorno de pruebas para analizar y experimentar el comportamiento de la CAN, así como sucontrol y gestión. El objetivo de este entorno de pruebas es facilitar el estudio de la asignación de VLANs de manera estática y dinámica, para analizar las ventajas que ofrece en la administración de redes de mediana escala, con miras a una posible implementación en una CAN yadesplegada.

1.3. Alcance

En el contexto de los antecedentes presentados en la Sección 1.2, el objetivo de este trabajo de titulación es proporcionar una solución a la falta de control centralizado y automatización en los procesos de asignación dinámica de VLANs en entornos universitarios basados en SDN. La falta de dinamismo en la asignación de VLANs a diferentes tipos de usuarios y dispositivos dela red universitaria, como administrativos, docentes, estudiantes, cámaras, entre otros, genera la necesidad de contar con personal capacitado en el campo de las redes de telecomunicaciones para llevar a cabo el registro, autenticación y asignación correspondiente de VLANs según el grupo al que pertenezca el usuario. Por lo tanto, es crucial implementar un nuevo modelo de asignación de VLANs que agregue dinamismo a la red y, al mismo tiempo, la automatice, centralizando este proceso mediante un controlador único a través de la implementación de una SDN. Esto permitirá evitar configuraciones manuales y optimizar la red para que sea programable, donde el controlador desempeñará el papel de administración y control.

En este trabajo se diseñará e implementará una topología de una red clásica con asignación de VLANs utilizando Open vSwitch (OvS) para crear los nodos switch en tarjetas RPi. Posteriormente, se procederá a implementar una topología SDN utilizando el protocolo OpenFlow en equipos RPi y el controlador OpenDaylight, con el objetivo de emular una CAN dividida en diferentes departamentos. Además, se incorporará un servidor de control de acceso a la red(NAC) en la topología para autenticar a los usuarios y asignar VLANs según su tipo y departamento al que pertenezcan. Este enfoque nos

permitirá obtener una red SDN capaz de asignarVLANs de forma dinámica a diversos hosts, lo cual nos permitirá realizar un análisis comparativo entre las VLANs dinámicas y estáticas en una red SDN. Este análisis se basará en métricas relacionadas con el tiempo de asignación de VLANs tanto de manera estática como dinámicaen una SDN. La evaluación comparativa entre la asignación estática y dinámica de VLANs secentrará en el modo de configuración del puerto del switch, ya que el segundo modo dependerá del servidor NAC y no de la configuración manual del administrador de la SDN. Además, serealizará una prueba de concepto en un entorno de pruebas de SDN para evaluar la viabilidad de implementarla en una CAN real. Cabe mencionar que la topología se logrará mediante el uso de adaptadores USB-Ethernet, lo cual permitirá habilitar múltiples interfaces en las tarjetas RPi.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar e implementar un entorno de pruebas de redes definidas por software (SDN) para laasignación de VLANs en entornos universitarios, que fundamente su implementación en unaCAN desplegada.

1.4.2. Objetivos específicos

El presente trabajo tiene los siguientes objetivos específicos:

- Diseñar e implementar un entorno de pruebas para redes definidas por software, utilizando equipos tecnológicos que soporten controladores SDN de software libre.
- Diseñar e implementar un entorno de pruebas para una red sin SDN con asignación deVLANs.
- Diseñar e implementar un entorno de pruebas para SDN con asignación estática deVLANs.
- Diseñar e implementar un entorno de pruebas para SDN con asignación dinámica de VLANs con el uso de un servidor de control de acceso a la red, el cual permita realizarla asignación dinámica de VLANs según el host conectado a la red.
- Evaluar el funcionamiento y administración de VLANs con asignaciones dinámicas y

estáticas en redes definidas por software para seleccionar mejor la opción dentro de una CAN desplegada.

- Desarrollar una guía de procedimiento general para la actualización de asignación de VLANs de manera estática a dinámica en una CAN universitaria con SDN.

1.5. Estructura del documento

Este capítulo presentó la evolución del paradigma de redes de telecomunicaciones y las ventajas asociadas a las SDN y VLAN enfocadas en las redes de campus, presentando los retosy beneficios dentro de este nuevo enfoque basado en la capa de aplicación de una SDN. A continuación, se detalla la estructura del documento:

- **Capítulo 2:** expone los trabajos principales relacionados con el tema abordado y se analizan los resultados obtenidos por diversos autores para abordar la problemática planteada. Asimismo, se presentan las contribuciones de este trabajo de titulación en relación con el estado del arte expuesto.
- **Capítulo 3:** describe los fundamentos teóricos del nuevo paradigma SDN, abordando su conceptualización y principios clave. Además, se presentan los fundamentos teóricos delas VLAN, explicando su funcionamiento, características y su relevancia en la segmentación de redes. También se discuten los elementos que intervienen en la construcciónde entorno de pruebas utilizado en este trabajo.
- **Capítulo 4:** detalla de manera técnica la metodología seguida para llevar a cabo el diseño e implementación de VLAN en Entornos Universitarios Basados en Redes Definidas porSoftware (SDN).
- **Capítulo 5:** evidencia las pruebas de funcionamiento de las tres topologías implementadas en el desarrollo del presente trabajo de titulación y los resultados obtenidos en elentorno de pruebas.
- **Capítulo 6:** incluye las conclusiones a partir de los resultados obtenidos.
- **Anexos A-D:** se presenta evidencia gráfica de las pruebas de funcionamiento y la guía de procedimiento general para la actualización de asignación de VLAN de manera estáticaa dinámica en una CAN universitaria con SDN.

2. Estado del arte y trabajos relacionados

En este capítulo se realiza una revisión exhaustiva de los principales trabajos relacionados con los temas centrales abordados en este documento de titulación, como son la tecnología SDN yla implementación de VLAN para el control de acceso a la red. El objetivo de esta revisión delestado del arte es comprender el problema tratado y analizar las soluciones propuestas en los trabajos citados. Se han examinado diversos enfoques que implementan SDN y/o VLAN en entornos de pruebas utilizando hardware compatible con el protocolo OpenFlow. Esta revisiónproporciona una base sólida para contextualizar el presente trabajo y destacar su contribuciónen relación con las investigaciones previas.

2.1. Estudios sobre SDN y VLAN

En diversos trabajos se han presentado diferentes enfoques y diseños de redes para abordaraspectos relacionados con la autenticación de usuarios, el segmentado de la red y la aplicación de las ventajas proporcionadas por las redes definidas por software (SDN). Por ejemplo, en el trabajo realizado por los autores de [11], se propone un sistema de autenticación de usuariosbasado en el estándar IEEE 802.1x que incluye la asignación dinámica de VLAN mediante eluso de un servidor Cisco ACS y un servidor Token. Esta asignación se lleva a cabo en una red convencional sin controladores SDN, lo cual implica tener una infraestructura centralizada, y se obtuvieron resultados exitosos en diferentes escenarios de prueba con dos usuarios distintos. Los resultados demostraron que el servidor Cisco ACS permite a los usuarios autenticados acceder a la red y obtener posteriormente su VLAN correspondiente.

En [12], se propone el diseño de una red SDN con el objetivo de ofrecer una gestión centralizada de las configuraciones y una adecuada capacidad de escalabilidad en los nodos de la red de un proveedor de servicios. Se presenta un procedimiento para implementar una red SDNa gran escala, el cual puede ser aplicado también en un entorno universitario. Entre los resultados obtenidos, se destaca la automatización de procesos en las SDN, logrando tiempos deaprovisionamiento de servicios inferiores a dos segundos. Además, se observaron resultadossimilares durante el proceso de habilitación de nuevos nodos utilizando tanto redes tradicionales como SDN, lo que permitió reducir significativamente el tiempo requerido por el operador.

En [13], se implementa el estándar IEEE 802.1x en una red sin SDN, utilizando un servidor RADIUS para la asignación dinámica de VLAN. El sistema implementado logra incorporar funciones avanzadas de autenticación, autorización y seguridad. Además, según el perfil del usuario, el servidor RADIUS establece tanto la dirección IP como la configuración de VLAN, lo que garantiza que el usuario siempre obtenga la misma dirección IP y configuración de VLAN, sin importar la ubicación de la conexión.

En [14], se aborda la asignación estática de VLANs en una red SDN, junto con un proceso teórico de asignación de VLANs. Se propone un sistema que permite configurar VLANs de manera estática, proporcionando una vista de topología separada para cada VLAN. Los resultados obtenidos facilitan al administrador de la red monitorear todas las VLANs de manerasimultánea y sencilla.

En el artículo [15], se describe el diseño e implementación de un mecanismo de autenticacióny separación basado en SDN para usuarios de redes WiFi. Se propone un enfoque que permite separar el tráfico de datos entre diferentes proveedores de red dentro de un punto de acceso utilizando el protocolo OpenFlow. El sistema desarrollado es compatible con los protocolos deseguridad WLAN empresariales existentes. Los resultados obtenidos muestran que, mediante el uso de un conjunto de abstracciones, es posible dividir a los usuarios en función de la dirección MAC de sus dispositivos, lo que permite alojar dos redes dentro de un punto de accesoy proporcionar aislamiento entre ellas. En resumen, se logra una segregación de usuarios enredes WLAN empresariales.

En [16], se realiza un análisis de los dispositivos SDN Zodiac FX y Raspberry Pi (RPi). El objetivo de este estudio es determinar el rendimiento que pueden ofrecer estos dispositivos en redes de pequeña o mediana escala, como una Campus Area Network (CAN). Se utiliza el controlador Ryu en la implementación de la SDN, el cual se ejecuta en un ordenador con el sistema operativo Ubuntu. Los resultados obtenidos en este estudio demuestran que una Raspberry Pi puede desempeñarse de manera adecuada como un conmutador SDN.

En [17], se describe el diseño e implementación de un entorno de pruebas SDN utilizando Raspberry Pi como plataforma, Floodlight como controlador SDN y Open vSwitch (OvS) parapermitir que la tarjeta Raspberry Pi actúe como un conmutador que implementa el protocolo OpenFlow. Además, se evalúa el rendimiento del controlador mediante la aplicación MACtracker en la capa de aplicación. El entorno de pruebas ha sido capaz de realizar las funciones de una Software-Defined Networking (SDN), logrando identificar las

direcciones MAC de los computadores conectados a la SDN.

La configuración de sistemas basados en VLAN convencionales, siguiendo el estándar IEEE802.1Q, resulta tediosa y propensa a errores en entornos de redes de gran escala. En [18], sepresenta una propuesta de sistema VLAN flexible para una red de área campus (CAN) utilizando el protocolo OpenFlow. Además, se desarrolla un prototipo del sistema utilizando hardwarecompatible con SDN. Este sistema propuesto permite configuraciones de red flexibles y un control de acceso sofisticado, al mismo tiempo que simplifica considerablemente la administración de la red, solucionando los problemas asociados con la configuración estática de las VLAN.

También, existen diversos trabajos académicos que se centran en la implementación de SDNutilizando dispositivos de desarrollo de hardware, como la tarjeta RPi. Un ejemplo de ello es eltrabajo presentado en [19], donde se crea un entorno de pruebas para SDN utilizando tarjetasRPi modelo 3B como conmutadores de red, empleando el software de código abierto OvS y elcontrolador Flooding. En otro estudio, [20] también utiliza las tarjetas RPi para desarrollar SDN, pero en este caso se emplea el controlador Ryu. Por último, en [21], se diseña e implementa un prototipo de SD-WAN (Software-Defined Wide Area Network) utilizando el controlador ODL.

Varios estudios han abordado el análisis del rendimiento de las SDN, evaluando métricas queproporcionan información sobre cómo este tipo de redes funcionan en su operación. En [22],se analiza el rendimiento de una SDN al configurar Virtual Local Area Networks (VLANs). El estudio trabaja con métricas de calidad de servicio y tráfico de tipo Broadcast, HTTP, FTP y SMTP. Los resultados obtenidos revelan que al enviar tráfico a la red, todos los grupos de VLAN reciben el tráfico asignado de acuerdo con las políticas de calidad de servicio establecidas para cada uno (por ejemplo, 64 Kbps, 25 Mbps, 4 Mbps y Mbps para los grupos de Voz, Servidor,Facultad y Estudiantes, respectivamente), independientemente de si se encuentran dentro ofuera de la SDN. Esto demuestra que tanto la red SDN como la red VLAN imponen límites en el rendimiento. Además, durante las pruebas de captura por tipo de tráfico, se observó unrendimiento similar en redes con y sin SDN, lo que indica que la presencia de SDN no afectasignificativamente el rendimiento en estas pruebas ya que se han obtenido la misma tasa depaquetes.

En el estudio realizado por [23], se investiga el rendimiento de una red definida por software(SDN) en términos de throughput, latencia y jitter utilizando la herramienta `iperf` como generador de tráfico. Las pruebas realizadas comparan el rendimiento entre las

implementacionesbasadas en SDN y el enrutamiento de Linux puro, es decir, una red clásica. Los resultados revelan que el throughput en la SDN es ligeramente inferior al de la red clásica, pero la diferencia es mínima, con un throughput de 490 Mbps en la red SDN frente a 500 Mbps en la redclásica. En cuanto a la latencia, la red SDN muestra una menor latencia en comparación conla red clásica, con un pico máximo de latencia de 120 μs en la SDN frente a 155 μs en la red clásica. Sin embargo, al evaluar el jitter, se observa una diferencia más significativa, ya que laSDN presenta un peor desempeño con un jitter de 60 μs, mientras que la red clásica logra unjitter de 20 μs.

En [24], se analiza el rendimiento de una SDN en términos de calidad de servicio (QoS), utilizando el emulador Mininet y el controlador Ryu. La métrica principal evaluada fue la capacidadde la red al utilizar la herramienta `iperf` como generador de tráfico. Los resultados obtenidos demostraron la alta capacidad de la red implementada, siendo capaz de transferir 0.08 Gb/s de un host a otro en períodos cortos de tiempo. Esto indica un rendimiento eficiente y satisfactorio en términos de velocidad de transferencia de datos en la red SDN.

En el estudio realizado por [25], se evaluó el rendimiento de una SDN en función de la tasa depérdida de paquetes, los requisitos de ancho de banda y la latencia de la red. Para llevar a cabo el estudio, se utilizó el emulador Mininet, conmutadores Open vSwitch (OvS) y el controladorPOX. Los resultados obtenidos mostraron una notable reducción en la pérdida de paquetes alimplementar el controlador en comparación con el envío de tráfico solo a través de OvS, conun total de 7 % de paquetes perdidos en la configuración sin SDN frente al 2 % alcanzado en la configuración SDN. En términos de ancho de banda, se observó que este disminuía a medidaque aumentaba el número de hosts en la red. Por otro lado, al analizar la latencia, se encontróque la implementación de una SDN generaba una mayor latencia en comparación con una red clásica, independientemente del número de hosts presentes. En este estudio, se alcanzóun pico máximo de latencia de 25000 ms en la configuración SDN con 60 hosts conectados,mientras que en la red clásica la latencia alcanzó casi 500 ms.

3. Marco Teórico

3.1. SDN

La idea de las SDN es clara: en lugar de hacer cumplir políticas y protocolos dispersos en dispositivos individuales, la red se enfoca en el simple reenvío de información, mientras que el controlador de red toma las decisiones, lo que agiliza los flujos de información en la red [26, 27]. El controlador SDN proporciona un nuevo paradigma que permite a las aplicacionesinteractuar con la red a través de API abstractas declarativas, encargándose de administrar laconfiguración y el funcionamiento de la red. Además, se emplea una segunda API de consultapara solicitar información a la red con el objetivo de planificar y optimizar las operaciones [28].De esta manera, una SDN se puede definir sobre cuatro pilares fundamentales [29]:

- Los planos de control y datos están desacoplados. Se elimina la funcionalidad de controlde los dispositivos de red, que pasarán a ser simples elementos de reenvío de paquetes.
- Las decisiones de reenvío se basan en el flujo, en lugar de en el destino. En el contextoSDN/OpenFlow, un flujo es una secuencia de paquetes entre un origen y un destino.
- La lógica de control se traslada a una entidad externa, el llamado controlador SDN oSistema Operativo de Red (NOS).
- La red es programable mediante aplicaciones de software que se ejecutan sobre el NOS e interactúan con los dispositivos del plano de datos subyacente. Esta es una característica fundamental de SDN, considerada como su principal propuesta de valor.

De esta manera, una SDN tiene la capacidad de administrar de manera centralizada toda la red a través del controlador SDN, lo que permite programar y gestionar los recursos medianteaplicaciones SDN. Esto implica un control lógico y centralizado de los elementos de la red encargados de funciones como el transporte y procesamiento de datos.

3.1.1. Características principales

3.1.1.1. Programabilidad

SDN revoluciona la forma en que las aplicaciones de red son controladas al utilizar software de programación centralizado, independiente de los dispositivos de red que proporcionan conectividad física. Esto brinda a los administradores de red la capacidad de introducir nuevos servicios sin verse limitados por las plataformas existentes. Además, las aplicaciones SDN tienen la capacidad de automatizar las operaciones de los recursos de red de acuerdo a las necesidades específicas, liberando a los administradores de tareas repetitivas y permitiéndolesenfocarse en tareas más estratégicas [30].

3.1.1.2. Abstracción

Una SDN puede definirse mediante tres abstracciones fundamentales: reenvío, distribución yespecificación [29].

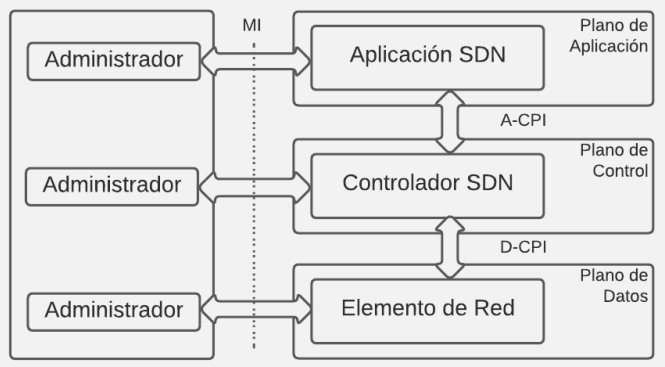
- La abstracción de reenvío en SDN permite que la aplicación de red, o programa de control, defina cualquier comportamiento deseado de reenvío de datos, al mismo tiempo que oculta los detalles del hardware subyacente. Esto brinda flexibilidad y permite que las decisiones de reenvío se tomen a nivel lógico, basadas en las políticas y necesidades específicas de la red, sin depender de las capacidades y limitaciones de los dispositivosfísicos involucrados.
- La abstracción de distribución protege a las aplicaciones SDN de los cambios del estadodistribuido, haciendo que el problema del control distribuido sea lógicamente centralizado. Su realización requiere una capa de distribución común, que en SDN reside en el NOS.
- La abstracción de especificación permite a una aplicación de red expresar el comportamiento de red deseado sin ser responsable de implementar ese comportamiento por sí misma. Esto puede lograrse mediante soluciones de virtualización, así como lenguajes de programación de red.

3.1.2. Arquitectura

La Open Networking Foundation (ONF) ha presentado una arquitectura de referencia para SDN que sigue un enfoque de tres capas, como se ilustra en la Figura 3.1. El objetivo principalde esta arquitectura es ofrecer una visión general de alto nivel de los puntos de

referencia y las interfaces abiertas que deben estar presentes en cualquier implementación de SDN. Esto garantiza un conjunto mínimo de capacidades necesarias para controlar la conectividadproporcionada por los recursos de red y dirigir el flujo de tráfico a través de ellos. El propósito de esta arquitectura es establecer un marco común que permita la interoperabilidad y la adopción más amplia de SDN en diferentes entornos [1].

Figura 3.1: Arquitectura SDN. Extraído de [1].



3.1.2.1. Plano de aplicación

La capa de aplicación, como su nombre lo indica contiene aplicaciones y servicios que definen, supervisan y controlan los recursos y el comportamiento de la red [31]. Las aplicaciones SDNcomunican sus requisitos a la red a través de una API que conecta con la capa de control y están diseñadas para satisfacer las necesidades de los usuarios. De esta manera, a la capade aplicación la componen las aplicaciones de usuarios [27].

3.1.2.2. Plano de control

El plano de control SDN se encarga de convertir las solicitudes provenientes de la capa de aplicación en comandos específicos para los conmutadores del plano de datos. El plano de control puede considerarse el cerebro de la red, puesto que toda la lógica de control descansaen las aplicaciones y controladores que forman el plano de control [29, 31]. La capa de controlposee las siguientes funciones:

- **Soporte de aplicaciones:** permite que las aplicaciones SDN accedan a la informaciónde la red y programen el comportamiento de la red de manera específica para cada aplicación. Esto se logra a través de una interfaz especializada conocida como la Interfaz deControl de APIs, que brinda a las aplicaciones un medio para

interactuar con el controlador de la red. A través de esta interfaz, las aplicaciones pueden obtener datos sobre elestado de la red, enviar comandos y configuraciones, y adaptar el comportamiento de lared de acuerdo a sus necesidades.

- **Instrumentación:** permite supervisar y gestionar tanto la topología de red física como lavirtual, los elementos de red y el tráfico que fluye a través de ellos. Esta capacidad de interactuar con las funciones de gestión multicapa facilita la administración y el control de diversas operaciones relacionadas con la aplicación SDN.
- **Abstracción:** proporciona una abstracción de los recursos, capacidades y características de la red, para respaldar la administración y orquestación de los recursos de la redfísica y virtual.

3.1.2.3. Plano de datos

El plano de datos en una red SDN se refiere a los componentes de red, como los hosts, conmutadores/enrutadores físicos y virtuales, y los medios de transmisión, que desempeñan el papel fundamental de reenviar los datos. En este contexto, los conmutadores SDN son los elementos de reenvío específicos del plano de datos. Estos dispositivos son responsables derecopilar información sobre el estado de la red, como la topología y las estadísticas de tráfico,y transmitirla al controlador. A su vez, el controlador se encarga de proporcionar instrucciones a los conmutadores sobre cómo deben reenviar los paquetes de datos según las reglas establecidas [27].

Los dispositivos que se ubican en el plano de datos son capaces de soportar las siguientes funciones [31]:

- **Función de soporte de control:** admite la capacidad de programación de las funcionesde la capa de recursos a través de la interfaz de control de recursos.
- **Función de reenvío de datos:** se encarga de gestionar los flujos de datos que deben ser enviados a través de las diferentes rutas establecidas según los requerimientos de las aplicaciones SDN. Esta función es proporcionada por la capa de control SDN, lo quepermite minimizar la funcionalidad de reenvío de datos en la capa de recursos de la red.

3.1.2.4. Interfaces

El controlador SDN se divide en dos interfaces principales: la interfaz Northbound (NBI), quese conecta al plano de gestión y permite la interacción con aplicaciones y servicios externos alcontrolador, y la interfaz Southbound (SBI), que se comunica con el plano de datos y establecela comunicación con los dispositivos de red. Además de estas interfaces, también existen las interfaces Eastbound (EBI) y Westbound (WBI), diseñadas para facilitar la comunicación conotros controladores en la red. Estas interfaces permiten la agrupación de controladores y la colaboración entre diferentes dominios de red, ampliando las capacidades y la escalabilidadde la red SDN [32].

- **Interfaz Southbound:** Formaliza el modo en que los elementos de control y el plano dedatos se conectan, permitiendo al controlador comunicarse y programar la lógica en el hardware de los dispositivos de red. A través de la interfaz Southbound, el controlador SDN puede enviar instrucciones y recibir información sobre el estado de la red, lo que le permite tomar decisiones y controlar de manera centralizada el funcionamiento de losdispositivos de red [29].
- **Interfaz Northbound:** Permite el intercambio de datos entre el controlador y la aplicación. El tipo de información que se intercambia, su forma y su frecuencia depende de cada aplicación [33].
- **Interfaces Eastbound y Westbound:** Permite la interconexión de redes convencionales con las SDN. Además, sirven como conducto de información entre varios planos decontrol de diferentes dominios de SDN [33].

3.1.3. Protocolo OpenFlow

OpenFlow es un *framework* desarrollado originalmente en la Universidad de Stanford y actualmente se encuentra en desarrollo de estándares activos para este protocolo, bajo la supervisión de la Open Networking Foundation (ONF). Este protocolo define una forma de comunicación entre un controlador centralizado lógicamente y un conmutador OpenFlow. A través delprotocolo OpenFlow, el controlador puede enviar mensajes al conmutador para programarlo y tener un control detallado sobre la conmutación del tráfico de usuarios. El protocolo constade un conjunto de mensajes que se intercambian entre el controlador y el conmutador, permitiendo al controlador definir, modificar y eliminar flujos de manera

programática. Además, el protocolo OpenFlow se implementa comúnmente sobre TLS (*Transport Layer Security*), lo que garantiza un canal seguro para la comunicación entre el controlador y el conmutador. Dentrode la arquitectura del conmutador OpenFlow, se pueden encontrar diferentes tipos de tablas,como las tablas de flujo y las tablas de grupo, que permiten realizar diversas operaciones deenrutamiento y manipulación del tráfico [2, 31, 34].

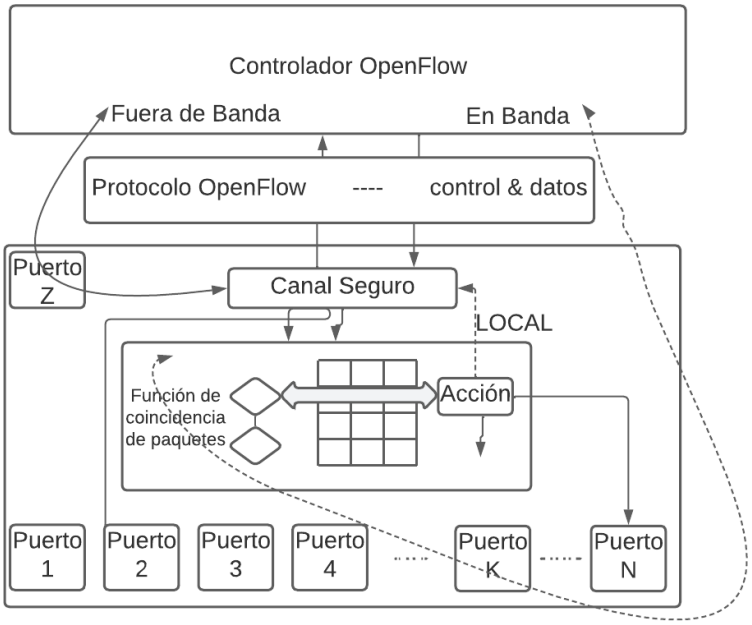


Figura 3.2: Componentes de un conmutador OpenFlow. Extraído de [2].

3.1.3.1. Tablas de flujo

Las tablas de flujo desempeñan un papel similar a las tablas de direcciones MAC en un conmutador tradicional, donde se almacenan las direcciones de hardware de los hosts. En el contexto de un conmutador SDN, las tablas de flujo almacenan entradas o flujos que contienen instrucciones sobre cómo deben tratarse los paquetes cuando llegan al conmutador. Cada paqueteque ingresa al conmutador atraviesa una o más tablas de flujo, cada una compuesta por filasllamadas entradas. Estas entradas están formadas por siete componentes definidos en la lista[31, 35]:

- **Campos de coincidencia:** se utiliza para seleccionar paquetes que coincidan con losvalores de los campos especificados en la API programada.
- **Prioridad:** indica la prioridad relativa de las entradas de la tabla. Este es un campo

de16 bits, donde 0 corresponde a la prioridad más baja pudiendo haber 64000 niveles deprioridad.

- **Contadores:** la especificación OpenFlow define una variedad de contadores, los cualesproporcionan estadísticas del flujo.
- **Instrucciones:** empleadas y aplicadas al conmutador configurado en la tabla de flujo almomento que se produce una coincidencia.
- **Tiempos de espera:** cantidad máxima de tiempo de inactividad antes de que el conmutador caduque un flujo.

  Cookie: valor de datos de 64 bits que puede ser utilizado por el controlador para filtrarestadísticas de flujo, modificación de flujo y eliminación de flujo. No se utiliza al procesarpaquetes.

  Indicadores: alteran la forma en que se gestionan las entradas de flujo.

La estructura de una tabla de flujo se indica en la Figura 3.3

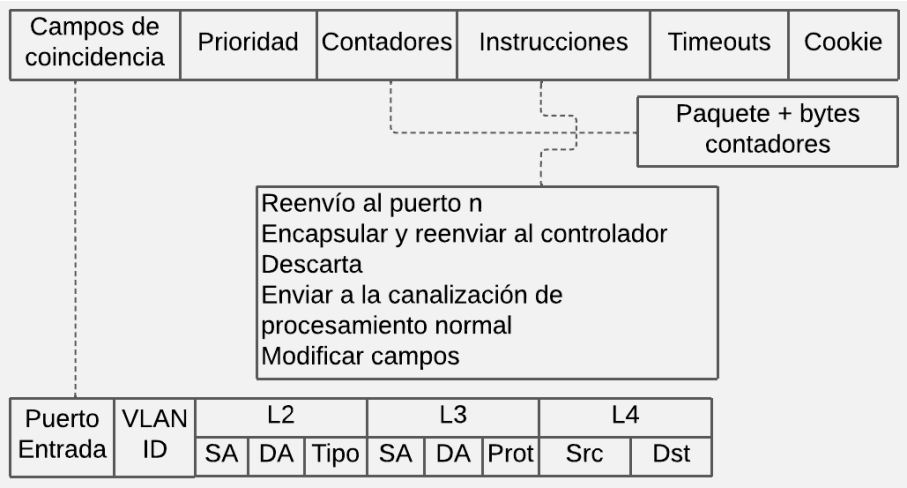


Figura 3.3: Estructura de una tabla de flujo. Extraído de [3].

3.1.3.2. Tablas de grupo

La tabla de grupos permite que el protocolo OpenFlow represente un conjunto de puertos como una sola entidad para reenviar paquetes. Al igual que las tablas de flujo, las tablas de grupo constan de varios componentes (Figura 3.4), estos se listan a continuación:

- **Identificador de grupo:** un número entero de 32 bits sin signo que identifica de formaúnica al grupo. Un grupo se define como una entrada en la tabla de grupos.
- **Tipo de grupo:** es utilizado para determinar la semántica del grupo.

- **Contadores:** es un campo que aumenta su valor cuando los paquetes son procesadospor un grupo.
- **Cubos de acción (buckets):** es una lista ordenada de cubos de acción, donde cadacubo de acción contiene un conjunto de acciones para ejecutar y parámetros asociados.

Figura 3.4: Estructura de una tabla de grupo.

3.1.3.3. Mensajes OpenFlow

La comunicación entre el controlador y el conmutador se lleva a cabo a través de un canal seguro establecido mediante una conexión inicial de TLS sobre TCP. Si el conmutador tieneconocimiento de la dirección IP del controlador, será el encargado de iniciar esta conexión. Cada mensaje intercambiado entre el controlador y el conmutador comienza con un encabezado OpenFlow que contiene información como la versión de OpenFlow, el tipo de mensaje, la longitud del mensaje y el ID de transacción del mensaje. Estos mensajes se clasifican en tres categorías generales: mensajes de controlador a conmutador, mensajes asíncronos y mensajes simétricos [2].

- **Controlador a conmutador:** estos mensajes permiten al controlador gestionar el estado lógico del conmutador, incluyendo su configuración y los detalles de las entradas en latabla de flujos y grupos. Entre estos mensajes se encuentra el mensaje Packet-out, el cual es enviado por el controlador al conmutador cuando este último envía un paquete alcontrolador y, en lugar de descartarlo, el controlador decide direccionarlo hacia un puerto de salida específico en el conmutador.
- **Asíncrono:** este tipo de mensajes son enviados por el conmutador sin que el controlador los solicite. Uno de estos mensajes es el Packet-in, el cual es utilizado por el conmutador para enviar un paquete al controlador cuando no se encuentra una coincidencia en la tabla de flujos. De esta manera, el conmutador informa al controlador sobre la presencia de un paquete que requiere decisiones adicionales por parte de la lógica de controlcentralizada.
- **Simétricos:** son simples pero útiles, y consisten en mensajes de saludo que

generalmente se envían de ida y vuelta entre el controlador y el conmutador al establecer la conexión inicial. Además, los mensajes de solicitud y respuesta de eco pueden ser utilizados por el conmutador o el controlador para medir la latencia o el ancho de banda dela conexión.

En la Tabla 3.1, se resume los tipos de mensajes OpenFlow.

Tipo de Mensaje	Categoría	Subcategoría
HELLO	Simétrico	Inmutable
ECHO_REQUEST	Simétrico	Inmutable
ECHO_REPLY	Simétrico	Inmutable
VENDOR	Simétrico	Inmutable
FEATURES_REQUEST	Controlador-*switch*	Configuración *switch*
FEATURES_REPLY	Controlador-*switch*	Configuración *switch*
GET_CONFIG_REQUEST	Controlador-*switch*	Configuración *switch*
GET_CONFIG_REPLY	Controlador-*switch*	Configuración *switch*
SET_CONFIG	Controlador-*switch*	Configuración *switch*
PACKET_IN	Asíncrono	NA
FLOW_REMOVED	Asíncrono	NA
PORT_STATUS	Asíncrono	NA
ERROR	Asíncrono	NA
PACKET_OUT	Controlador-*switch*	Cmd desde *switch*
FLOW_MOD	Controlador-*switch*	Cmd desde *switch*
PORT_MOD	Controlador-*switch*	Cmd desde *switch*
STATS_REQUEST	Controlador-*switch*	Estadísticas
STATS_REPLY	Controlador-*switch*	Estadísticas
BARRIER_REQUEST	Controlador-*switch*	Barrera
BARRIER_REPLY	Controlador-*switch*	Barrera
QUEUE_GET_CONFIG_REQUEST	Controlador-*switch*	Configuración de cola
QUEUE_GET_CONFIG_REPLY	Controlador-*switch*	Configuración de cola

Tabla 3.1: Estructura de una tabla de grupo. Extraído de [2].

3.1.4. Controladores SDN

Un controlador SDN, puede interpretarse como Network Operating System (NOS), el cual toma el papel de cerebro de la red. Al igual que con un sistema operativo convencional, un NOSproporciona servicios esenciales, interfaces de programación de aplicaciones y una abstracción de elementos de capa inferior. Las funciones de una SDN permiten a los desarrolladores definir políticas de red y administrar redes sin preocuparse por las características del dispositivo de red, que pueden ser heterogéneos y dinámicos. En las siguientes subsecciones se describen algunos controladores destacados.

3.1.4.1. OpenDaylight

OpenDaylight Controller es un controlador basado en Java que utiliza YANG como lenguaje de modelado para varios aspectos del sistema y las aplicaciones. Puede implementarse como un único controlador centralizado, pero también permite la distribución de controladores en escenarios donde se ejecutan una o varias instancias en servidores agrupados en la red. OpenDaylight Controller proporciona subsistemas basados en modelos que sirven como base para las aplicaciones Java. Estos subsistemas ofrecen funcionalidades esenciales para la gestión y control de la red, permitiendo a las aplicaciones desarrolladas sobre la plataforma aprovechar estas capacidades [36].

- **Subsistema de configuración:** es un marco integral que engloba la activación, inyección de dependencias y configuración de los componentes del sistema. Proporciona un entorno flexible y potente para gestionar los compromisos de configuración necesariospara el funcionamiento óptimo de los diversos elementos del sistema.
- **MD-SAL:** es la funcionalidad de mensajería y almacenamiento de información para da- tos, notificaciones y Remote Procedure Call (RPC) modelados por desarrolladores de aplicaciones. MD-SAL utiliza YANG como modelo para la interfaz y las definiciones de datos, y proporciona un tiempo de ejecución centrado en datos y mensajería para dichosservicios basado en el modelo YANG.
- **Agrupación en cluster MD-SAL:** permite la compatibilidad con clústeres para la funcionalidad principal de MD-SAL y proporciona accesos transparentes de

ubicación a losdatos modelados YANG.

El controlador OpenDaylight admite el acceso externo a aplicaciones y datos utilizando los siguientes protocolos basados en modelos:

- **NETCONF:** protocolo RPC basado en XML. Proporciona capacidades para que el cliente invoque RPC modeladas en YANG, reciba notificaciones, lea, modifique y manipule datos modelados en YANG.
- **RESTCONF:** protocolo basado en HTTP, que proporciona Application Programming Interfaces (API) similares a REST para manipular datos modelados YANG e invocar RPCmodelados YANG, utilizando XML o JSON como formato de carga útil.

3.1.4.2. Floodlight

Floodlight es un paquete de código abierto desarrollado por Big Switch Networks, que utilizaApache Ant como herramienta de construcción de software para facilitar y flexibilizar su desarrollo. Este proyecto cuenta con una comunidad activa y ofrece una amplia gama de funciones que se pueden incorporar según los requisitos específicos de una organización. Floodlight brinda opciones de interfaz gráfica de usuario tanto web como Java, y la mayoría de su funcionalidad se expone a través de una API REST, permitiendo una fácil integración con otros sistemas y aplicaciones [31].

3.1.4.3. Ryu

Ryu es un marco SDN basado en componentes de código abierto, desarrollado completamente en Python por NTT Labs. Se trata de un controlador SDN que ofrece componentes de softwarey API bien definidas, lo que facilita a los desarrolladores la creación de nuevas aplicaciones de gestión y control de redes. Esta estructura basada en componentes permite a las organizaciones personalizar las implementaciones de acuerdo con sus necesidades específicas.Los desarrolladores tienen la flexibilidad de modificar los componentes existentes o implementar sus propios componentes de manera rápida y sencilla, asegurando que la red subyacentepueda adaptarse a las demandas cambiantes de las aplicaciones [37].

3.1.4.4. POX

El controlador POX es una implementación de código abierto de OpenFlow desarrollada porvarios expertos y profesionales de SDN. Este controlador ofrece una interfaz gráfica de usuario basada en web y está escrito en Python, lo que acelera los ciclos de experimentación y desarrollo en comparación con otros lenguajes como C++ [38]. POX permite una serie de funcionalidades, como la instalación de reglas de inundación con comodines en los conmutadores, el comportamiento de los conmutadores OpenFlow como conmutadores de aprendizaje L2, la manipulación y construcción de solicitudes y respuestas Address Resolution Protocol (ARP),el descubrimiento de la topología de la red completa, y la configuración de reglas de reenvíobasadas en direcciones IP, entre otros.

3.1.5. Selección Controlador SDN

En la actualidad, hay una amplia variedad de controladores SDN disponibles para implementar el protocolo OpenFlow, permitiendo el control y la administración de conmutadores en los planos de datos y control. Después de comparar varios controladores SDN presentados previamente, se ha elegido OpenDaylight (ODL) como el controlador SDN para este proyecto detitulación. Esta elección se basa principalmente en las características que ODL ofrece en términos de abstracción, programabilidad y apertura, allanando el camino hacia una infraestructura inteligente definida por software. Además, ODL facilita el desarrollo de una plataforma abiertaque puede migrar redes a SDN y marcar el comienzo de una infraestructura de red abierta einteligente [39]. Entre las capacidades clave que proporciona ODL para mejorar la visibilidady el control se incluyen:

- Topología y estado centralizados lógicamente para los recursos de red físicos y virtuales, lo que permite una visibilidad sin precedentes en todo el dominio.
- Capacidades de supervisión no disruptivas que no afectan a las comunicaciones esenciales para el funcionamiento y la continuidad de un *software*.
- Capa de abstracción de servicios basada en modelos (MD-SAL) que aprovecha los modelos YANG estándar del sector para asignar aplicaciones de red a los dispositivos subyacentes.
- Una interfaz modular plug-in hacia el sur (de controlador a dispositivo) con amplia compatibilidad con interfaces de gestión de red estándar (OpenFlow) e interfaces y dispositivospropietarios.

- Interfaces hacia el norte basadas en intenciones (de aplicación a controlador) que exponen las capacidades de SDN a diversas aplicaciones de red, al tiempo que abstraen losdetalles de la infraestructura subyacente.
- Soporta fácilmente servicios de red propietarios y extendidos, para soportar visibilidad yanálisis en todo el dominio para gestionar dominios virtuales y físicos.

3.2. VLAN

Las redes de área local virtual (VLAN) se utilizan para crear múltiples LAN virtuales en una infraestructura de red, abordando problemas como la falta de aislamiento del tráfico, la utilización ineficiente de los conmutadores y la gestión de usuarios [40]. Las VLAN permiten queel tráfico de difusión, como los mensajes ARP y DHCP, no atraviese toda la red, limitando sualcance a la VLAN específica. Esta técnica de limitación del tráfico de difusión también brinda una opción para mejorar la seguridad y confidencialidad de los datos transmitidos en unared corporativa, evitando que el tráfico de datos alcance a determinados grupos de equipos conectados a la red.

La implementación de VLAN permite a los conmutadores optimizar sus recursos al permitir que los puertos sean utilizados por diversos tipos de usuarios y departamentos en una mismaubicación física, lo que resulta en una mayor eficiencia. Además, la gestión de usuarios se beneficia ampliamente con el uso de VLAN, ya que se utilizan en redes empresariales y de campus para asignar direcciones IP de la misma subred a usuarios, incluso si no están conectados al mismo conmutador. Asimismo, las VLAN simplifican la asignación de direcciones endiferentes departamentos administrativos y permiten tratar a usuarios ubicados en diferentes áreas físicas como una unidad lógica. Sin embargo, es importante tener en cuenta que la configuración de VLAN es un proceso manual que requiere una inversión significativa de tiempo[41].

A nivel de configuración, es importante que los administradores de la CAN tengan en cuenta las interdependencias existentes en toda la red. Incluso una modificación sencilla, como agregarun host a una VLAN, puede implicar la reconfiguración de conmutadores en la red,especialmente aquellos que actúan como enlaces troncales [42].

Los usuarios de extremo y los conmutadores pueden ser asignados a diferentes VLAN al modificar la configuración del puerto o interfaz del dispositivo de conmutación según el

estándar802.1Q al que esté conectada la estación de extremo o el conmutador. Este enfoque, junto con la forma en que se determina la dirección MAC, asegura que solo los usuarios de extremo pertenecientes a la misma VLAN puedan comunicarse entre sí, creando así una Red PrivadaVirtual (RPV). Para lograr la separación del tráfico de las tramas que pertenecen a diferentes VLAN en una infraestructura compartida, se inserta una etiqueta con un ID de VLAN (VID) encada trama. Se debe asignar un VID único a cada VLAN (del 1 al 4096) dentro de la misma infraestructura física, asegurando su unicidad a nivel mundial [43].

A continuación, se listan los tipos de Virtual Local Area Network (VLAN) que se puede configurar en un conmutador:

- **VLAN por puerto:** este tipo de VLAN permite a un grupo de puertos del conmutadorformar parte de una sola VLAN.
- **VLAN por Media Access Control (MAC):** este tipo de política se basa en la dirección MAC del equipo que se conecta al conmutador. Realiza un mapeo para determinar la VLAN con el uso de un software que administre las mismas.
- **VLAN por protocolo:** este tipo de política asigna una VLAN a los equipos que esténenrutados con un mismo protocolo IP, usando el encabezado de la capa de red.
- **VLAN por usuario:** esta política combina las políticas anteriores. Se crea una VLAN enbase al puerto, dirección MAC, protocolo, etc.

3.2.1. Estructura 802.1q

El estándar IEEE 802.1q es parte de la familia de estándares IEEE 802 y proporciona una especificación para la implementación VLAN. Su objetivo es permitir la segmentación de redes en múltiples dominios de difusión. Para lograr esto, IEEE 802.1q utiliza un mecanismo de etiquetado interno que inserta un campo de etiqueta de 4 bytes en la trama Ethernet original,situado entre los campos de Dirección de origen y Tipo/Longitud. Esta etiqueta identifica la pertenencia de la trama a una VLAN específica y facilita la gestión y el control del tráfico en lared [44].

3.2.1.1. Cabecera 802.1q

La cabecera 802.1q consta de 32 bits de longitud; en este espacio se encuentra toda la información necesaria para identificar correctamente la VLAN de la trama y garantizar que llega aldestino correcto [43]. En la Figura 3.5, se muestra todos los campos que contiene una cabecera 802.1q.

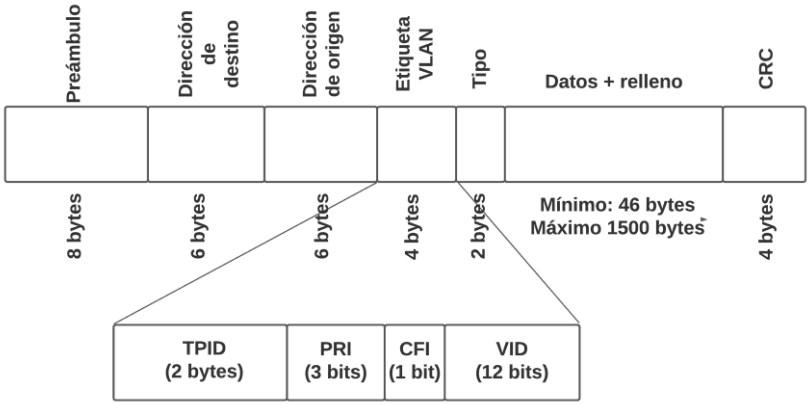


Figura 3.5: Estructura trama 802.1q. Extraido de [4].

3.2.1.2. Campo identificador de protocolo de etiqueta (TPID)

El campo TPID de 16 bits indica si una trama está etiquetada como VLAN. Por defecto, el valor TPID es 0x8100, lo que indica que la trama está etiquetada como VLAN. El dispositivo determina si una trama recibida lleva una etiqueta VLAN comprobando el valor TPID [45].

3.2.1.3. Campo prioridad

También conocido como prioridad de usuario, este campo de 3 bits hace referencia a la prioridad IEEE 802.1p. El campo indica el nivel de prioridad de la trama que puede utilizarse parala priorización del tráfico. El campo puede representar 8 niveles (de 0 a 7) [44].

3.2.1.4. Campo indicador de formato canónico (CFI)

El campo CFI de 1 bit indica si las direcciones MAC están encapsuladas en formato estándarcuando los paquetes se transmiten a través de distintos medios. Un valor de 0 indica quelas direcciones MAC están encapsuladas en formato estándar. Un valor de

1 indica que las direcciones MAC están encapsuladas en un formato no estándar. El valor de este campo es 0por defecto [45].

3.2.1.5. Campo identificador VLAN

Este ID es único para cada VLAN y ayuda a identificar a qué VLAN pertenece la trama actual.Es un campo de 12 bits que especifica la VLAN a la que pertenece la trama. Los valores 0 y 4095 (0x000 y 0xFFF en hexadecimal) están reservados. Todos los demás valores puedenutilizarse como identificadores de VLAN, permitiendo hasta 4.094 VLANs. El valor reservado0x000 indica que la trama no lleva un identificador VLAN. En los puentes, el valor VID 0x001(el identificador VLAN por defecto) suele reservarse para una VLAN de gestión de red; esto es específico del proveedor [46].

3.2.2. VLAN sobre SDN

3.2.2.1. VLANs estáticas sobre SDN

El funcionamiento de las VLAN en una red definida por software (SDN) se basa en el protocolo OpenFlow. Los conmutadores de la red deben ser compatibles con este protocolo para operar correctamente. En el plano de datos, un conmutador OpenFlow realiza un reenvío depaquetes relativamente sencillo. Cuando recibe un paquete, verifica la entrada de reenvío correspondiente. Si esta entrada contiene información de flujo, el paquete se reenvía según lo especificado. En caso de que no haya información de flujo disponible, el paquete se envía alcontrolador OpenFlow. En el plano de control, el controlador OpenFlow tiene una visión globalde todos los conmutadores de la red SDN y ejerce control sobre ellos. Es el controlador quientoma decisiones sobre el reenvío de paquetes, basándose en el puerto o la VLAN, y actualizala entrada de reenvío en el conmutador de destino correspondiente [47].

3.2.2.2. VLAN dinámicas sobre SDN

En este nuevo enfoque, se produce un cambio de paradigma en la asignación de VLAN, donde el controlador utiliza la capa de aplicación de SDN para consultar la base de datos de usuariosdel servidor al que está conectado. Esta base de datos contiene información sobre los usuarios de toda la red, lo que permite determinar la VLAN a la que pertenecen, como

por ejemplo, docentes, estudiantes, personal administrativo, entre otros. Una vez que el controlador SDNrecibe la respuesta del servidor de la base de datos con la VLAN asignada al usuario, la aplicaal puerto correspondiente, lo que permite llevar a cabo el proceso de VLAN estáticas en un entorno SDN.

El uso de VLAN dinámicas proporciona varios beneficios, ya que permite asignar el tráfico a diferentes VLAN, lo cual incrementa la seguridad, la implementación de políticas, la contabilidad, el monitoreo y la flexibilidad. Es posible conectar diferentes dispositivos finales al mismo puerto en momentos distintos y asignarlos automáticamente a VLAN diferentes sin necesidadde reconfigurar el puerto en cuestión [47].

3.3. NAC

El Control de Acceso a la Red (NAC) permite a los administradores de red establecer un control de acceso coherente. Se trata de un concepto de red que permite identificar usuarios y dispositivos mediante la aplicación de métodos de autenticación, y controlar el acceso a los recursos de la red a través de autorización y la aplicación de políticas. Existen diversas formasde implementar un NAC en una CAN, y se deben tomar decisiones durante las fases de diseño e implementación, las cuales estarán influenciadas por los acuerdos de nivel de servicio(SLAs) específicos establecidos entre la CAN y sus usuarios [48].

3.3.1. Funciones

Las funciones principales de un Network Access Control (NAC) se pueden resumir en [48]:

- **Control de preadmisión:** bloquea cualquier mensaje no autenticado para que no llegue a la red.
- **Detección de dispositivos y usuarios:** identifica usuarios y dispositivos con credenciales predefinidas o ID de máquinas.
- **Autenticación y autorización:** verifica y proporciona acceso.
- **Incorporación:** aprovisiona un dispositivo con software de seguridad, administración overificación de host antes de permitir el acceso a la red.
- **Creación de perfiles:** escanea dispositivos de punto final en función de un conjunto

específico de propiedades, o un perfil, definido por el personal de seguridad.

- **Cumplimiento de políticas:** aplica acceso basado en roles y permisos, implementadoen firewalls de capa 3 o enrutadores seguros.
- **Control posterior a la admisión:** hace cumplir la finalización y limpieza de la sesión.

3.4. Open vSwitch

Open vSwitch (OvS) es un conmutador de software que se encuentra en el nivel del hipervisor o dominio de administración. Es un software de código abierto distribuido bajo la licenciaApache 2.0 y tiene como objetivo proporcionar una plataforma de conmutación que permita utilizar funciones programables y de reenvío en la red. OvS convierte la plataforma en la quese ejecuta en un conmutador que implementa la conmutación Ethernet estándar, incluyendo características como VLAN, RSPAN y ACL. Aunque funciona como un conmutador básico decapa 2, OvS también permite la integración de entornos virtuales y exporta interfaces para manipular el estado de reenvío [49].

3.4.1. Interfaces

La inclusión de interfaces para la gestión global de la configuración y el estado de reenvío permite la distribución de las funciones del conmutador en varios servidores, lo que desacoplade forma efectiva la topología de la red lógica de la física.

- **Configuración:** A través la interfaz de configuración, es posible activar la duplicación de puertos, aplicar políticas de QoS a las interfaces, habilitar el registro de NetFlow parauna máquina virtual y combinar interfaces para mejorar el rendimiento y la disponibilidad. Esta interfaz también facilita la administración remota de la configuración y establece conexiones entre los puertos de red y el entorno virtual más amplio.
- **Ruta de reenvío:** Esta interfaz permite que un proceso externo escriba en la tabla de reenvío directamente, especificando cómo se manejan los paquetes en función de susencabezados de capa 2, capa 3 y capa 4. La interfaz de ruta de reenvío implementa unsuperconjunto del protocolo OpenFlow.
- **Gestión de conectividad:** Es una interfaz de administración local a través de la cualla capa de virtualización puede manipular su configuración topológica. Esto

incluye la creación de conmutadores, la gestión de la conectividad Virtual Interface (VIF) y Physical Interface (PIF) [49].

3.4.2. Funciones

OvS es un software escrito en Lenguaje C en su mayor parte. Es multiplataforma, lo que permite la compatibilidad con varias tecnologías de virtualización basadas en Linux, incluidas KVM yVirtualBox. La versión 2.17.5 LTS brinda las siguientes funciones [50]:

- Modelo estándar de VLAN 802.1q con puertos troncales y de acceso.
- Vinculación de NIC con o sin LACP en el conmutador de flujo ascendente.
- NetFlow y sFlow(R).
- Configuración de Quality of Service (QoS) incluido monitoreo.
- Túneles a través de Geneve, GRE, VXLAN, STT y LISP.
- Gestión de fallos de conectividad 802.1ag.
- OpenFlow y sus extensiones.
- Base de datos de configuración transaccional con conectores a C y Python.
- Reenvío de alto rendimiento mediante un módulo de kernel de Linux, soporte para Linux

  3.10 o superior.

3.4.3. Características y consideraciones de diseño

3.4.3.1. La movilidad del estado

OvS ofrece soporte tanto para la configuración como para la migración entre instancias, lo que permite no solo migrar la configuración asociada, como reglas SPAN, ACL y QoS, sino también cualquier estado de red en tiempo real. Además, el estado de OvS se almacena y respalda utilizando un modelo de datos concreto, lo que facilita el desarrollo de sistemas de automatización estructurados y organizados. [51].

3.4.3.2. Respuesta a redes dinámicas

OvS ofrece una variedad de funciones que permiten a un sistema de control de red

responder y adaptarse a medida que la topología de la red cambia. Esto incluye características como lacontabilidad y visibilidad mediante NetFlow, IPFIX y sFlow. Además, OvS cuenta con una base de datos de estado de red (OVSDB) que admite activadores remotos, lo que facilita la gestión y control remoto de la red. También es compatible con OpenFlow, un método de exportaciónremota que permite el control del tráfico de forma remota [51].

3.4.3.3. Mantenimiento de etiquetas lógicas

OvS ofrece diversas opciones para definir y gestionar las reglas de etiquetado. Estas reglas pueden almacenarse de manera eficiente, evitando la necesidad de utilizar dispositivos de red complejos. Además, OvS tiene la capacidad de manejar miles de túneles GRE simultáneosy permite su configuración remota, lo que facilita la creación, configuración y eliminación de túneles de forma remota [51].

3.4.3.4. Integración de hardware

La ruta de reenvío de OvS, también conocida como ruta de datos en el kernel, está diseñada para aprovechar el procesamiento de paquetes en conjuntos de chips de hardware, ya sea enun chasis de conmutador tradicional o en una tarjeta de interfaz de red del host. Esto posibilita que la ruta de control de OvS funcione tanto con una implementación de software puro como con un conmutador de hardware, brindando flexibilidad en las opciones de implementación [51].

3.4.4. Arquitectura

La arquitectura de OvS se divide en dos grupos principales: el grupo de gestión y el grupo decontrol. En el grupo de gestión se encuentran los gestores que utilizan el protocolo OVSDB para administrar las instancias de OvS. Por otro lado, en el grupo de control se encuentran loscontroladores que utilizan el protocolo OpenFlow para instalar y mantener el estado de reenvío en los conmutadores OpenFlow. [51].

Como se muestra en la Figura 3.6, OvS se ejecuta tanto en el kernel como en el espaciodel usuario. Los componentes que se encargan del reenvío de paquetes son: ovs-vswitchd ydatapath kernel. El primero de estos es un daemon que se ejecuta en el espacio

de usuario,por lo general es el sistema operativo y entorno operativo en donde está instalado OvS. El segundo componente generalmente está escrito en el sistema operativo del host.

En el entorno físico, se encuentran las tarjetas de interfaz de red a las cuales se conectan los usuarios, tanto físicos como virtuales. Cuando un usuario envía un paquete a través de

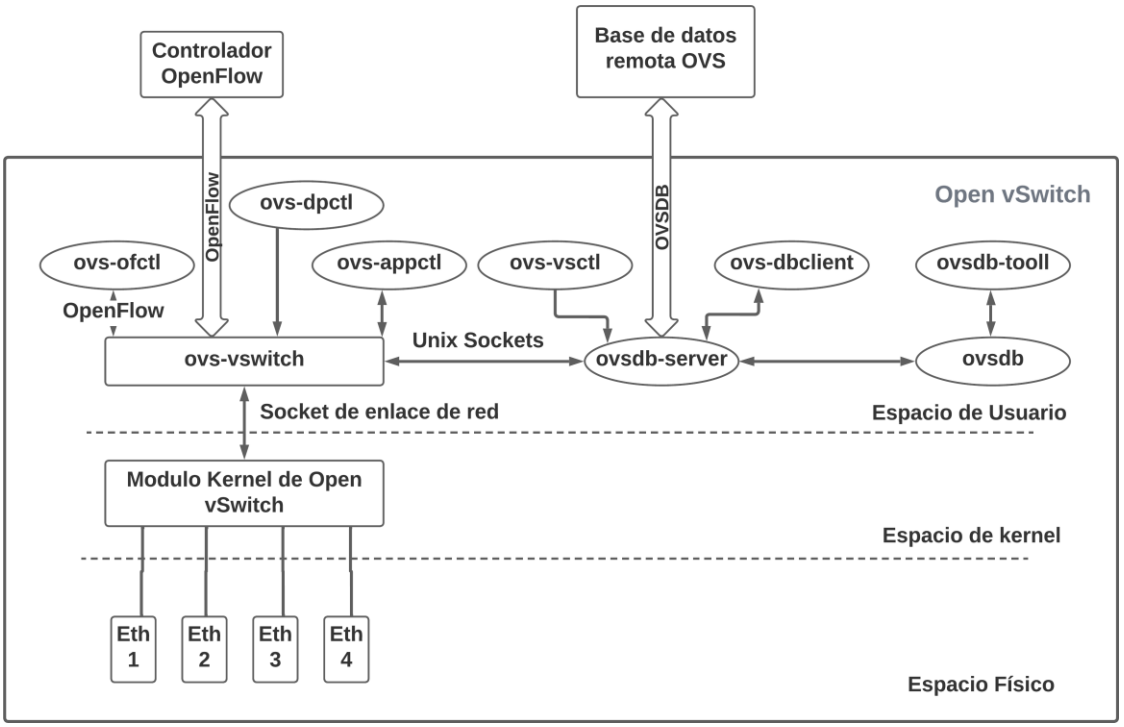


Figura 3.6: Arquitectura de OvS. Extraído de [1].

la interfaz de red, éste llega al módulo correspondiente en el kernel de OvS, el cual sigue lasinstrucciones, llamadas acciones, proporcionadas por ovs-vswitchd. Estas acciones indican los puertos físicos o túneles a los cuales el paquete debe ser transmitido. En caso de que la ruta de datos no tenga instrucciones específicas para el paquete, éste es entregado a ovs-vswitchd.En el espacio de usuario, ovs-vswitchd determina cómo debe ser gestionado el paquete y luego lo devuelve a la ruta de datos con el procesamiento deseado.

En OvS, la ruta de datos tiene dos capas de almacenamiento en caché: una de microflujo y una capa secundaria, llamada megaflujo, que almacena en caché las decisiones de reenvío para agregados de tráfico más allá de las conexiones individuales [52]. OvS tiene la capacidad de funcionar por completo en el espacio de usuario sin la necesidad de un

módulo de kernel,esta opción facilita la implementación de un conmutador basado en un kernel específico [50].

En OvS, se presentan dos interfaces claramente definidas: OpenFlow, que se utiliza para controlar el comportamiento de reenvío del conmutador, y OVSDB, que se encarga de la configuración del mismo. OVSDB es una base de datos que almacena la información de configuración del conmutador y es manipulada a través del protocolo de administración de OVSDB. Duranteel inicio, el daemon ovs-vswitchd recupera la información almacenada en OVSDB y la utilizapara establecer la configuración del conmutador y las rutas de datos correspondientes [1].

3.5. Raspberry Pi

La tarjeta Raspberry Pi 3 es capaz de realizar una amplia gama de tareas, como trabajarcon hojas de cálculo, procesar texto, navegar por Internet, programar y jugar, entre otras. Estedispositivo utiliza sistemas operativos basados en el kernel de Linux y se inicia y ejecuta desde una tarjeta SD, ya que no cuenta con memoria interna aparte de la ROM. La ranura para tarjeta SD puede admitir capacidades de hasta 32 GB. Los pines GPIO (General Purpose Input/Output) de la Raspberry Pi 3 Model B se pueden programar utilizando el lenguaje de programación Python, lo que permite asignar dispositivos de entrada/salida, como sensores,a los pines GPIO según sea necesario [53]. La Raspberry Pi 3 Model B está disponible en tresformatos distintos.

- Modelo B: estas son las placas de tamaño completo que incluyen puertos Ethernet yUSB.
- Modelo A: son consideradas como una versión más liviana de Raspberry Pi. Generalmente tienen especificaciones más bajas que el Modelo B principal, con menos puertosUSB y sin Ethernet, sin embargo, a un precio más bajo.
- Zero: es el Raspberry Pi más pequeño disponible. Está diseñado para aplicaciones integradas, dispositivos portátiles y creación de prototipos. Esta tarjeta tiene menos capacidades informáticas que el Modelo B, pero también usan mucha menos energía y no tienen puerto USB ni Ethernet.

3.5.1. Principales modelos y características

La Tabla 3.2 presenta una comparación de los modelos previamente descritos, detallando eltipo de plataforma, procesador, memoria RAM y puertos disponibles en cada tarjeta. Proporciona información clave sobre las características técnicas, como el tipo de plataforma en la que operan, el procesador utilizado, la capacidad de memoria RAM y los puertos disponiblesen cada modelo.

3.5.2. Sistema operativo y kernel

En la actualidad, Raspberry Pi cuenta con varios sistemas operativos disponibles, como RISCOS, Pidora, Arch Linux y Raspbian. Sin embargo, Raspbian es el sistema operativo basado en Linux más popular para Raspberry Pi. Raspbian es una distribución de código abierto basada en Debian, que ha sido adaptada específicamente para su uso en Raspberry Pi, de ahísu nombre. Raspbian incluye modificaciones y personalizaciones diseñadas para hacer que Raspberry Pi sea más accesible y fácil de utilizar, y viene con una amplia selección de paquetes de software listos para su uso. Además de ser un sistema operativo completo, Raspbian proporciona más de 35,000 paquetes de software precompilados en un formato convenientepara facilitar su instalación en Raspberry Pi. Este sistema operativo se basa en una combinación de diversos componentes clave que forman una distribución moderna de Linux, los cuales trabajan en conjunto para brindar todas las características y funcionalidades que se esperande una computadora [54]. Estos componentes son:

- El gestor de arranque de Raspberry Pi
- El Kernel Linux
- Daemons
- La Shell
- Utilidades de shell

3.5.2.1. El gestor de arranque de raspberry pi

La función del gestor de arranque en Raspberry Pi es la de inicializar el hardware a un estado conocido y cargar el kernel de Linux. Para ello, se utilizan los cargadores de arranque de primera y segunda etapa. El cargador de arranque de primera etapa está programado en la ROM durante la fabricación y no se puede modificar. Por otro lado, los

cargadores de arranque de la segunda y tercera etapa se encuentran almacenados en la tarjeta SD, y son ejecutados automáticamente por el cargador de arranque de la etapa anterior. De esta manera, se asegura un proceso de arranque adecuado en Raspberry Pi [54].

3.5.2.2. El kernel Linux

El kernel de Linux desempeña un papel fundamental en Raspbian, ya que se encarga de gestionar todas las funciones de Raspberry Pi. Cuando se conecta un dispositivo de hardware ala Raspberry Pi, el kernel debe reconocerlo y saber cómo utilizarlo. Una vez que el kernel hafinalizado su carga, se ejecuta automáticamente un programa llamado “init”. Este programa tiene la tarea de completar la inicialización de la Raspberry Pi y luego cargar el resto del sistema operativo. Durante este proceso, se cargan los diferentes servicios en segundo plano yfinalmente se muestra la interfaz gráfica de usuario para su interacción [54].

3.5.2.3. Demonios

Un demonio es una pieza de software que se ejecuta en segundo plano para proporcionar diferentes características al sistema operativo [54].

3.5.2.4. Shell

Un shell es una interfaz de línea de comandos que se utiliza en la Raspberry Pi para monitorear y controlar el sistema. Raspbian, el sistema operativo utilizado en la Raspberry Pi, utiliza el shell Bourne Again (bash), que es común en Linux. Bash ofrece numerosas funcionalidades y permite ejecutar comandos, administrar archivos y directorios, y personalizar el sistema [54].

3.5.2.5. Utilidades de la shell

Un intérprete de comandos depende de los comandos para ser útil. Mientras que bash incluyecomandos básicos, las utilidades de shell constituyen la mayoría de los comandos disponibles. Estas utilidades forman una parte vital de Raspbian, ya que sin ellas el sistema

no podríafuncionar correctamente. Ofrecen una amplia gama de funciones, como copiar archivos, crear directorios y utilizar herramientas avanzadas como APT (Advanced Packaging Tool), una aplicación de gestión de paquetes que permite la instalación y desinstalación de software en la Raspberry Pi [54].

Tabla 3.2: Especificaciones técnicas modelos Raspberry Pi. Extraído de [5].

Raspberry Pi Platform	CPU	RAM	I/O Ports
Raspberry Pi 400	1.8 Hz, Quad-core Broadcom BCM2711 (Cortex-A72)	4GB	2 × USB 3.0, 1 x USB 2.0 ports, 2 x micro HDMI, 1 x Gigabit Ethernet
Raspberry Pi 4B	1.8 Hz, Quad-core Broadcom BCM2711 (Cortex-A72)	8GB	2x USB 3.0, 2x USB 2.0, 1x Gigabit Ethernet, 2x micro HDMI
Raspberry Pi 4B	1.8 Hz, Quad-core Broadcom BCM2711 (Cortex-A72)	4GB	2x USB 3.0, 2x USB 2.0, 1x Gigabit Ethernet, 2x micro HDMI
Raspberry Pi 4B	1.8 Hz, Quad-core Broadcom BCM2711 (Cortex-A72)	2GB	2x USB 3.0, 2x USB 2.0, 1x Gigabit Ethernet, 2x micro HDMI
Raspberry Pi 3B+	1.4-GHz, 4-core Broadcom BCM2837B0 (Cortex-A53)	1GB	4 x USB 2.0, HDMI, 3.5mm audio
Raspberry Pi Zero WH	1-GHz, 1-core Broadcom BCM2835 (ARM1176JZF-S)	512MB	1x micro USB, 1x mini HDMI
Raspberry Pi Zero W	1-GHz, 1-core Broadcom BCM2835 (ARM1176JZF-S)	512MB	1x micro USB, 1x mini HDMI

4. Metodología

En este capítulo se describe, a nivel técnico, las herramientas empleadas en el entorno de pruebas que se utilizan para el desarrollo y configuración de diversas topologías basadas en VLANs. Además, se aborda el proceso de descarga, instalación y configuración de varias herramientas de software utilizadas en este trabajo de titulación, junto con la explicación de lascaracterísticas del diseño de la estructura para el hardware.

4.1. Descripción entorno de pruebas

En ingeniería, realizar pruebas en un entorno controlado es esencial para cualquier proyecto, ya que garantiza la verificación y correcto funcionamiento del mismo. Estas pruebas son fundamentales para identificar y solucionar errores y problemas en etapas tempranas, lo queayuda a reducir costos y tiempos de resolución durante la producción. Además, las pruebas en este entorno brindan información valiosa que permite mejorar y optimizar proyectos futuros.

En el presente trabajo de titulación, se desarrolla un entorno de pruebas dividido en tres etapas, cada una de las cuales configura e implementa VLANs marcando la transición desde unared clásica hasta una SDN. En primer lugar, se diseña e implementa una topología de red clásica con asignación de VLANs utilizando tarjetas RPi, como se muestra en la Figura 4.1. Posteriormente, se lleva a cabo la implementación de una topología SDN utilizando el protocolo OpenFlow en tarjetas RPi, con el objetivo de simular una red CAN dividida en diferentesdepartamentos y tipos de equipos, como se muestra en la Figura 4.2. Finalmente, se concluye con la implementación de un entorno de pruebas que consta de al menos cuatro equipos (tarjetas RPi) compatibles con el protocolo OpenFlow, formando así una SDN a la cual se conectan varios *hosts* pertenecientes a diferentes VLANs. Además, se incorpora un servidor NAC encargado de ejercer el control de acceso a la red, brindando visibilidad de los dispositivos yusuarios que intentan acceder a la red de campus. Este entorno de pruebas incluye múltiples*hosts* que se conectan a la SDN para ser asignados a su grupo específico de VLAN, como seaprecia en la Figura 4.3.

4.2. Hardware utilizado para la implementación de la topología de red

El entorno de pruebas planificado está orientado a una implementación en una CAN real, porlo tanto, se han considerado los componentes de hardware necesarios para cumplir con este

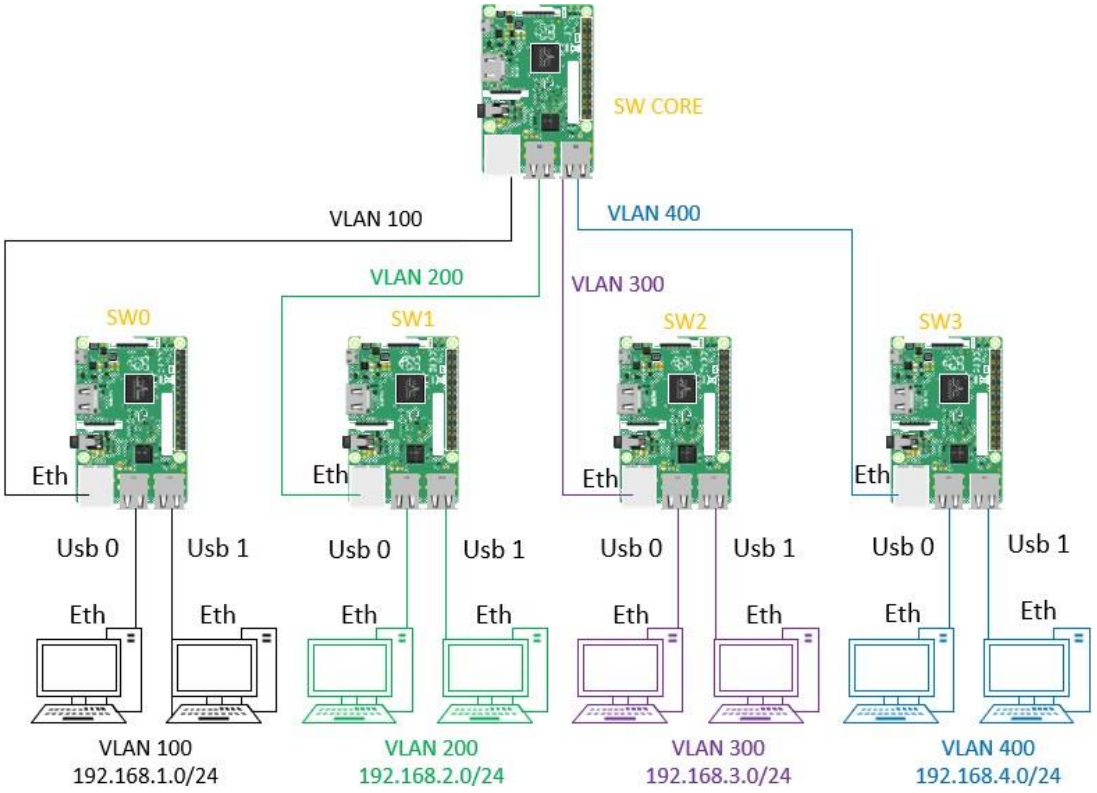


Figura 4.1: Entorno de pruebas para asignación de VLANs en una red clásica.

propósito. Se ha tenido en cuenta la minimización de la inversión monetaria, con el objetivo de que sea asequible para cualquier persona. Por esta razón, se han priorizado el uso de tarjetas RPi, computadoras, cables Ethernet y adaptadores USB-Ethernet como los principales componentes requeridos.

Las tarjetas RPi en conjunto con OvS ofrecen capacidades técnicas que les permiten actuarcomo conmutadores y son compatibles con el protocolo OpenFlow, lo que facilita su integración tanto en el diseño de la red clásica como en la SDN. Dado que estas tarjetas funcionan comoconmutadores, se hace necesario utilizar adaptadores USB-Ethernet para aumentar el número de interfaces de red y poder conectar múltiples hosts o conmutadores. Los ordenadores se utilizan para verificar el correcto funcionamiento de cada una de las topologías mencionadasanteriormente (Figuras 4.1, 4.2 y 4.3), ya que a través de ellos

se llevan a cabo las pruebas de comunicación entre los *hosts* asignados a la misma VLAN.

En la Tabla 4.1 se muestra un resumen del *hardware* necesario para llevar a cabo la implementación del presente trabajo de titulación.

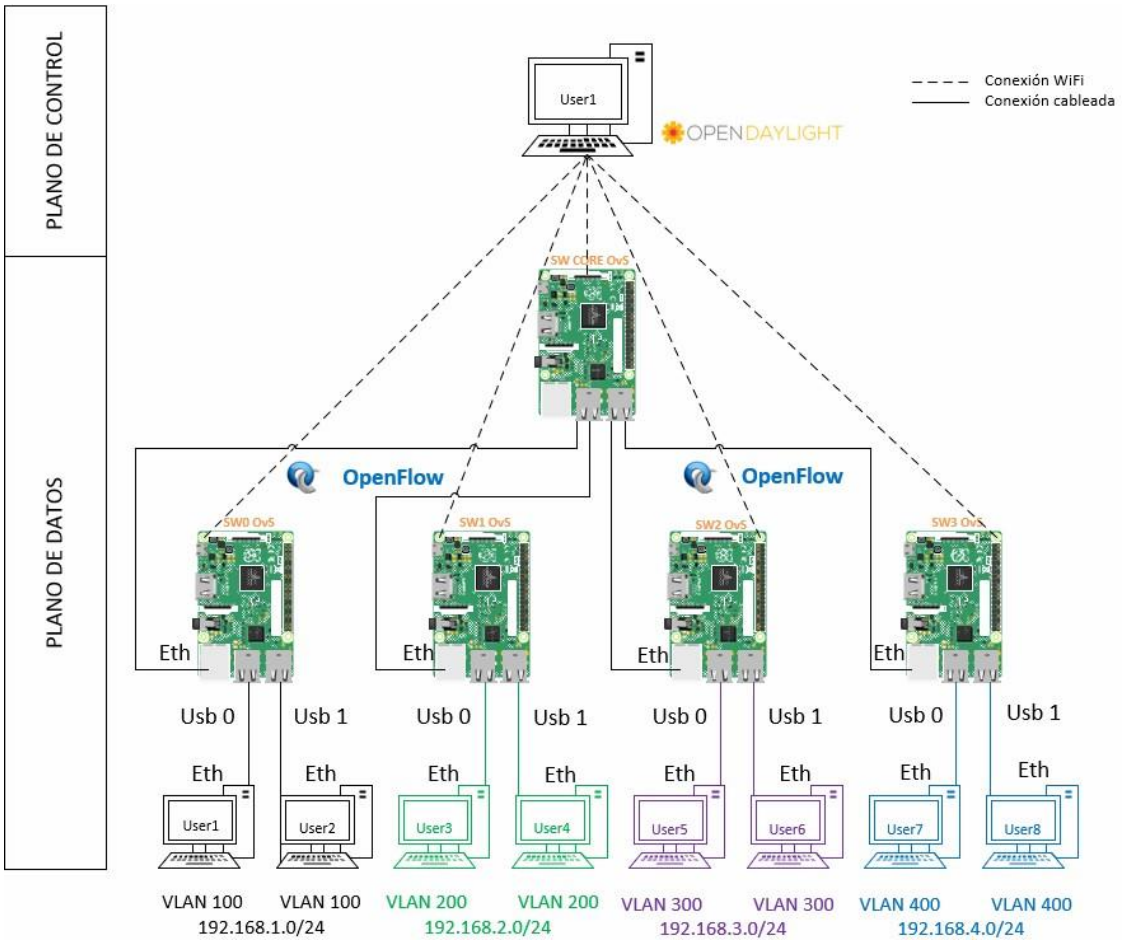


Figura 4.2: Entorno de pruebas para asignación estática de VLANs en una SDN

4.3. Diseño de estructura para el montaje del entorno de pruebas

Uno de los objetivos principales de este trabajo de titulación es implementar un entorno de pruebas a nivel de *hardware* que facilite el uso de controladores SDN y proporcione la flexibilidad necesaria en los entornos de pruebas. Con este propósito en mente, se ha diseñado unrack de piso que permite agrupar las tarjetas RPi y los adaptadores de red USB-RJ45, creando así un banco de pruebas ordenado y eficiente para su utilización.

La metodología seguida para el proceso de diseño del rack a escala se detalla en el ApéndiceA.

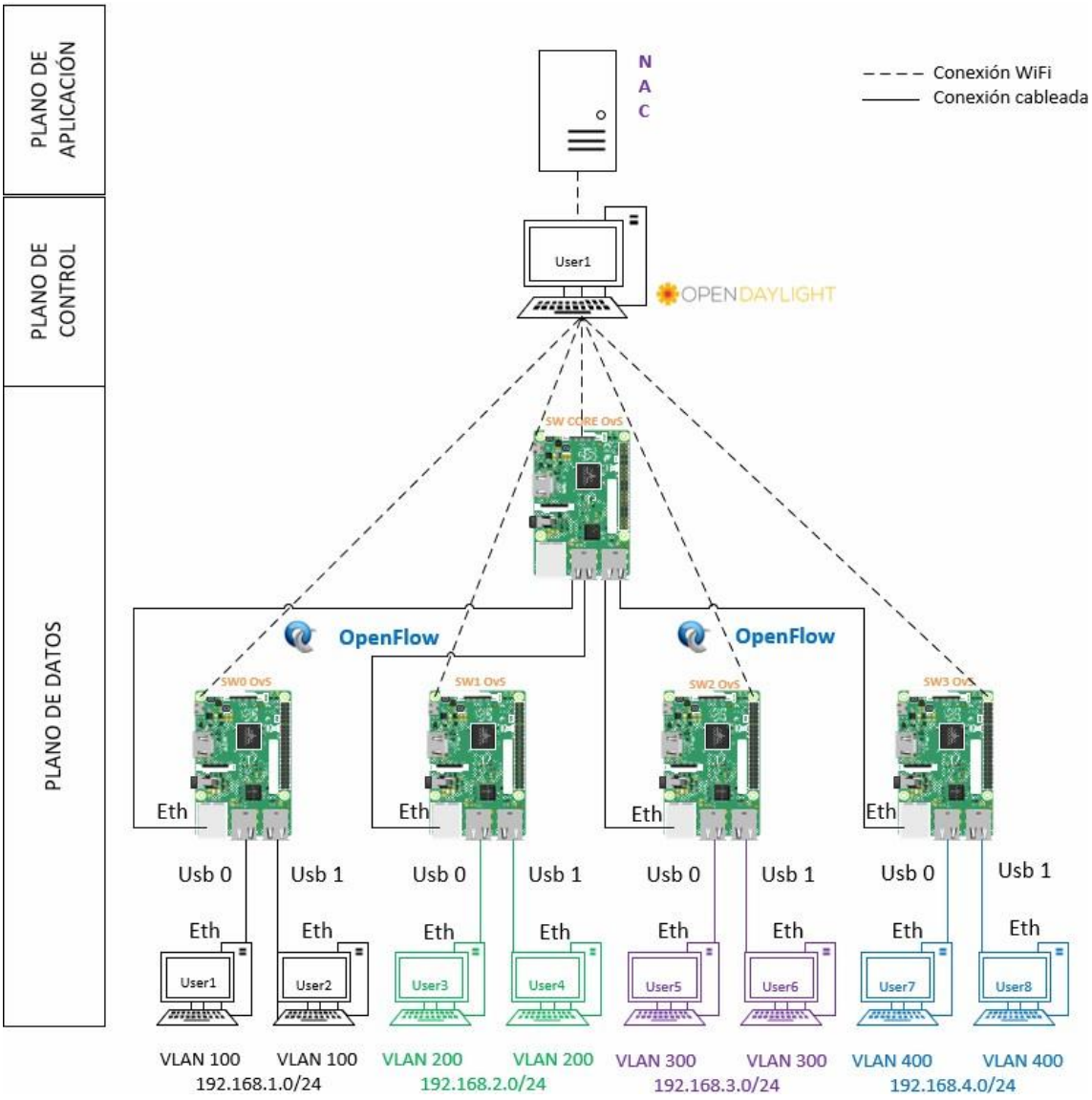


Figura 4.3: Entorno de pruebas para asignación dinámica de VLANs en una SDN.

4.4. Presupuesto

La inversión monetaria realizada en la implementación física se ha estimado en una cantidad de $ 589.20, la cual ha sido destinada a la adquisición de hardware y materiales necesarios para llevar a cabo dicha implementación. Los precios de los componentes adquiridos corresponden a los precios actuales del mercado. En la Tabla 4.2 se detallan los componentes adquiridos.

El valor económico de aproximadamente **USD 590** es cubierto por los autores de este Trabajode Titulación.

Tabla 4.1: Hardware adquirido para la implementación física

Cant	Dispositivo	Función
1	PC Ubuntu 22 3B	Controlador SDN
1	Raspberry Pi 3B	Conmutador OpenFlow Principal
4	Raspberry Pi 3B	Conmutador OpenFlow Acceso
8	Ordenadores	Clientes
12	Cables Eth	Conexión entre conmutadores Conexión controlador-conmutador
12	Adaptador USB-Eth	Incrementar interfaces Ethernet tarjetas Raspberry Pi

4.5. Configuración tarjeta raspberry pi para implementación del plano de datos

La configuración que se realiza sobre las tarjetas Raspberry Pi se divide en tres etapas, lascuales se listan a continuación.

- Implementación servidor Dynamic Host Configuration Protocol (DHCP).
- Instalación OvS.
- Configuración de VLAN sobre Raspberry Pi (RPi) empleando OvS.

4.5.1. Diagrama para la implementación del plano de datos

En la Figura 4.4 se presenta el diagrama de bloques que resume el proceso a seguir para laimplementación del plano de datos sobre una tarjeta RPi. Donde, en primer lugar es necesario contar con un servidor de Protocolo de Configuración Dinámica de Host (DHCP), que es ampliamente utilizado en redes debido a su función de facilitar el acceso a la red. Los servidores DHCP actúan como agentes de los administradores de red y automatizan el proceso deasignación de direcciones y configuración de parámetros en dispositivos de red que utilizan TCP/IP [55, 56]. Para implementar un servidor DHCP en la tarjeta RPi se requiere seguir loscomandos indicados en el Apéndice B.1.

Posteriormente, se requiere de un *software* que permita a la tarjeta RPi funcionar como

Tabla 4.2: Presupuesto para la implementación Física

Cantidad	Componente	P. Unit. $	Total $
5	Raspberry Pi Model 3B+	80.00	400.00
12	Adaptador USB-Ethernet	8.00	96.00
12	Cable Ethernet RJ45	1.10	13.2
1	Estructura impresa en 3D	70.00	70.00
1	Regleta eléctrica	10.00	10.00
		Total $	**589.20**

conmutador de red. Para esto es necesaria la instalación de OvS. Los comandos requeridos parallevar a cabo esta instalación se describen en el Apéndice B.2. Finalmente, se procede con laconfiguración de VLAN sobre los conmutadores que

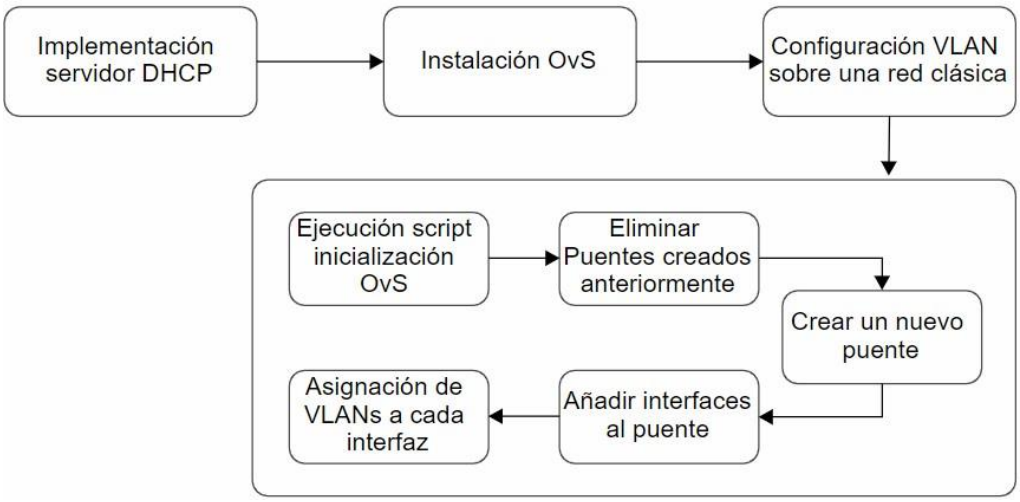


Figura 4.4: Diagrama de bloques implementación plano de datos.

forman la red clásica.

4.5.2. Configuración de VLAN sobre raspberry pi empleando Open vSwitch en una redclásica

Como se mencionó anteriormente, el presente trabajo de titulación se divide en tres etapas. En esta subsección se describe la primera de estas, que tiene como objetivo la implementaciónde VLAN en una red clásica. Para lograr esto, se propone la topología mostrada en la Figura 4.1. Donde, se utiliza el script del Extracto de código B.11 para

iniciar el servicio OvS, y se agregan los siguientes comandos:

```
1    ovs-vsctl          --if-exists          del-br
2    Nombre_Puente_Previo     ovs-vsctl       add-br
3    Nombre_Puente
4    ovs-vsctl     add-port     Nombre_Puente     Interfaz_1
5    tag=VLAN_tag    ovs-vsctl     add-port     Nombre_Puente
6    Interfaz_2      tag=VLAN_tag     ovs-vsctl      add-port
     Nombre_Puente Interfaz_3 tag=VLAN_tag
     ovs-vsctl show
```

Extracto de código 4.1: Comando OvS para la generación de VLANs.

Los comandos añadidos al script del Extracto de código B.11, junto con los que se muestranen el Extracto de código 4.2, tienen los siguientes objetivos: eliminar cualquier configuraciónprevia de un `bridge` con el mismo nombre, crear un nuevo `bridge` para conectar las distintas interfaces de la tarjeta RPi, añadir las interfaces al `bridge` recién creado y asignar las VLANs acada interfaz. A continuación, se muestra el script necesario para asignar VLANs a dos puertos del conmutador OvS con las configuraciones necesarias para el funcionamiento adecuado.

```
1  ovsdb-server    --remote=punix:/usr/local/var/run/openvswitch/db.sock\
                --remote=db:Open_vSwitch,Open_vSwitch,manager_options
2                  --private-key=db:Open_vSwitch,SSL,private_key \
                   --certificate=db:Open_vSwitch,SSL,certificate \
3                  --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
4                  --pidfile     --
5    detach ovs-vswitchd  --pidfile
6    --detach ovs-vsctl  --no-wait
7    init
8    ovs-vsctl       --if-exists       del-br
9    Prev_Brigde1ovs-vsctl  --if-exists  del-
10   br    Prev_Brigde2   ovs-vsctl    add-br
11   Nombre_Puente
12   ovs-vsctl     add-port     Nombre_Puente     Interface1
13   tag=VLAN_tag    ovs-vsctl     add-port     Nombre_Puente
14   Interface2 tag=VLAN_tag
```

Extracto de código 4.2: Script para iniciar OvS junto con VLANs.

4.6. Configuración de sistema operativo Ubuntu 22 para implementación del plano decontrol

Dentro del ámbito de las redes definidas por software (SDN), existen numerosos controladores con distintas características y capacidades que los hacen destacar. Algunos de estos controladores han sido abordados en secciones anteriores. Sin embargo, para este trabajo de titulación se ha seleccionado el controlador OpenDayLight (ODL) debido a sus destacadascaracterísticas, mencionadas en el capítulo 3.

4.6.1. Obtención controlador OpenDaylight

En esta sección se describe el proceso de obtención del controlador ODL. El primer paso es seleccionar la versión de ODL a instalar. Para este trabajo, se ha elegido la versión `0.4.4-Bery- llium-SR4`, disponible en el repositorio de ODL. En la Figura 4.6 se presenta el diagrama debloques que resume el proceso a seguir para la obtención del controlador ODL en el SistemaOperativo Ubuntu 22.

El primer paso consiste en seleccionar la versión deseada del controlador, en este caso, la versión `0.4.4-Beryllium-SR4`, disponible en el repositorio de ODL. A continuación, se des- carga el controlador desde el repositorio de *The Linux Foundation*. Para asegurar el correctofuncionamiento de esta versión, es necesario tener instalado Java Development Kit (JDK) ensu versión 8.

Después de la descarga, se crea una variable de entorno para definir la versión principal de Java en el sistema operativo. Luego, se ejecuta el script `karaf` para iniciar el controlador ODL. Una vez que el controlador está en funcionamiento, se procede a instalar varias características que garantizan su correcto rendimiento y permiten el reconocimiento y visualizaciónde los dispositivos de red a través de la interfaz web. Finalmente, la conexión al controlador mediante la interfaz web se realiza introduciendo la siguiente dirección en un navegador.`http://Direccion_IP:8181/index.html`, tal como se observa en la Figura 4.5.

Los comandos requeridos para la obtención del controlador ODL se describen en el ApéndiceB.3.

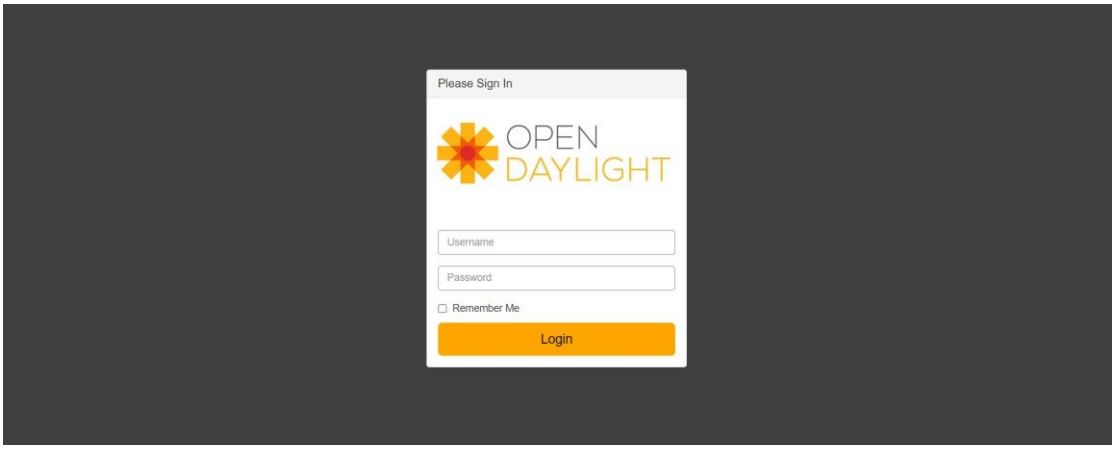


Figura 4.5: Página de inicio controlador ODL.

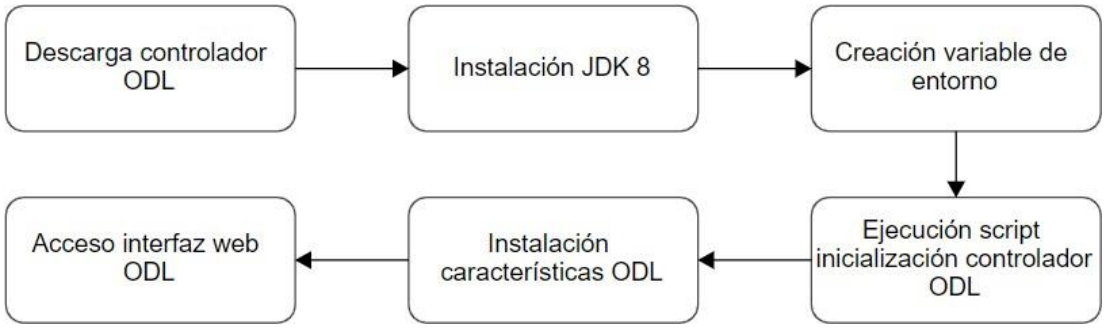


Figura 4.6: Diagrama de bloques obtención controlador ODL.

4.6.2. Conexión Open vSwitch - OpenDaylight

En esta subsección se proporciona información detallada sobre la integración de dos de las herramientas previamente instaladas, las cuales son el punto de partida para construir la topología SDN deseada (Figura 4.2). A continuación, se describe el procedimiento para establecerla conexión entre OvS y ODL.

En primera instancia, se inicia el servicio de OvS ejecutando el script B.11 y se crea un nuevopuente (bridge) para facilitar la conexión. Esto se logra mediante la implementación de los comandos del Extracto de código 4.1. El siguiente paso consiste en establecer la conexión entre OvS y el controlador ODL utilizando el protocolo de transporte TCP a través del puerto6633, que se utiliza para el protocolo OpenFlow. Para lograr esto, se introduce el siguiente comando en la consola:

```
1    sudo ovs-vsctl set-controller Nombre_Puente
     tcp:IP_controlador:6633
```

Extracto de código 4.3: Comando OvS para conexión con el controlador ODL.

Una vez establecida la conexión entre OvS y ODL, es fundamental definir el protocolo de comunicación a utilizar. En este caso, se utilizará el protocolo OpenFlow versión 1.3, para locual se ejecutará el siguiente comando:

```
1    sudo ovs-vsctl set-bridge Nombre_Puente protocols=OpenFlow13
```

Extracto de código 4.4: Comando OvS para establecer el protocolo OpenFlow.

Al ejecutar los dos comandos previos, se establece la conexión entre OvS y ODL, lo que permite crear una SDN. El siguiente paso es detectar los distintos hosts a través de ODL. Para ello,se requiere conectar físicamente un host a la tarjeta RPi y agregar la interfaz correspondiente al bridge que se creó anteriormente. Este objetivo se logra con el siguiente comando:

```
1    sudo ovs-vsctl add-port Nombre_Puente Nombre_Interfaz
```

Extracto de código 4.5: Comando OvS para agregar un puerto al puente.

Al seguir el procedimiento detallado anteriormente, se establece la conexión entre el controlador ODL y OvS, como se muestra en la Figura 4.8. En la figura, se puede observar que el controlador ha detectado el conmutador OvS a través del protocolo OpenFlow. Esta conexión proporciona la base para organizar las tarjetas RPi y construir la topología deseada de la reddefinida por software (SDN), que se ha descrito en secciones anteriores (Figuras 4.2 y 4.3).

En la Figura 4.7 se presenta el diagrama de bloques que resume el proceso a seguir para laconexión entre OvS y el controlador ODL.

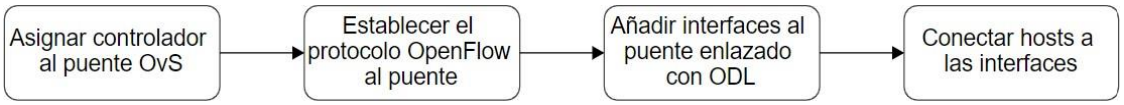


Figura 4.7: Diagrama de bloques conexión Open vSwitch - OpenDaylight.

4.6.3. Conexión OpenDaylight - Open Flow Manager

En esta sección se describe la conexión entre ODL y una herramienta adicional llamada Open-

Flow Manager (OFM). OFM es una aplicación de código abierto desarrollada por Cisco, diseñada para ejecutarse sobre ODL y ofrecer funcionalidades como la visualización de topologías

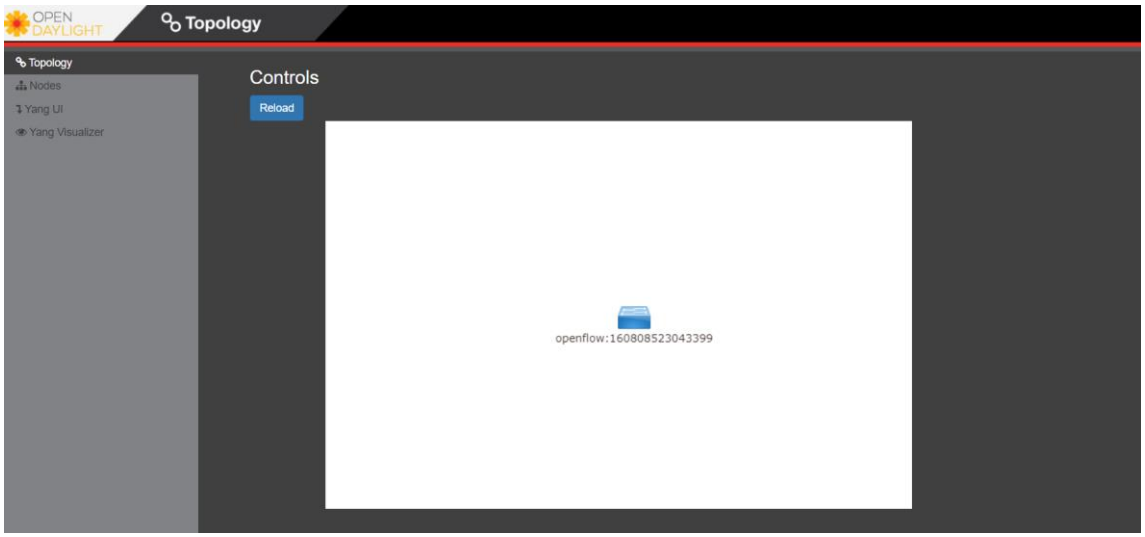


Figura 4.8: Conexión controlador ODL y conmutador OvS.

OpenFlow, programación de rutas OpenFlow y recopilación de estadísticas OpenFlow. También proporciona una interfaz web intuitiva y amigable para los administradores de red, con diversas opciones para la gestión de SDN. Por estas razones, se ha decidido integrar esta aplicación SDN en el proyecto.

El proceso para configurar OFM implica la instalación de algunas herramientas específicas en el sistema operativo Linux. Para comenzar, se instala `npm`, una herramienta que proporcionaun entorno de ejecución para JavaScript, un lenguaje esencial para el proyecto OFM. Además, es necesario instalar `curl`, una herramienta enfocada en la transferencia de datos. También,es necesario contar con la herramienta Node.js, que proporciona el entorno de ejecución para los scripts de OFM escritos en JavaScript. Después de instalar las dependencias requeridaspara ejecutar OFM, se procede a obtener el software desde su repositorio correspondiente.

Una vez clonado el proyecto y accediendo al directorio descargado, se realiza una modificación en uno de los scripts generados, ubicado en la ruta `/ofm/src/common/config/`. En concreto, se necesita reemplazar el valor del campo correspondiente a la dirección IP a la cualOFM apuntará. Esta modificación se realiza en la línea 7 del Extracto de código B.21. Para finalizar la configuración de OFM, se procede a instalar el cliente de la herramienta `Grunt`. Grunt es una herramienta que automatiza

diversas tareas relacionadas con JavaScript.

Finalmente, para administrar la SDN utilizando OpenFlow Manager (OFM), basta con accedera la dirección `https://Direccion_IP:9000` en un navegador web (Figura 4.9). En la Figura 4.10 se presenta el diagrama de bloques que resume el proceso a seguir para la conexión entre el controlador ODL y la API OFM.

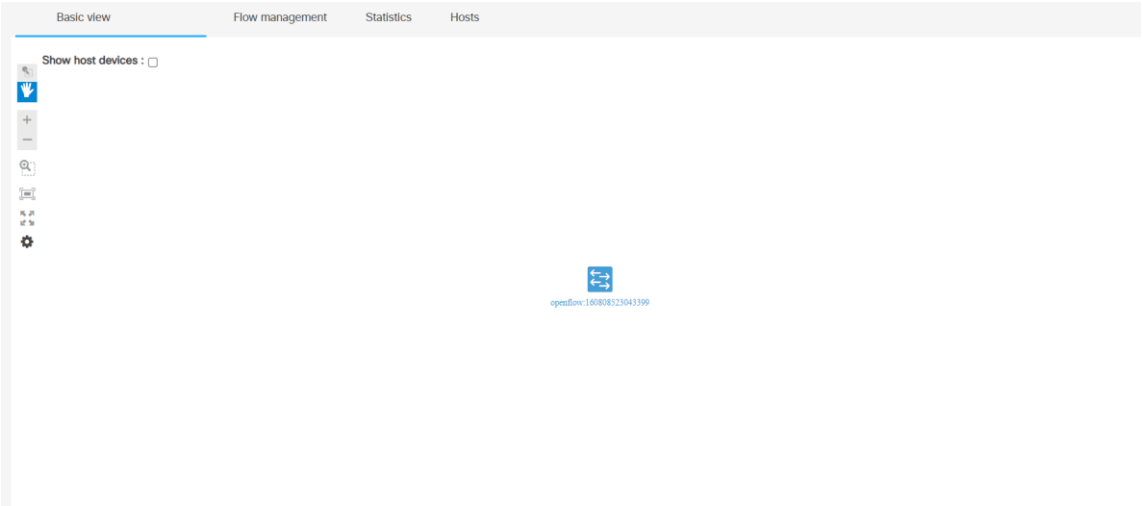


Figura 4.9: Conexión controlador ODL y OFM.

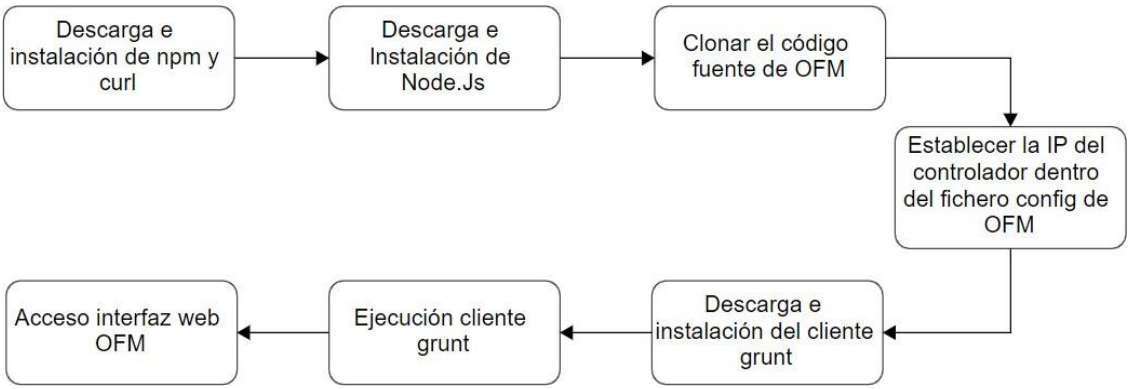


Figura 4.10: Diagrama de bloques conexión OpenDaylight - Open Flow Manager.

4.6.4. Configuración de VLAN sobre raspberry pi empleando Open vSwitch en una SDN

Las redes definidas por software (SDN) han ganado popularidad como el nuevo paradigma adoptado por empresas, instituciones educativas, corporaciones y otros, debido a su capacidad para lograr un control centralizado de todos los dispositivos de red.

Dentro de esta categoría se encuentran las redes de área campus (CAN), las cuales pueden variar en tamaño desde medianas hasta muy grandes, dependiendo de factores como la cantidad de estudiantes, docentes y personal administrativo, entre otros. Por lo tanto, es crucial que los administradores de red puedan gestionar los dispositivos desde una ubicación centralizada para agilizar tareasy simplificar las configuraciones individuales. Antes de abordar el objetivo principal, es necesario continuar con la transición de asignación de VLANs, y en este sentido, se presentarán los beneficios y dificultades de realizar la asignación estática de VLANs en una SDN.

En esta sección se exploran dos enfoques para la configuración y asignación de VLANs en una red SDN utilizando el controlador ODL. La primera opción consiste en programar el controlador utilizando las capacidades de la característica `odl-mdsal`, la cual ofrece una abstracción dedatos y permite enviar acciones a los conmutadores OpenFlow (tarjetas RPi con OvS) para controlar el flujo de datos. La segunda opción implica utilizar el administrador OpenFlow OFMde Cisco, el cual proporciona una interfaz web que permite al administrador de red seleccionar y ejecutar acciones a través de una interacción intuitiva con dicha interfaz.

4.6.4.1. Asignación estática de VLAN con controlador OpenDaylight (programación)

El controlador ODL cuenta con una amplia gama de plugins y funciones que facilitan la implementación de una SDN. Uno de estos plugins es `OVSDB MD-SAL Southbound Plugin`, elcual amplía las capacidades del controlador al permitir la configuración de OvS a través de lainterfaz Sur dentro del contexto de la SDN. Al utilizar este plugin en conjunto con `mdsal`, es posible programar la SDN mediante el protocolo RPC utilizando archivos JSON o XML, lo que brinda un control total sobre la topología a través de las reglas programadas en el controlador.

Es crucial aprovechar los beneficios significativos que proporciona la implementación de un controlador en una CAN basada en la tecnología SDN. Con el objetivo de maximizar dichas ventajas, se procede a programar el controlador utilizando la topología ilustrada en la Figura4.2, con el fin de asignar VLANs a distintos grupos de hosts que pertenecen a departamentoso grupos de estudiantes dentro de la CAN. De esta manera, se logra una gestión eficiente y segmentada de la red, optimizando el rendimiento y la seguridad en el entorno.

Después de haber construido físicamente la topología, el siguiente paso consiste en identificar el conmutador que se va a configurar a través de la topología reconocida por el controlador ODL. Como se mencionó anteriormente en la sección 4.6.1, es posible acceder a la interfaz web utilizando la siguiente dirección: `https://Direccion_IP:8181/index.html` en la sección de *Topology*. La Figura 4.11 muestra la topología identificada por el controlador, la cual coincide con la representación mostrada en la Figura 4.2. De esta manera, se puede visualizar y verificar la configuración de los conmutadores y los hosts en la red.

La Figura 4.11 muestra la topología de red en la interfaz de ODL, que permite identificar los conmutadores OvS y los hosts conectados a cada uno de ellos. En la sección *Nodes*, se pueden visualizar las interfaces y los puertos asociados a cada conmutador lo cual es relevante paracrear el archivo JSON que contendrá las reglas específicas para la

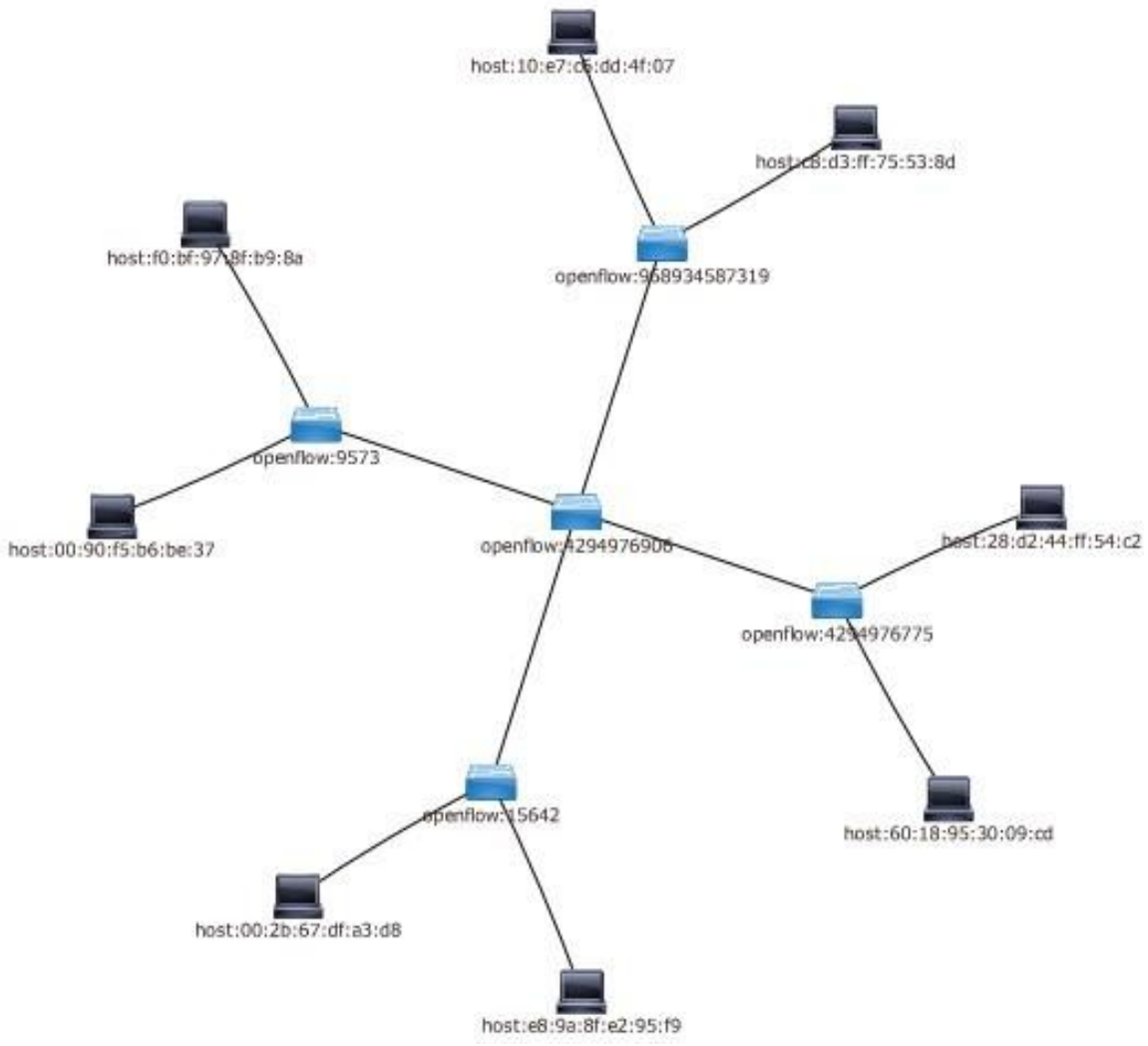


Figura 4.11: Topología para asignación estática de VLANs en una SDN vista desde ODL.

asignación de las VLANs.En la Figura 4.12, se puede apreciar el contenido de la pestaña *Nodes* de la interfaz web deODL, donde se identifican los conmutadores OpenFlow de la red y se muestra su respectivoidentificador. Asimismo, en la Figura 4.13 se pueden ver las interfaces del conmutador principal, proporcionando información detallada sobre su conectividad.

A continuación (Extracto de código 4.6), se muestra el código utilizado para realizar la asignación estática de VLANs mediante el controlador ODL. El código está en formato JSON y enél se puede observar claramente que se trata de un flujo agregado al conmutador con identificador (4294976775) a través del puerto 2, con el propósito de asignar y detectar paquetespertenecientes a la VLAN 400.

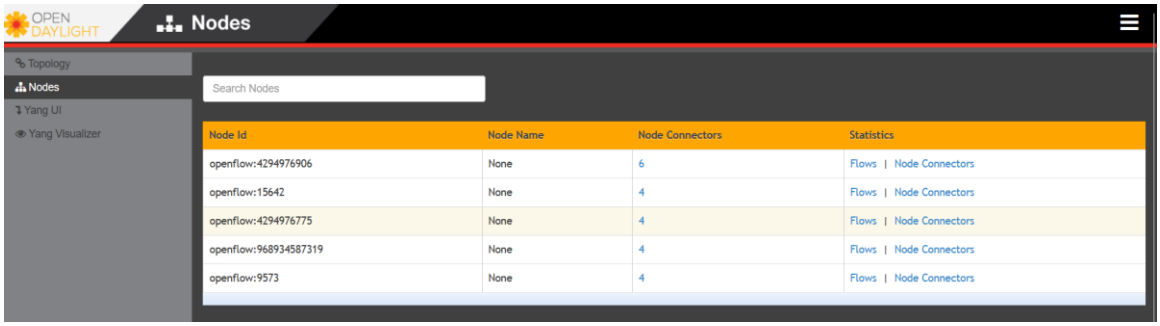


Figura 4.12: Nodos identificados en la SDN.

Node Connector Id	Name	Port Number	Mac Address
openflow:4294976906:1	eth0	1	B8:27:EB:7B:6F:0C
openflow:4294976906:3	eth2	3	00:01:00:00:26:38
openflow:4294976906:2	eth1	2	00:01:00:00:58:BC
openflow:4294976906:5	eth4	5	00:E2:99:00:2E:3D
openflow:4294976906:4	eth3	4	00:01:00:00:25:8A
openflow:4294976906:LOCAL	br0	LOCAL	00:01:00:00:25:8A

Figura 4.13: Interfaces del conmutador principal (429476906).

El primer paso consiste en establecer las características básicas para asignar el flujo, lo cualimplica crear la estructura de la tabla de flujo (Figura 3.3). Para ello, se asigna el ID de la tablacorrespondiente al conmutador a configurar en la base de datos de ODL. Luego, se estableceun identificador para reconocer el flujo, se define la prioridad y los contadores asociados. A continuación, se especifican los parámetros de emparejamiento, en este caso, los paquetes entrantes por el puerto 2 del conmutador 4294976775 deben pertenecer a la VLAN 400. Finalmente, se configuran las instrucciones de salida para los

paquetes, indicando que deben serasignados a la VLAN correspondiente a su grupo o departamento (400). Donde, se crea la estructura del encabezado de un paquete 8021.q, que incluye el tipo de enlace Ethernet (33024)con el tag apropiado, la prioridad (pcp), el indicador de formato canónico (cfi) y el identificadorde VLAN.

Para permitir la comunicación entre los hosts pertenecientes a la misma VLAN, es necesario configurar el puerto al que están conectados. La configuración del puerto restante del conmutador 4294976775 se encuentra detallada en el Extracto de código 4.7. Este proceso debe repetirse para cada uno de los puertos presentes en la topología mostrada en la Figura 4.11,los cuales se encuentran detallados en el Apéndice E.2.1.

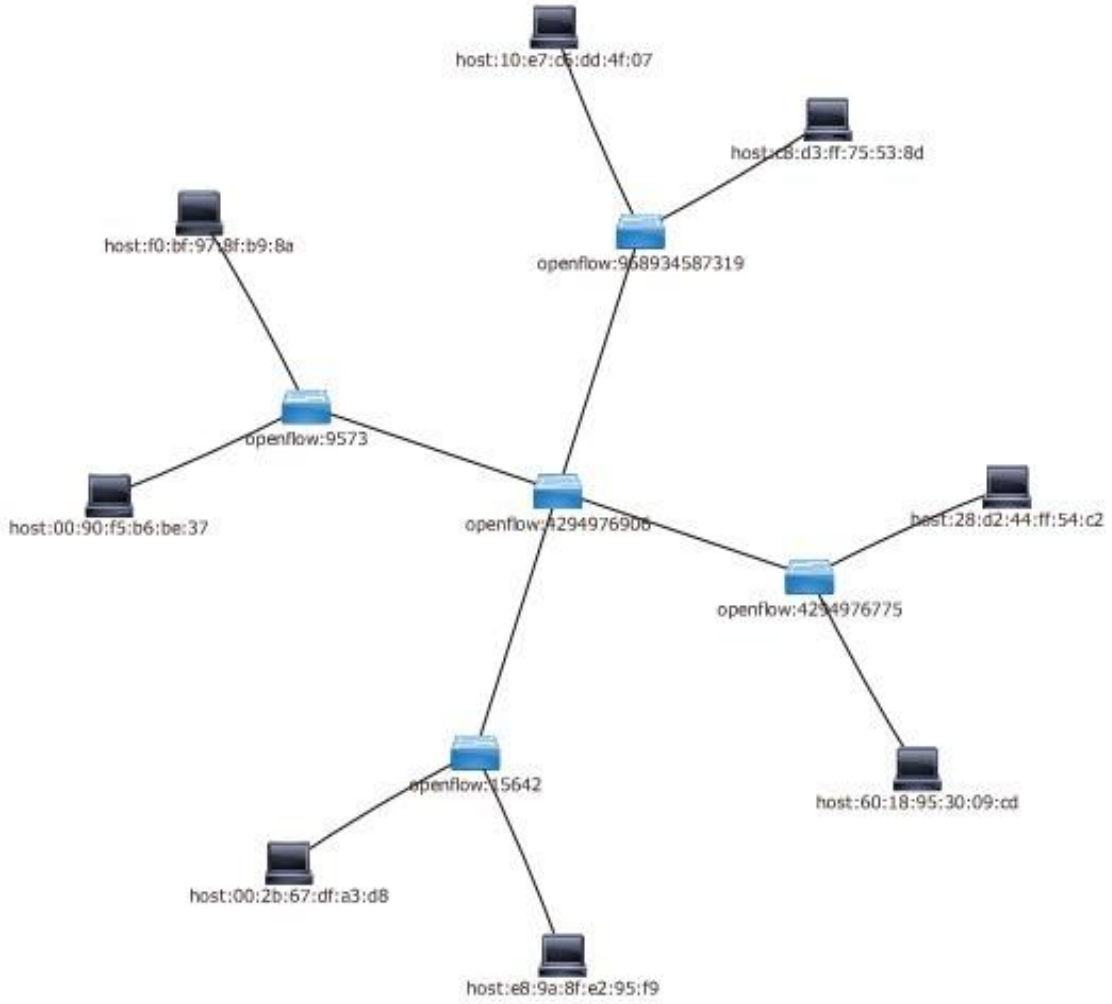


Figura 4.11: Topología para asignación estática de VLANs en una SDN vista desde ODL.

```
1
    {
        "flow":
            [
```

```
                "table_id": 0,
                "id": "89-400-e1",
                "priority": 1145,
                "hard-timeout": 0,
                "idle-timeout": 0,
                "match": {
                    "in-port": "openflow:4294976775:2",
                    "vlan-match": {
                        "vlan-id": {
                            "vlan-id": 400,
                            "vlan-id-present": "true"
                        }
                    }
                },
                "instructions": {
                    "instruction": [
                        {
                            "order": 0,
                            "apply-actions": {
                                "action": [
                                    {
                                        "order": 0,
                                        "push-vlan-action": {
                                            "ethernet-type": 33024,
                                            "tag": 400,
                                            "pcp": 7,
                                            "cfi": 0,
                                            "vlan-id": 400
                                        }
                                    }
                                ]
                            }
                        }
                    ]
                }
            }
```

```
        ]
    }
```

Extracto de código 4.6: Asignación estática de VLAN 400 (puerto 2) vía ODL.

```
{
    "flow": [
        {
            "table_id": 0,
            "id": "89-400-e1",
            "priority": 1145,
            "hard-timeout": 0,
            "idle-timeout": 0,
            "match": {
                "in-port": "openflow:4294976775:2",
                "vlan-match": {
                    "vlan-id": {
                        "vlan-id": 400,
                        "vlan-id-present": "true"
                    }
                }
            },
            "instructions": {
                "instruction": [
                    {
                        "order": 0,
                        "apply-actions": {
                            "action": [
                                {
                                    "order": 0,
                                    "push-vlan-action": {
                                        "ethernet-type": 0x24,
                                        "tag": 400,
                                        "pcp": 7,
                                        "cfi": 0,
                                        "vlan-id": 400
                                    }
```

```
33                                                  }
34                                              ]
35                                          }
36                                      }
37                                  ]
38                              }
39                          }
40                      ]
41                  }
```

Extracto de código 4.7: Asignación estática de VLAN 400 (puerto 1) vía ODL.

4.6.4.2. Asignación estática de VLAN con controlador OpenDaylight (Interfaz web)

La asignación estática de las VLANs a diferentes hosts también puede realizarse mediante la interfaz web de OFM. En una SDN en la dirección Norte, se establece una abstracción dela red mediante el uso de APIs REST estándar, lo que permite a las aplicaciones administrartoda la red. La Figura 4.14 ilustra la topología planificada en el diseño mostrado en la Figura4.2, donde se utilizan los datos del controlador para mapear la topología de los dispositivos OpenFlow de la red y los hosts conectados a cada uno de ellos.

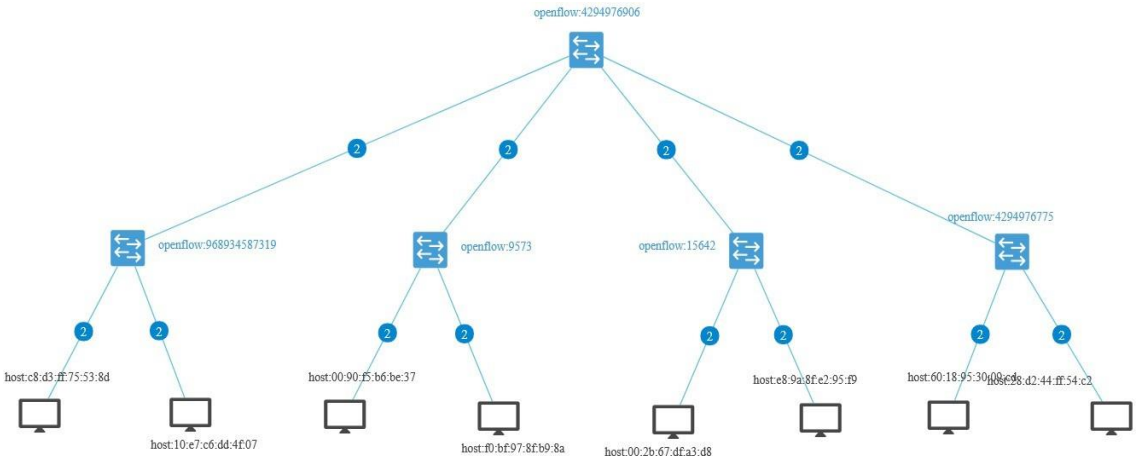


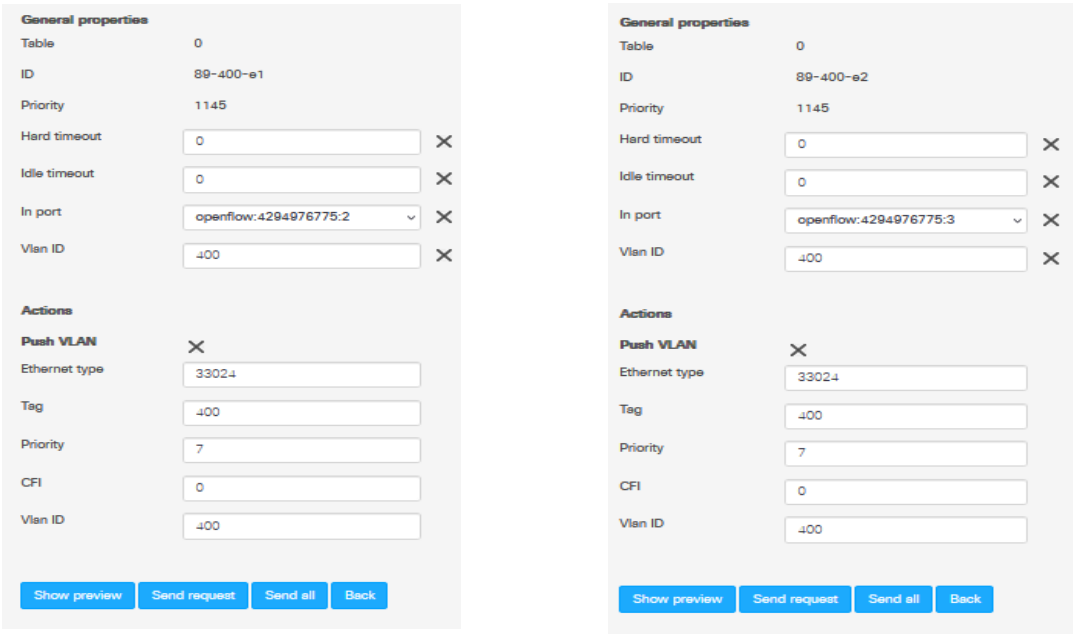
Figura 4.14: Topología para asignación estática de VLANs en una SDN vista desde OFM.

Para lograr el objetivo de asignar la VLAN a varios hosts, es necesario acceder a la pestaña *Flow Management*, la cual permite determinar la cantidad de flujos para cada

dispositivo OpenFlow en la SDN, así como listar, agregar, modificar y eliminar los flujos configurados. Una vez dentro de esta pestaña, es posible agregar un nuevo flujo y especificar el dispositivoa configurar, utilizando el identificador observado en la topología.

El proceso de asignación de VLAN mediante ODL es bastante intuitivo para el administrador de red como se ilustra en las Figuras 4.15(b) y 4.15(a). Una vez seleccionado el dispositivo (conmutador) a configurar es posible habilitar o deshabilitar la asignación de las propiedadesgenerales del flujo a agregar. Las propiedades generales son: Flow name, Table, ID, Hard timeout, Idle timeout, Cookie, Cookie mask, Priority; los cuales coinciden con lo configurado en los Extractos de códigos 4.6 y 4.7. A continuación, se habilitan los parámetros de coincidenciaque se requieran, en nuestro caso es necesario habilitar: **In port** y **Vlan ID**.

Mediante esta configuración, se indica al controlador que envíe el flujo al puerto de entrada seleccionado y permita el paso de paquetes que contengan la VLAN asignada, todo ello a través de la interfaz web. Además, se agrega la VLAN correspondiente al grupo específico a los paquetes de salida, utilizando la instrucción **Push Vlan**. Esta instrucción implica configurarel encabezado de un paquete 802.1q, como se puede observar en las Figuras 4.15(a) y 4.15(b).



((a)) Puerto 1. ((b)) Puerto 2.

Figura 4.15: Asignación estática de VLAN 400 vía OFM.

4.7. Configuración de servidor de acceso a la red para implementación del plano deaplicación

En esta sección se presenta la implementación del servidor NAC desarrollado en Python. Esteservidor opera de manera secuencial, obteniendo información directamente desde el controlador ODL sobre los switches OvS y los hosts que conforman la topología de la SDN. Utilizandoconsultas GET, el servidor NAC extrae del controlador, las direcciones físicas (MAC) de los usuarios conectados, y luego compara esta información con los datos locales en el servidor.A partir de esta comparación, se asigna dinámicamente la VLAN correspondiente al departamento al que pertenece cada host.

4.7.1. Obtención topología SDN

En una CAN, la movilidad de los estudiantes, docentes, administradores, investigadores, entre otros, genera cambios constantes en la topología de la red debido a la necesidad de conectarse en distintas ubicaciones dentro del campus. Por esta razón, es fundamental tener conocimiento del estado actual de la topología para administrar eficientemente los recursos de red. En estesentido, la utilización de SDN ofrece una ventaja significativa: la abstracción. A través de la capa de aplicación y consultas GET utilizando el protocolo HTTP, se puede solicitar informaciónal controlador ODL, el cual actúa como servidor web. Es importante conocer la dirección específica donde se encuentra almacenada la información de la topología en el controlador ODL,para lo cual se utiliza el protocolo RESTCONF a través del plano de aplicación. El Extracto de código 4.8 proporciona la dirección a la cual se deben dirigir las consultas para obtener información sobre la topología de la red en modo operativo.

```
1 http://{controller_ip}:8181/restconf/operational/network-
    topology:network-topology/topology/flow:1/'
```

Extracto de código 4.8: Dirección url que contiene información de la topología.

El código implementado para llevar a cabo este proceso se detalla en el Apéndice C.1.

4.7.2. Validación de usuarios de la CAN

La validación de usuarios en una CAN desempeña un papel crucial al garantizar la protecciónyel uso adecuado de los recursos compartidos de la red. Este proceso verifica la identidad de los usuarios, autoriza su acceso y previene cualquier intento de acceso no autorizado, creando así un entorno seguro y confiable para los usuarios de la red. En consecuencia, es esencial implementar un sistema de validación de usuarios en el servidor NAC. Este proceso implica realizar comparaciones secuenciales entre los datos obtenidos de la SDN por el servidor NACy los archivos locales que contienen información sobre las direcciones MAC de los usuarios,los departamentos a los que pertenecen y las VLAN asignadas a dichos departamentos.

El código implementado para llevar a cabo este proceso se detalla en el Apéndice C.2.

4.7.3. Asignación de VLANs

La asignación de VLANs se lleva a cabo de forma dinámica utilizando una matriz denominada*matriz NAC*. Esta matriz se actualiza con información validada extraída de la topología SDN, lo que permite mantener un control ordenado de los eventos que ocurren en la red de área campus (CAN) a lo largo del tiempo. La *matriz NAC* almacena información importante provenientede la SDN, que incluye:

- Identificador del conmutador OvS al que está conectado el host.
- Puerto del conmutador OvS al que está conectado el host.
- Dirección MAC del host.
- Nombre de usuario de host conectado.
- Grupo al que pertenece el host conectado.
- VLAN que se asigna al host conectado.

En la Figura 4.16 se ilustra de manera gráfica un ejemplo de *matriz NAC* que es utilizada porel servidor NAC para asignar VLANs.

El código implementado para llevar a cabo este proceso se detalla en el Apéndice C.3.

4.7.4. Diagrama de implementación de servidor NAC con python

El proceso de asignación dinámica de VLANs utilizando un servidor de acceso a la red

se representa en el diagrama de flujo de la Figura 4.20, el cual resume los pasos descritos en las secciones 4.7.1, 4.7.2 y 4.7.3. El primer paso consiste en la creación de la matriz NAC. Acontinuación, se obtiene la información de la topología SDN mediante la escucha de mensajes

Identificador Conmutador OvS	Puerto OvS	Dirección MAC usuario	Usuario	Departamento	VLAN
9573	9573:1	00:90:f5:b6:be:37	user 3	Docente	200
9573	9573:2	f0:bf:97:8f:b9:8a	user 4	Docente	200
...	...	...	...	...	...

Figura 4.16: Ejemplo de Matriz NAC.

OpenFlow (PACKET IN) enviados por los conmutadores OvS cuando un host se conecta. Estos mensajes notifican al controlador ODL la presencia de un paquete que el conmutador no puede manejar por sí mismo, lo que permite al controlador tomar decisiones y acciones para procesar y gestionar ese paquete específico, así como establecer políticas o flujos en el conmutador para futuros paquetes similares. La información de la topología incluye los identificadores delos conmutadores OvS, los puertos utilizados y las direcciones MAC de los hosts conectados,lo que permite organizar y clasificar la información para iniciar la validación de los usuarios.

La validación de usuarios se realiza mediante la comparación de parámetros específicos conarchivos locales, concretamente tres archivos de texto. En primer lugar, se compara la dirección MAC detectada con las existentes en un archivo que contiene los nombres de usuario permitidos junto con sus respectivas direcciones MAC. Si el usuario se encuentra en la lista permitida, se relaciona con su dirección MAC para luego determinar a qué departamento pertenece. Debido a que el proceso es secuencial, se valida la existencia del departamento paraasignar la VLAN correspondiente. Posteriormente, se actualiza la matriz NAC y, en caso de haber cambios en la topología, se asigna la VLAN correspondiente. En caso contrario, se registra un mensaje LOG indicando que no se han detectado cambios en la SDN. Este procesose repite de forma continua, con un intervalo

de tiempo corto entre cada actualización.

El intervalo de actualización se refiere a la frecuencia con la que el servidor NAC verificalos cambios en la topología de la SDN. Si este intervalo es menor, la asignación dinámicade VLANs a un nuevo usuario se retrasará durante ese período. Para lograr una topología *plug&play* en una CAN, el servidor debe monitorear constantemente el estado de la red. Por esta razón, se ha definido el intervalo de verificación de cambios en la topología SDN en unsegundo.

user1	c8:d3:ff:75:53:8d
user2	10:e7:c6:dd:4f:07
user3	00:90:f5:b6:be:37
user4	f0:bf:97:8f:b9:8a
user5	00:2b:67:df:a3:d8
user6	e8:9a:8f:e2:95:f9
user7	60:18:95:30:09:cd
user8	28:d2:44:ff:54:c2

Figura 4.17: Usuarios con direcciones MAC.

estudiante	user1
estudiante	user2
docente	user3
docente	user4
administracion	user5
administracion	user6
investigador	user7
investigador	user8

Figura 4.18: Departamentos con usuarios.

4.8. Métricas de evaluación de la red

En el presente trabajo de titulación se han planteado tres topologías que definen la evoluciónde la manera de asignar VLANs a usuarios en una CAN. Por tal motivo, se ha diseño dos experimentos que permitan evaluar estadísticamente el funcionamiento

100	estudiante
200	docente
300	administracion
400	investigador

Figura 4.19: VLANs con departamentos.

de la red, a partir devarias métricas, tales como: *delay*, tasa de paquetes recibidos (PRR) y *throughput*.

En primer lugar, se lleva a cabo un análisis estadístico utilizando el tráfico de paquetes ICMPintercambiados entre los hosts de una misma VLAN. El tráfico generado es a partir de comando `ping`, con un total de 100 paquetes enviados. Durante este experimento, se prestará especial atención a las métricas de retraso (*delay*) y tasa de paquetes recibidos (PRR). Estas métricasde evaluación proporcionarán una visión del funcionamiento de la red tanto en su versión clásica como en su implementación SDN, lo que permitirá emitir juicios objetivos basados en los resultados obtenidos de estas métricas evaluadas. El experimento se repetirá 10 veces con las mismas características, con el fin de obtener resultados estadísticos que permitan graficarlos valores medios, así como los intervalos de confianza (95 %).

Finalmente, se realizará un segundo experimento, en el cual se transmite contenido multimedia (audio y video) entre dos *hosts* de la misma VLAN. Este experimento permitirá analizar dos métricas clave: *throughput* y *delay*. El *throughput* medirá la cantidad de datos que se pueden transmitir en un periodo de tiempo determinado, lo que reflejará la eficiencia de la red entérminos de capacidad de transmisión. Por otro lado, el *delay* medirá el tiempo de respuestaexperimentado por la red al enviar y recibir los paquetes de

contenido multimedia. El experimento se basa en la captura de paquetes Real Time Protocol (RTP), mediante el analizador de red *Wireshark*. Donde, se extraerá los datos de tiempo de los paquetes capturados para obtener el *delay*, así como el throughput por segundo de la comunicación entre los *hosts*. Deigual manera al primer experimento, los datos extraídos del analizador de red permiten graficar resultados estadísticos que permitan graficar los valores medios, así como los intervalosde confianza (95 %). Estas métricas proporcionarán resultados que permitan justificar la elección del despliegue una SDN por sobre una red clásica, demostrando su mayor idoneidad en términos de rendimiento y eficiencia.

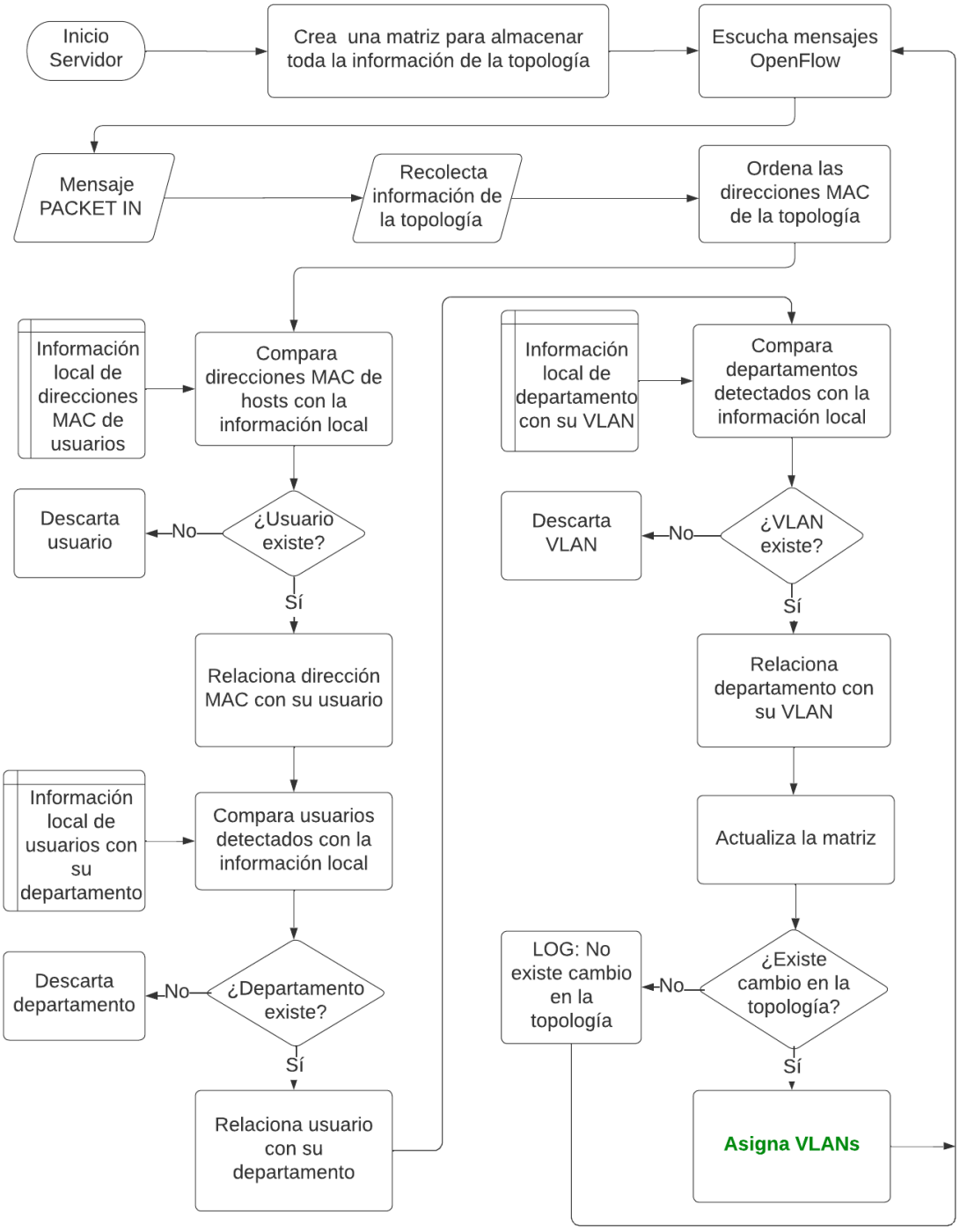


Figura 4.20: Diagrama de flujo de funcionamiento de servidor NAC.

5. Pruebas de funcionamiento y análisis de resultados

En este capítulo se exponen los resultados obtenidos a partir de los conceptos y configuraciones descritos en capítulos anteriores. Se comienza con la implementación física de la estructura rack y se aborda la evolución de las VLANs, diferenciadas por el tipo de red (clásica, SDN) y la asignación de usuarios. Además, en el Apéndice E se incluyen los resultados detallados de todas las configuraciones en cada uno de los hosts conectados a la red, lo que permite evidenciar el funcionamiento en el entorno de pruebas para las tres topologías evolutivas deVLANs. A continuación, se resumen los resultados más relevantes obtenidos.

5.1. Resultados obtenidos para el entorno de pruebas

En el Capítulo 4, se ha realizado el diseño en 3D de una estructura de rack a escala, logrando crear una estructura capaz de albergar todos los componentes de red utilizados en este trabajo. La estructura diseñada, que ha sido impresa en 3D, permite el agrupamiento de los conmutadores de red, que en este caso son tarjetas RPi integradas con OvS, así como los adaptadores USB-RJ45 que funcionan como interfaces Ethernet para los conmutadores. El resultado final se muestra en la Figura 5.1.

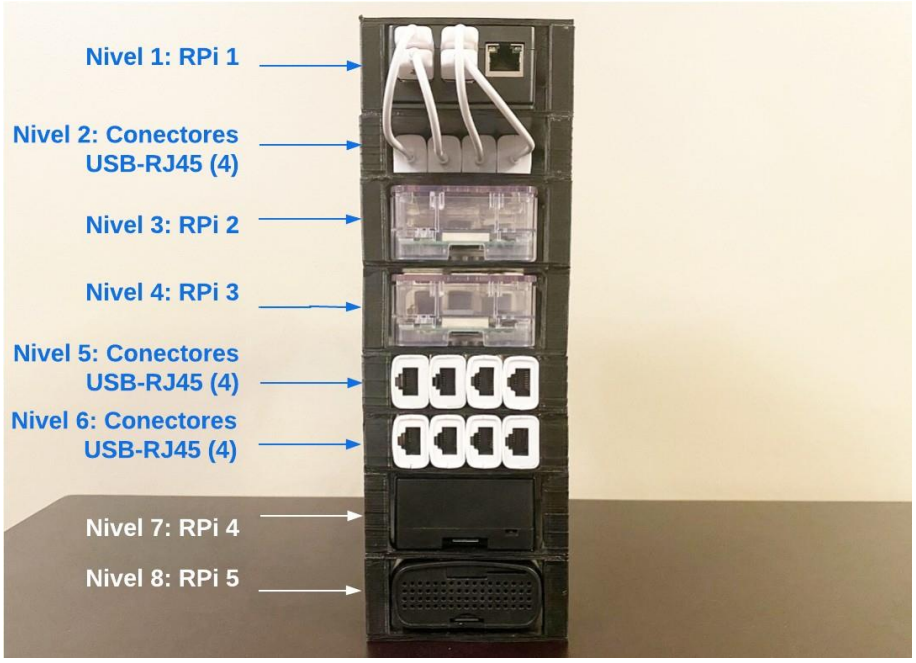


Figura 5.1: Rack a pequeña escala.

- **Nivel 1:** Tarjeta RPi 1 con funcionamiento de conmutador principal.
- **Nivel 2:** Conectores USB - RJ45 (4) con funcionamiento de interfaces Ethernet a las quese conectan los conmutadores de acceso.
- **Nivel 3:** Tarjeta RPi 2 con funcionamiento de conmutador de acceso.
- **Nivel 4:** Tarjeta RPi 3 con funcionamiento de conmutador de acceso.
- **Nivel 5:** Conectores USB - RJ45 (4) con funcionamiento de interfaces Ethernet a las quese conectan los usuarios de las conmutadores 2 y 3.
- **Nivel 6:** Conectores USB - RJ45 (4) con funcionamiento de interfaces Ethernet a las quese conectan los usuarios de las conmutadores 4 y 5.
- **Nivel 7:** Tarjeta RPi 4 con funcionamiento de conmutador de acceso.
- **Nivel 8:** Tarjeta RPi 5 con funcionamiento de conmutador de acceso.

El entorno de pruebas desarrollado cumple con el primer objetivo establecido en este trabajoal proporcionar un espacio unificado para la integración de todos los equipos tecnológicos. Además, simplifica la configuración de diversas topologías, ya que con una simple conexión entre las interfaces Ethernet de las tarjetas RPi se pueden crear varias configuraciones de red. Asimismo, permite una conexión fácil de hasta 8 clientes potenciales, lo que proporciona un entorno de prueba que puede simular tanto una red clásica como una SDN, replicando así elfuncionamiento de una CAN.

La implementación física de la topología final se presenta en la Figura 5.2. Este entorno de pruebas permite la ejecución completa del presente trabajo de titulación sin necesidad de manipular los equipos de red directamente, ya que se puede realizar la configuración de forma remota mediante la conexión a los conmutadores (VNC, SSH, Telnet, etc.) o a través de la conexión de dispositivos de entrada y salida a los puertos de las tarjetas RPi. La configuración de las interfaces de los conmutadores OvS es sencilla e intuitiva, ya que se ha utilizadolos puertos USB de las tarjetas RPi consecutivamente en horizontal y se han denominado dela misma manera, es decir, de izquierda a derecha se identifican las interfaces como `eth0` (puerto Ethernet), `eth1` y `eth2`, tal y como se muestra en la Figura 5.3.

5.2. Resultados obtenidos en asignación de VLANs en red clásica

Una vez completada la configuración de las tarjetas RPi en OvS, se procede a realizar las pruebas de funcionamiento de la implementación de las VLANs, siguiendo la primera de las tres topologías establecidas en este trabajo (Figura 4.1), que representa

Figura 5.2: Topología física completa.

una red clásica descentralizada donde cada equipo debe ser configurado de forma individual. Para evaluar el funcionamiento, se genera tráfico en la red utilizando el protocolo ICMP y el comando `ping`,como se ilustra en las Figuras E.5 y E.6. Estas figuras evidencian la comunicación exitosa entre los dos clientes utilizados en la prueba, logrando así uno de los objetivos principales de la implementación de VLANs: permitir la segmentación lógica de la red y garantizar un tráfico independiente entre las VLANs, mejorando la calidad y seguridad de la red. Además de estos beneficios, esta implementación simplifica la gestión y asignación de recursos al permitir la agrupación de dispositivos según criterios específicos, como departamento, función o tipo dedispositivo, dentro de la CAN. Las VLANs también facilitan la priorización del tráfico, lo que ayuda a controlar la congestión en cada VLAN de la CAN, donde el tráfico de datos puede ser intenso y variado. Asimismo, la implementación de VLANs reduce los costos de la red al permitir la consolidación de dispositivos y la disminución de los gastos en cables y conmutadores.

Sin embargo, al realizar esta implementación, se puede observar que estas topologías resultan algo obsoletas para una CAN, ya que, al tratarse de una red descentralizada, los dispositivosdeben ser configurados manualmente uno por uno desde sus terminales respectivas, lo que implica una mayor planificación y aumento en la complejidad de la red. Asimismo, la implementación de VLANs puede generar una carga administrativa adicional, debido a que en una red clásica cada VLAN debe ser gestionada y

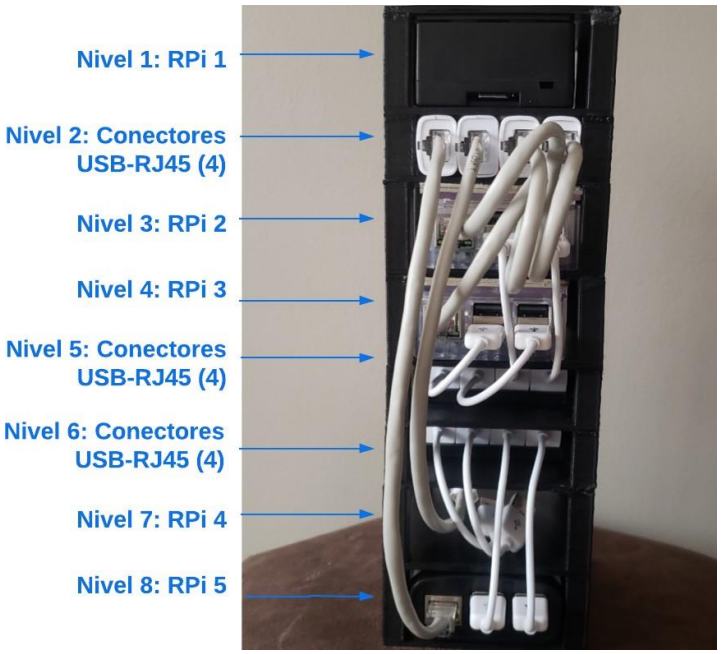


Figura 5.3: Rack a pequeña escala (vista posterior).

configurada de forma individual. Además, si surgealgún error durante la configuración, se requiere solucionarlo físicamente en el lugar donde seencuentra el equipo afectado, lo que puede ocasionar un incremento en la latencia de la red.

En síntesis, a pesar de las numerosas ventajas que proporciona la implementación de VLANsen una red de campus, como se ha descrito anteriormente, también es importante consideraralgunas desventajas, como la complejidad, la carga administrativa y la posible latencia. Por lotanto, es fundamental evaluar cuidadosamente estos aspectos antes de implementar VLANsen una red clásica, especialmente en el contexto de una CAN.

5.3. Resultados obtenidos en asignación de VLANs en SDN

A partir de los resultados obtenidos en la sección anterior, se ha identificado la necesidad deimplementar un controlador en la red para lograr un control centralizado de toda la

topologíacreada. Por tal motivo, se incorporó el controlador de red ODL, el cual actúa como el centro de control de la red. En el capítulo anterior se proporcionó una descripción detallada de la configuración del controlador ODL y la conexión de los conmutadores OvS a él, lo que permitió alcanzar la topología representada en la Figura 4.2, como se muestra en la Figura 4.11. De este modo, se logró otro objetivo clave del proyecto, que consistía en diseñar e implementarun entorno de pruebas para SDN con asignación estática de VLAN.

La asignación de VLANs se llevó a cabo de dos formas: mediante programación y a través deuna interfaz web, y puede analizarse en dos aspectos distintos. Al implementar la asignación de VLANs en una red definida por software (SDN), se logró centralizar las funciones de controlde la red, lo que permitió identificar la topología completa y los hosts conectados a ella. Sin embargo, es fundamental considerar cuidadosamente los métodos de asignación de VLANs,ya que esto puede introducir cierta complejidad en la gestión de la red.

En este tipo de asignación de VLANs se eliminan varios procesos que el administrador de lared realiza, en comparación a una CAN sin controlador SDN. Estos procesos son: visita técnica al usuario que solicita la VLAN, vista técnica al conmutador al que se conecta el usuario,configuración sobre el conmutador en cuestión. Todos los procesos mencionados previamente son reemplazados por la configuración remota desde el controlador SDN, lo que reduce significativamente los tiempos de aprovisionamiento de servicios a los usuarios de la red. Además,este paradigma de red permite el control y manejo de la red de manera remota, mediante la configuración de flujos en base a los requerimientos de los usuarios.

5.3.1. Asignación estática de VLAN con programación de flujos

Al concluir con la configuración del controlador ODL y los conmutadores OvS, se logró abstraerel plano de datos y control (programable) de una red definida por software. Esta abstracciónbrindó una mayor flexibilidad en las instrucciones de gestión de la red, lo que permitió la creación de APIs y/o instrucciones específicas utilizando archivos XML o JSON.

Utilizando los archivos JSON mostrados en los Extractos de código 4.6 y 4.7, se pudo configurar los puertos 1 y 2 del conmutador OvS con ID 4294976775 (nivel 3 del rack) para que transmitan y reciban paquetes pertenecientes a la VLAN 400. Este proceso se

realizó desde un equipo con sistema operativo Ubuntu 22, lo que permitió la abstracción necesaria para configurar la VLAN de manera centralizada, sin que el administrador de red tuviera que desplazarse físicamente al lugar donde se encuentra el conmutador. La ventaja de este enfoque radica enla significativa reducción del tiempo necesario para configurar y poner en funcionamiento la red SDN, especialmente en comparación con el tiempo requerido para realizar la misma tarea enuna CAN con conmutadores físicamente separados a una distancia considerable. En otras palabras, el administrador de red puede configurar los puertos de un conmutador, e incluso todos los puertos de todos los conmutadores de la SDN, desde su propio puesto de trabajo utilizando el protocolo OpenFlow. Esto permite que el tiempo requerido para asignar VLANs en unared SDN sea considerablemente menor en comparación con el tiempo que el administrador necesita para configurar una red sin SDN.

En cuanto a la configuración del controlador ODL mediante archivos JSON, el administradorde red tiene la capacidad de modificar los parámetros del flujo agregado, como el `table_id`, `id`, `priority`, `hard-timeout` e `idle_timeout`, así como los parámetros de coincidencia, como `in-port` y `vlan-match`. Posteriormente, se realiza la configuración de la instrucción para los paquetes de salida, donde se asigna la VLAN correspondiente con su encabezado 802.1q. Finalmente, se logró establecer exitosamente la comunicación entre los clientes conectados a uno de los conmutadores OvS dentro del rack diseñado y en funcionamiento. Para comprobar la correcta configuración de la conexión, se llevaron a cabo pruebas mediante el envío de paquetes ICMP utilizando el comando `ping`.

5.3.2. Asignación estática de VLAN con interfaz web OFM

Las redes definidas por software (SDN) ofrecen la posibilidad de implementar aplicaciones enel controlador mediante la interfaz norte, lo que facilita y permite la configuración de la red. Una de estas aplicaciones es proporcionada por Cisco (OFM). Esta API permite implementar una interfaz web con diversas funcionalidades, como el mapeo de la topología de la red, la agregación de flujos, la revisión de las tablas de flujo y la obtención de estadísticas de la red.

En consecuencia, la programación de archivos JSON es reemplazada por OFM, ya que a través de su interfaz web es posible realizar las mismas configuraciones y agregar flujos

a lared mediante una serie de opciones habilitadas, tal como se muestra en la Figura 4.15(b). Entre las capacidades de OFM se incluye la capacidad de agregar VLANs a los paquetes de salida de un puerto específico y aceptar paquetes entrantes etiquetados con el protocolo 802.1q.

En este contexto, el uso de OFM ha permitido reducir significativamente el tiempo de configuración de la SDN, ya que no fue necesario programar el archivo JSON desde cero. MedianteOFM, el administrador de red tuvo que accionar únicamente en la habilitación de los parámetros de configuración para la agregación y aceptación de VLANs en puertos específicos. Una vez configurado el flujo, simplemente completando el “formulario” que se genera, OFM construyó automáticamente el archivo JSON y lo envió al controlador.

En conclusión, el cambio de un enfoque de red tradicional a uno basado en SDN presenta claros beneficios para el administrador de red, ya que se logra una notable reducción en el tiempo necesario para configurar y supervisar la red ya que se eliminan varios procesos que se aplican para las redes clásicas. Además, al emplear aplicaciones y automatizaciones, es posible disminuir aún más el tiempo de configuración y minimizar errores de programación gracias al uso de APIs como OFM, lo que resulta en una red más eficiente en general.

En el Apéndice E.2.1 se ilustran las pruebas de conexión y configuraciones de los clientes conectados a la red.

5.3.3. Asignación dinámica de VLAN con servidor NAC

En la sección anterior (sección 5.3.2), se utilizó una interfaz de programación de aplicaciones(API) de Software Defined Networking (SDN) desarrollada por CISCO para asignar VLANs mediante la configuración manual de flujos que contienen información del encabezado del protocolo 802.1q. Esta solución resulta adecuada cuando la topología de la red es de escalareducida, con pocos conmutadores y hosts. Sin embargo, en una red de área de campus (CAN), donde los conmutadores forman un centro de datos con múltiples racks que conectan varioshosts, la asignación estática de VLANs se vuelve ineficiente. Por lo tanto, en la sección 4.7, se presentó la configuración de un servidor NAC desarrollado en Python. Este servidor NACpermite la asignación dinámica de VLANs basándose en la distribución de los hosts y sus direcciones MAC.

Una vez que se ha ejecutado el código del Apéndice C, se logra implementar la

asignación dinámica de VLANs a los hosts conectados en la red. Este procedimiento permite al administrador de red centrarse en la configuración de los archivos locales de acceso a la red, los cuales requieren la adición o eliminación de usuarios en la red y su asignación a departamentos específicos. El servidor utiliza la interfaz Norte de la SDN para recopilar, almacenar y validar la información de la topología. En nuestro caso, obtenemos información sobre la configuraciónde la SDN, incluyendo las conexiones entre los conmutadores OvS y los hosts, a través de consultas GET. Utilizando esta información, hemos creado nuestra propia API que funciona como servidor de acceso a la red, verificando las direcciones MAC de los usuarios conectados en relación con los usuarios permitidos según los archivos locales.

Finalizado el proceso de validación de direcciones MAC (como se muestra en la Figura 4.20),el servidor NAC logra asignar VLANs de forma automática, abordando así el problema planteado inicialmente en este trabajo. La solución propuesta por el servidor NAC supera la limitaciónde la asignación estática de VLANs y elimina la necesidad de configurar cada VLAN por separado. Además, se eliminan más procesos que el administrador de red efectúa, en comparación a una CAN con red clásica y con SDN, pero sin asignación dinámica de VLANs. Esto conlleva una reducción significativa en el proceso de asignación de VLANs, ya que en este punto el controlador ODL realiza todo el procedimiento de manera dinámica. La limitación de este enfoque radica en el tiempo de actualización de la topología, lo que está directamente relacionadocon la capacidad de procesamiento del servidor NAC.

Al comparar los resultados obtenidos en este trabajo con los analizados en la sección 2, se identifican similitudes con los hallazgos de otros estudios. Por ejemplo, al implementar la tercera topología propuesta, se observa una reducción de procesos similar a la encontrada en los trabajos de [11] y [18]. La integración de la asignación dinámica de VLANs en la red haceque la infraestructura de la CAN sea más centralizada y optimiza los procesos asociados a la configuración manual presentes en los escenarios anteriores. Además, de manera similar a los estudios descritos en [15], al realizar la asignación dinámica de VLANs, el servidor NAC implementado permite a los usuarios conectarse a su VLAN correspondiente en cualquier parte dela CAN, siempre y cuando su dirección física esté registrada en los ficheros locales y el dispositivo se conecte a la red a través de Ethernet. El servidor NAC se encarga automáticamentede validar y asignar VLANs a los usuarios, lo que optimiza la red y simplifica su administración. Similarmente, al trabajo de [13], aunque en dicho estudio la implementación no utilizó SDN, sedestaca la importancia de incorporar un servidor de acceso a la red para validar y autenticara los usuarios, brindando un control

e integridad y evitando accesos no autorizados. En estetrabajo, al integrar el servidor NAC en una infraestructura SDN, se garantiza la seguridad dela red de manera similar a lo propuesto por [13], pero además se obtiene una centralización que mejora el rendimiento y la administración de la red en general.

5.3.4. Comparación de procesos en la evolución de asignación VLANs

Las Figuras desde 5.4 hasta 5.6 ilustran de manera gráfica los procesos involucrados en la asignación de VLANs a los usuarios en cada una de las topologías propuestas en este trabajo. Es evidente que a medida que se progresa desde una red clásica hacia una SDN con asignación dinámica de VLANs, se observa una reducción significativa en la cantidad de procesos requeridos para su administración.

5.4. Resultados y análisis de métricas evaluadas

En esta sección se presentan los resultados obtenidos tras llevar a cabo los experimentos mencionados en el capítulo 4, con el propósito de evaluar el rendimiento tanto de la red clásica

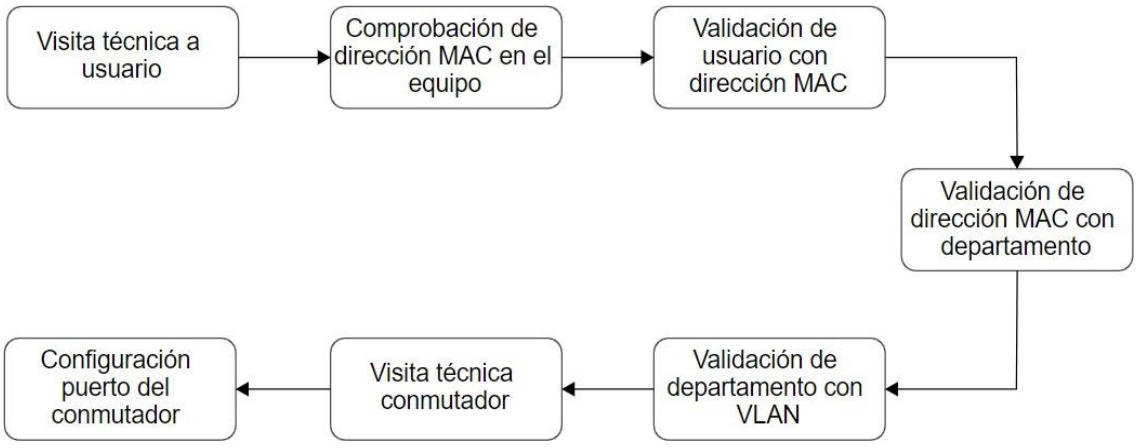


Figura 5.4: Procesos para asignación de VLAN en red clásica.

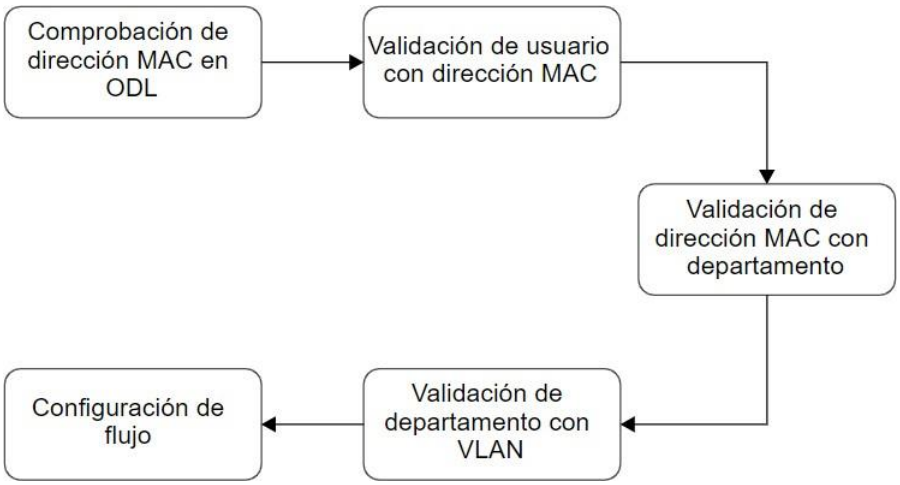


Figura 5.5: Procesos para asignación estática de VLAN en SDN.

como de la SDN frente a diferentes tipos de tráfico de red.

5.4.1. Resultados frente a tráfico ICMP

El primero de los dos experimentos consistió en observar el funcionamiento de la red al momento de intercambiar mensajes Internet Control Message Protocol (ICMP) entre dos *hosts* pertenecientes a la misma VLAN. En este experimento se ha evaluado el *delay*, así como el *PRR*. Para la obtención de las gráficas, se ha realizado 10 repeticiones, enviando en cada unade estos 100 paquetes ICMP desde un *host* a otro.

Los resultados obtenidos se ilustran en las Figuras 5.7 y 5.8.

El primer gráfico (Figura 5.7) muestra que la red clásica tiene un menor delay con un promedio de 1.227 ms, seguida de la SDN con asignación estática de VLANs con 1.247 ms, y finalmente la SDN con asignación dinámica de VLANs con 1.259 ms.

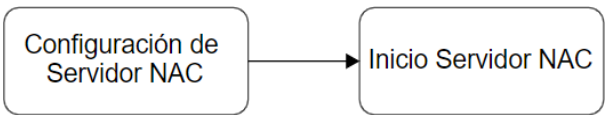


Figura 5.6: Procesos para asignación dinámica de VLAN en SDN.

La diferencia entre las tres opciones es mínima, oscilando en el rango de 20 µs a 32 µs entre la red clásica y las SDN con asignaciones estática y dinámica de VLANs. En conclusión, la red clásica presentaun mejor desempeño. Sin embargo, el intervalo de confianza de la red clásica indica que susvalores varían más en comparación con los

resultados de la SDN, lo que sugiere que la SDNcon asignación dinámica de VLANs iguala el retardo de la red clásica.

Esto se debe a la comunicación directa existente entre los *hosts* que forman el experimento.Es decir, los usuarios se encuentran conectados al mismo conmutador OvS, donde se realizael procesamiento (comunicación) de forma local. Sin embargo, esto no sucede en una SDN, donde el conmutador de acceso debe comunicarse con el controlador ODL, teniendo así un retardo en la comunicación. En resumen, en la red clásica se elimina la necesidad de comunicación constante entre el dispositivo de la red (switch OvS) y el controlador, lo que disminuyeel retraso en el procesamiento de los paquetes de datos, en este caso de tipo ICMP.

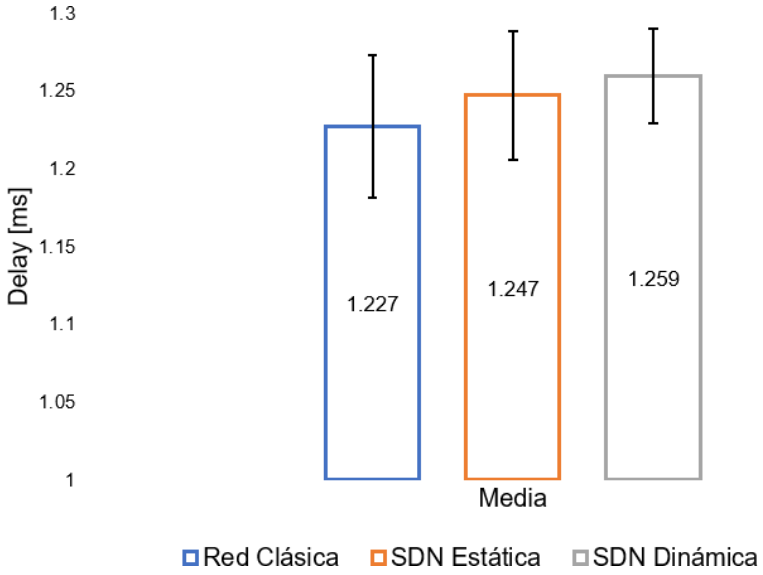


Figura 5.7: *Delay* promedio tráfico ICMP.

En la Figura 5.8, se presentan los resultados de la tasa de entrega de paquetes ICMP. En este caso, a diferencia de los resultados obtenidos en el análisis previo, tanto la SDN con asignación dinámica como la estática muestran una mejor tasa de entrega de paquetes, con un 97.8 % y un 95.6 %, respectivamente, en comparación con el 91.7 % obtenido en la red clásica. Estose debe a la gestión centralizada que ofrece la implementación de una SDN, lo que permite tener un control efectivo del plano de datos a través del plano de control. Al tener un mayor control sobre el direccionamiento de los paquetes, en este caso a través de la configuración de VLANs en los puertos de los conmutadores OvS, se optimiza el flujo de datos y se minimizala pérdida de paquetes, tal como se puede observar en la Figura 5.8.

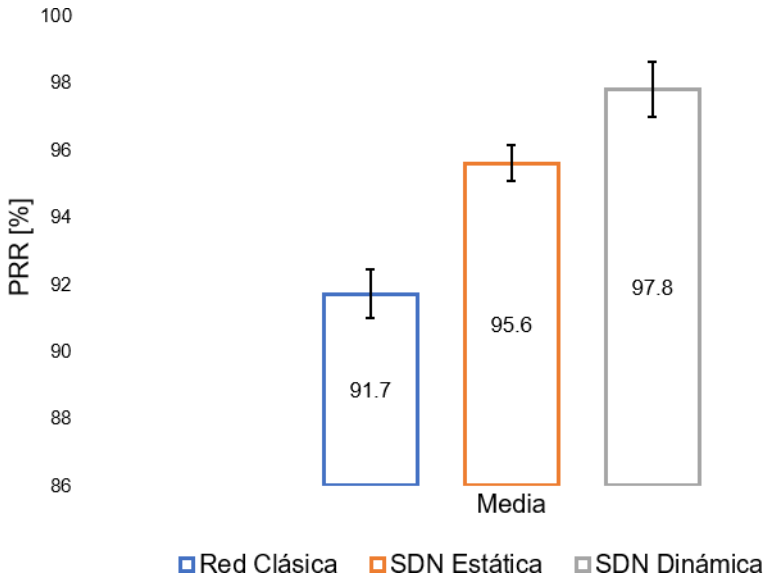


Figura 5.8: PRR promedio tráfico ICMP.

5.4.2. Resultados frente a tráfico RTP

El segundo experimento consistió en la evaluación del comportamiento de la red cuando estase somete intensidades de tráfico elevadas y variables. Por tal motivo, se ha realizado la captura de paquetes RTP al transmitir contenido de audio y video con el uso del software VLC. Eneste caso un *host* actúa como servidor, mientras que otro es el cliente. Además, los usuariosson pertenecientes a la misma VLAN.

En la Figura 5.9, se presentan los resultados del análisis del delay generado durante la transmisión, y se observa una tendencia similar a la obtenida en el primer experimento. Es decir, la red clásica muestra un menor delay con 91.7 ms, en comparación con los 95.6 ms y 97.8 ms obtenidos en las SDN con asignación estática y dinámica, respectivamente. Sin embargo,la diferencia en este segundo experimento es mucho más significativa en comparación con el primero siendo estas de 38.949 ms y 49.936 ms entre la red clásica respecto las SDN con asignación de VLANs estáticas y dinámicas, respectivamente. Esto se debe principalmente al tipo de tráfico utilizado, ya que al tratarse de paquetes RTP, requieren un mayor tiempo de procesamiento. Como se detalló en el análisis del primer experimento, existe un retardo adicional debido al proceso de comunicación entre el conmutador OvS y el controlador, un proceso queno se presenta en la red clásica. Esta es la principal razón de las diferencias observadas en los resultados.

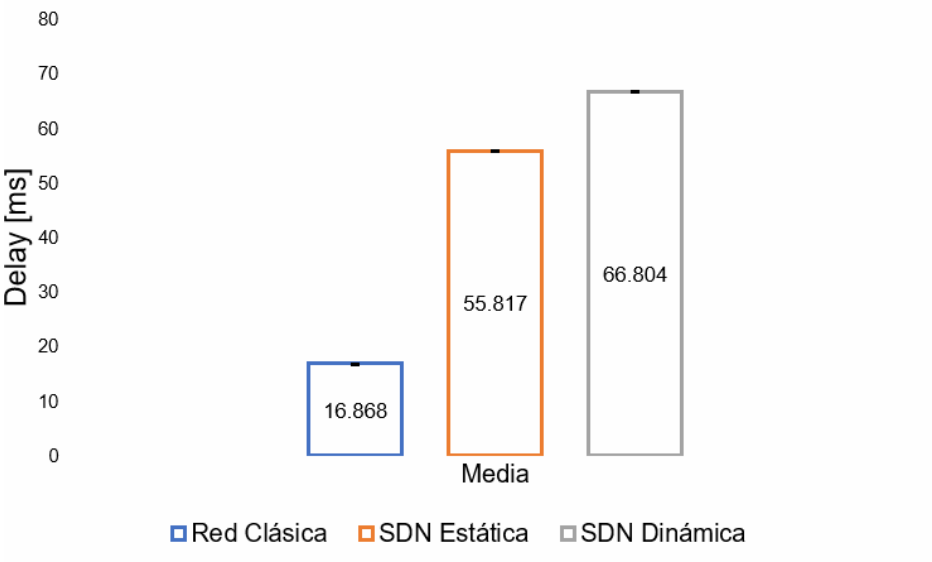


Figura 5.9: *Delay* promedio transmisión de audio y video.

En la Figura 5.10, se presentan los resultados del análisis del *throughput* en el *host* cliente. Lagráfica muestra que las SDN con asignación de VLANs, tanto estática como dinámica, presentan un mejor rendimiento en esta métrica, con 239.82 kbps y 239.256 kbps, respectivamente,en comparación con los 236.162 kbps obtenidos en la red clásica. Estos resultados van de lamano a los analizados en el primer experimento (PRR), ya que este resultado se debe nuevamente a la centralización de la red proporcionada por el controlador, así como a su capacidadde adaptarse a las demandas de tráfico mediante el uso de VLANs en el plano de datos. Esto contribuye a garantizar un flujo de datos constante, lo que se traduce en un mayor *throughput*en la recepción.

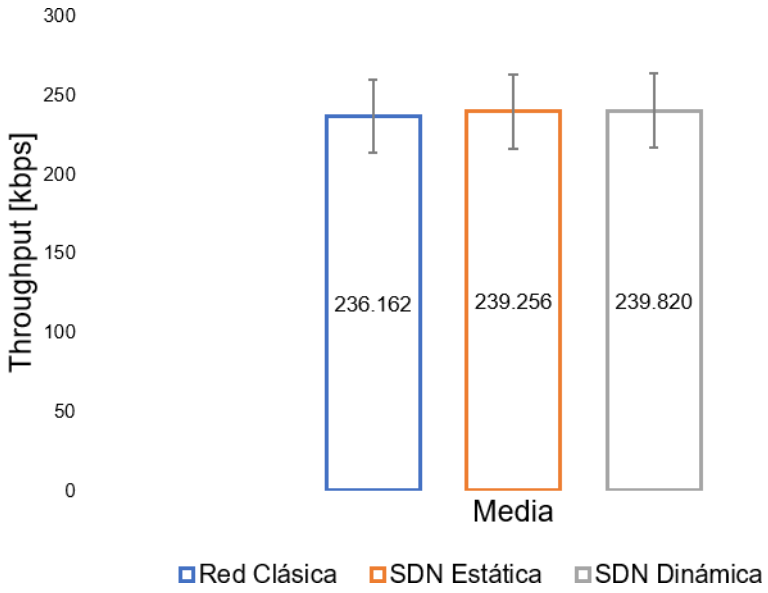


Figura 5.10: *Throughput* promedio transmisión de audio y video.

6. Conclusiones

- En el presente trabajo de titulación se implementó un entorno de pruebas para configurar VLANs en diferentes escenarios de redes CAN. El entorno constó de tres etapasque reflejaron la evolución de las redes CAN. En la primera etapa, se emuló una CAN utilizando tarjetas RPi como conmutadores. Los resultados obtenidos revelaron que la administración de la red era poco eficiente y que la configuración de VLANs resultaba imposible en una CAN convencional. En la segunda etapa, se implementó el controladorODL para emular una SDN. Los resultados demostraron que gracias a la abstracción dela capa de datos y control, el administrador de red podía controlar todos los recursos mediante la programación orientada a fines específicos, como en el caso de la asignaciónde VLANs. Finalmente, en la tercera etapa, se desarrolló un servidor de acceso a la red(NAC) en la capa de aplicación de la SDN. Este servidor NAC permitió asignar VLANs de manera dinámica, reduciendo así todos los procesos de administración y programaciónasociados.
- Durante la implementación del escenario de pruebas, se enfrentó el desafío de no contarcon los recursos adecuados para instalar un servidor NAC preexistente. Sin embargo, se pudo superar esta limitación mediante el desarrollo de un servidor personalizado utilizando el lenguaje de programación Python. Este servidor se encarga de autenticar a los usuarios utilizando archivos locales como base de datos. Esta solución permitió explotar las capacidades de la SDN, brindando una alternativa funcional y adaptada a laslimitaciones de recursos disponibles.
- La asignación dinámica de VLANs ofrece una mayor flexibilidad y adaptabilidad a la red, además de reducir considerablemente el tiempo y esfuerzo requeridos para configurar y asignar VLANs. Esta mejora se pudo observar a medida que se progresaba en la transición desde las redes clásicas hasta alcanzar la asignación dinámica de VLANs en una infraestructura de SDN. Por lo tanto, se concluye mencionando que la mejor opción parala administración de VLANs en una CAN con SDN debe realizarse de manera dinámicacon el uso de servidores de acceso a la red que contengan la información necesaria quepermita autenticar a los usuarios para la correcta asignación de VLANs.
- Además, al analizar el funcionamiento de las redes creadas en este trabajo, se logró definir a la red SDN con asignación dinámica de VLANs como la mejor opción. Esto

ya queal analizar las gráficas de las Figuras 5.8 y 5.10 referentes al PRR y *thoughput* de la red,se pudo observar que una SDN arrojó mejor resultados en ambos casos. Sin embargo,

este nuevo paradigma de redes de computadores tiene una desventaja, vista en las Figuras 5.7 y 5.9, donde el delay es menor en una red clásica, donde no es necesaria la comunicación y procesamiento de un controlador.

- En conclusión, se lograron cumplir los objetivos establecidos al diseñar e implementar un entorno de pruebas funcional que facilitó el análisis y estudio de la asignación de VLAN en redes CAN. Además, en el Anexo F se presentó el procedimiento recomendado para realizar la incorporación de VLANs dinámicas sobre SDN, extendiendo el presentetrabajo hacia un entorno de aplicación real.

A. Diseño de estructura para el montaje del entorno de pruebas

En primer lugar, se establecen las medidas exactas de los componentes que conformanla estructura del rack. El elemento principal de las topologías de red es el conmutador, elcual está siendo emulado mediante el uso de las tarjetas RPi. Las dimensiones de cadauna de estas tarjetas, junto con su correspondiente carcasa, son las siguientes:

- RPi 1: 63mm x 94mm x 30.08mm
- RPi 2: 62mm x 90.07mm x 34mm
- RPi 3: 67.3mm x 92.2mm x 34mm
- RPi 4: 62mm x 90.07mm x 34mm
- RPi 5: 63mm x 94mm x 30.08mm

Dado que las carcasas de las tarjetas tienen dimensiones diferentes, se ha decidido diseñar el rack de acuerdo a las medidas de la tarjeta RPi 3, ya que sus dimensiones son mayores que las demás. De lo contrario, se presentarían dificultades durante el procesode montaje.

- El siguiente paso consiste en tomar las medidas de los adaptadores USB-RJ45 que seutilizarán como interfaces de red en los diferentes conmutadores. En este caso, es suficiente con una sola medida, ya que todos los adaptadores provienen del mismo fabricante y tienen las mismas dimensiones. A continuación, se resumen las medidas correspondientes:
  - 23mm x 59.05mm x 11mm

- Una vez que se han obtenido las medidas de los diversos dispositivos, se procede ala creación del diseño 3D utilizando el software Google Sketchup. El diseño del rack vertical se basa en la construcción de múltiples plataformas destinadas a los diferentescomponentes, que al ser ensambladas formarán el rack deseado. En la Figura A.1 se muestra el diseño de la plataforma específicamente diseñada para las tarjetas RPi.
- Posteriormente, utilizando las medidas obtenidas para los adaptadores USB-RJ45, se procede al diseño de la plataforma correspondiente a stos dispositivos, la cual se muestra en las Figuras A.2 y A.3. A diferencia de la plataforma para las tarjetas RPi,

esta plataforma ha sido diseñada con una muesca en su interior para garantizar la estabilidadde los adaptadores.

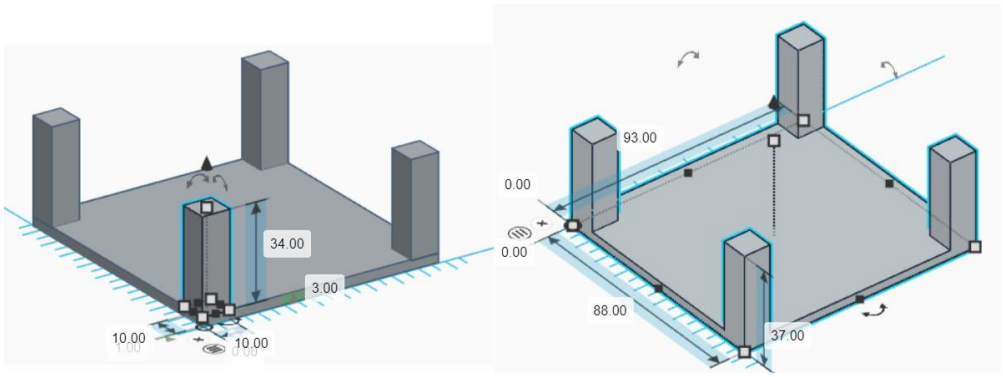


Figura A.1: Plataforma para tarjeta RPi.

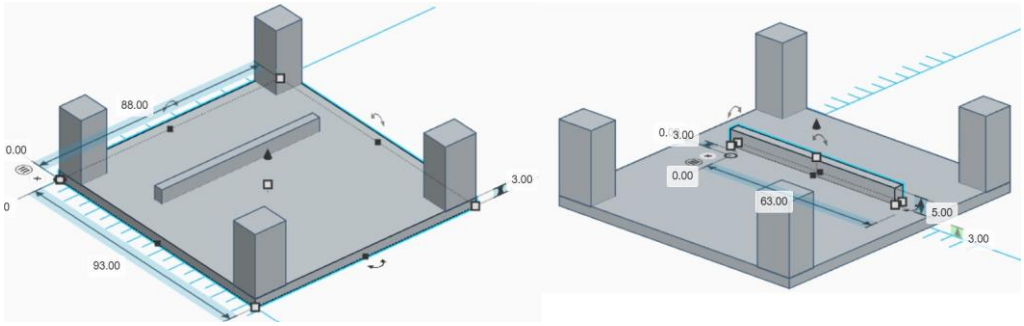


Figura A.2: Plataforma adaptadores Ethernet.

- Una vez diseñadas las diversas plataformas para los componentes, el siguiente paso consiste en ensamblarlas según la distribución elegida, obteniendo así el rack a escalamostrado en la Figura A.4.

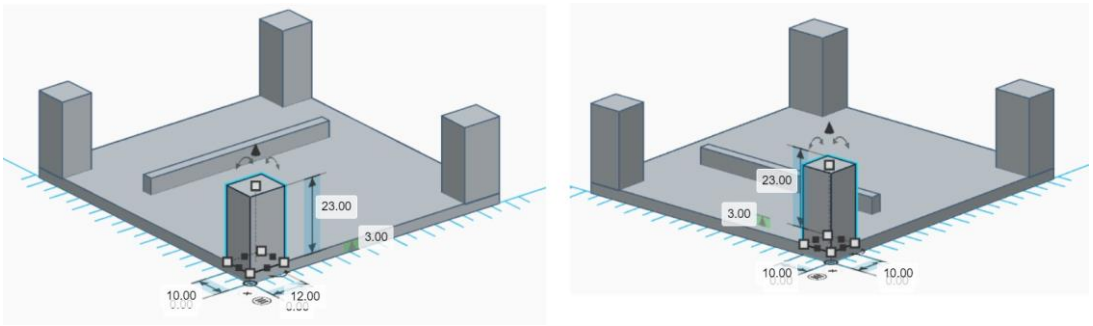


Figura A.3: Plataforma adaptadores Ethernet.

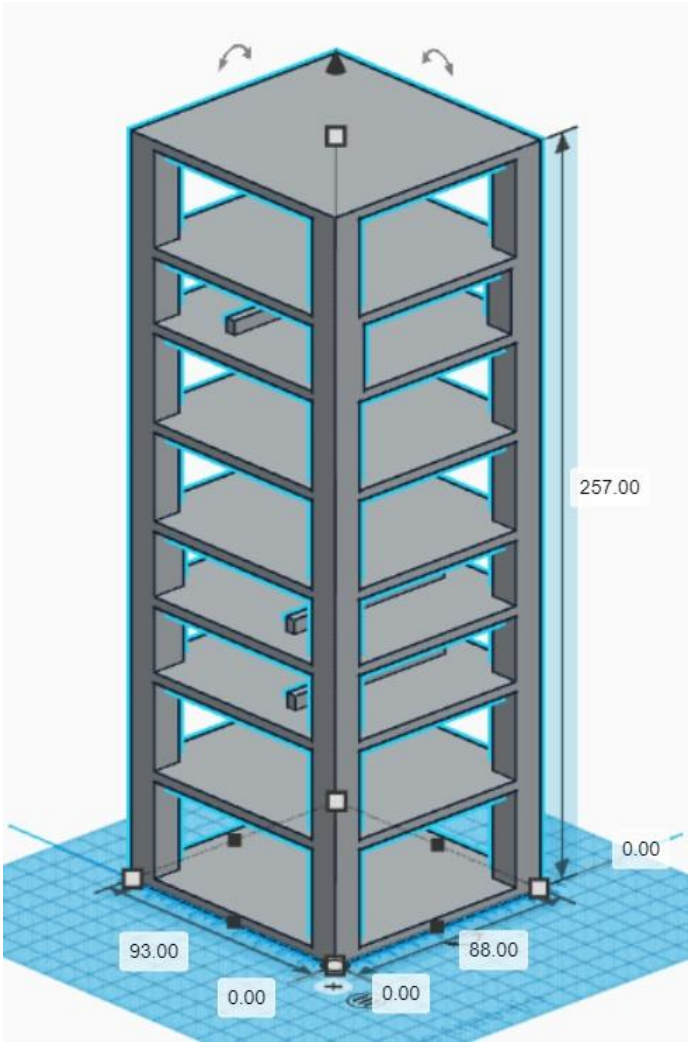


Figura A.4: Diseño rack 3D

B. Configuraciones de software

B.1. Implementación de servidor DHCP

En primer lugar, es necesario contar con el servicio que permita a la tarjeta RPi actuar comoservidor DHCP, para ello se ingresa la siguiente sentencia en la terminal:

```
1    sudo apt-get install isc-dhcp-server -y
```

Extracto de código B.1: Descarga servidor DHCP.

A continuación, se declara una dirección IP estática en la interfaz Ethernet a la cual se conectan los hosts que solicitan una dirección IP al servidor DHCP. Esto se logra modificando el archivo de configuración del servicio ubicado en la ruta `/etc/dhcpcd.conf` e ingresando las siguientes líneas:

```
1    static value
2        interface Nombre_Interfaz
3        static
4        ip_address=Direccion_IP/Máscarastatic
5        routers=Direccion_GW
```

Extracto de código B.2: Generación Internet Protocol (IP) estática.

Donde, la primera línea indica el tipo de dirección (estática). Dentro de este bloque, se defineel nombre de la interfaz a la cual se le asignará la dirección IP estática (por ejemplo, eth0) y se especifica su dirección IP. Además, se define la dirección del gateway y se finaliza indicandola dirección IP de un servidor DNS. Para aplicar los cambios realizados, es necesario reiniciar la tarjeta mediante el comando correspondiente en la terminal (`reboot`).

El siguiente paso consiste en editar el archivo de configuración del servidor DHCP, que se encuentra en la ruta `/etc/dhcp/dhcpd.conf`. Al final de este archivo, se añade la siguienteconfiguración:

```
subnet Red_IP netmask Máscara_Red {
      range Lim_Inf_direcciones_IP Lim_Sup_direcciones_IP;
      option routers Direccion_GW;
      option domain-name-servers direccion_DNS;
}
```

Extracto de código B.3: Configuración de rango de direcciones IP.

El Extracto de código B.3 configura la tarjeta RPi como servidor DHCP. En primer lugar, se define la red en la que se aplicará el servicio, junto con la máscara de red correspondiente. Acontinuación, se establece el rango de direcciones IP que se asignarán a los hosts, así comola dirección del gateway y del servidor DNS. Por último, se guardan los cambios realizados enel archivo y se inicia el servidor DHCP mediante el siguiente comando en la terminal:

```
sudo dhcpd -cf /etc/dhcp/dhcpd.conf
```

Extracto de código B.4: Comando para iniciar el servidor DHCP.

B.2. Instalación de Open vSwitch

Antes de comenzar la instalación, se recomienda realizar una actualización de los paquetes del sistema, con el comando `apt update`. A continuación, se procede a descargar la versiónespecífica de OvS que se desea instalar utilizando el comando `wget`. En este caso, se ha elegido la última versión dentro de la rama LTS de OvS, que es la versión `2.17.5`. Para llevar a cabo la descarga, se utiliza el siguiente comando:

```
wget https://www.openvswitch.org/releases/openvswitch-
2.17.5.tar.gz
```

Extracto de código B.5: Comando para la descarga de OvS.

Además, es necesario que el sistema operativo cuente con ciertas dependencias específicaspara el correcto funcionamiento de OvS, mismas que se instalan empleando el siguiente comando:

```

```

Extracto de código B.6: Instalación de dependencias para OvS.

En este punto, es necesario obtener la versión de los “headers” del kernel de Linux en la tarjeta RPi. Para ello, se utiliza el comando de la primera línea del Extracto de código B.7. Una vez que se ha identificado la versión de los “headers” de Linux, se procede a reemplazar el valor obtenido por VERSION en el siguiente comando (por ejemplo, 4.9.0-6). Los “headers” especificados se encargan de instalar y construir las dependencias necesarias específicamente para la versión de “headers” seleccionada:

```
1  sudo apt-cache search linux-headers
2  sudo apt-get install -y linux-headers-VERSION-rpi
3  sudo ./configure --with-linux=/lib/modules/VERSION-rpi/build
```

Extracto de código B.7: Instalación y configuración de “headers” sobre RPi.

El siguiente paso consiste en la construcción (`make && make install`) del framework descargado, teniendo en cuenta que se debe ejecutar en modo superusuario dentro de la terminal. Una vez finalizada la construcción de OvS, es posible verificar los módulos agregados. Paraello, se accede a la ruta `datapath/linux` dentro del directorio de OvS descomprimido previamente, y se realiza la verificación ejecutando los siguientes comandos:

```
1  modprobe
2  openvswitchcat
```

Extracto de código B.8: Verificación del módulo OvS.

Sin embargo, al ser la primera vez que se instala OvS, el módulo no suele asignarse automáticamente. Para asignar el módulo de OvS al sistema operativo de la tarjeta RPi, se ejecutan los siguientes comandos (extracto de código B.9). Además, es posible verificar la agregación del módulo utilizando el comando `cat`. Si el módulo se ha agregado correctamente, se mostrarála palabra “openvswitch” en la consola.

```
1  echo "openvswitch" >> /etc/modules
2  cat /etc/modules
```

Extracto de código B.9: Agregación del módulo OvS.

Una vez que se ha agregado el módulo de OvS al sistema operativo, se procede a crear el archivo de configuración, así como los directorios y bases de datos necesarios para el funcionamiento adecuado de OvS. Para realizar esta configuración, se ejecutan los

siguientes comandos:

```
touch /usr/local/etc/ovs-
vswitchd.confmkdir -p
/usr/local/etc/openvswitch mkdir -p
/usr/local/var/run/openvswitch
ovsdb-tool create /usr/local/etc/openvswitch/conf.db vswitchd/
```

Extracto de código B.10: Generación de directorios y ficheros necesarios para OvS.

Posteriormente, se crea un script que configura el servidor de base de datos, el demonio queactúa como conmutador y se encarga de iniciar el servicio de OvS. El contenido del script semuestra a continuación:

```
ovsdb-server    --remote=punix:/usr/local/var/run/openvswitch/db.sock
      \
              --remote=db:Open_vSwitch,Open_vSwitch,manager_options
                   \
                 --private-key=db:Open_vSwitch,SSL,private_key \
                 --certificate=db:Open_vSwitch,SSL,certificate \
                 --bootstrap-ca-cert=db:Open_vSwitch,SSL,ca_cert \
                 --pidfile     --
   detachovs-vswitchd --pidfile
   --detach ovs-vsctl  --no-wait
   init
```

Extracto de código B.11: Script para iniciar OvS.

Por último, se otorgan los permisos necesarios al script creado y se procede a ejecutarlo. Con todos los pasos descritos en esta subsección, la herramienta OvS estará en funcionamiento ylista para su uso.

```
chmod +x
scriptsudo
```

Extracto de código B.12: Asignación de permisos e inicialización de OvS.

B.3. Obtención controlador OpenDaylight

Para proceder con la descarga, se utiliza el comando `wget` y se introduce el siguiente

código en la terminal de Linux:

```
wget https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/integration/distribution-karaf/0.4.4-Beryllium-SR4/distribution-karaf-0.4.4-Beryllium-SR4.tar.gz
```

Extracto de código B.13: Comando para la descarga controlador ODL.

Luego, se procede a descomprimir el archivo descargado utilizando el comando `tar`. Antesde instalar el controlador ODL, es crucial considerar las dependencias necesarias para su correcto funcionamiento. En el caso de la versión `0.4.4-Beryllium-SR4`, se requiere específicamente la versión 8 de Java. Para descargar esta dependencia, se ejecuta el siguientecomando:

```
sudo apt-get install -y maven git openjdk-8-jre openjdk-8-jdk unzip
```

Extracto de código B.14: Instalación OpenJDK 8.

Después de haber descargado la versión necesaria de Java, el siguiente paso consiste en exportar la variable de entorno correspondiente. Esta variable se utiliza para definir la versiónprincipal de Java en el sistema operativo. Para ello, se introduce en la terminal (modo superusuario) la ruta donde se encuentra la librería, de la siguiente manera:

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
```

Extracto de código B.15: Exportación de la variable de entorno Java.

Una vez se han cumplido los requisitos de software solicitados por ODL, se procede a iniciarel controlador. Para ello, primero se ingresa a la carpeta descomprimida y se ejecuta el script`karaf` ubicado en la ruta `/bin/karaf`, mediante el siguiente comando:

```
./bin/karaf
```

Extracto de código B.16: Ejecución de script para correr ODL.

Cuando el controlador ha sido iniciado, se despliega su terminal y se puede proceder a la instalación de las herramientas necesarias para el funcionamiento adecuado de ODL, así como para reconocer y visualizar los dispositivos de red mediante la interfaz web. Estos pasos se realizan ingresando los siguientes comandos en la terminal desplegada:

```
feature:install odl-restconf-all
feature:install odl-openflowplugin-all
feature:install odl-l2switch-all
feature:install odl-mdsal-all
feature:install odl-yangtools-common
feature:install odl-dlux-core
feature:install odl-dlux-node
feature:install odl-dlux-yangui
feature:install odl-dlux-yangvisualizer
```

Extracto de código B.17: Características instaladas en ODL.

La funcionalidad de cada una de las características instaladas se detalla a continuación:

- **odl-restconf:** habilita el acceso REST API a la MD-SAL, incluido el almacén de datos.
- **odl-openflowplugin:** permite descubrir y controlar los conmutadores OpenFlow y la topología entre ellos.
- **odl-l2switch:** proporciona reenvío a nivel de la capa L2 (Ethernet) a través de conmutadores OpenFlow conectados y compatibilidad con el seguimiento de hosts.
- **odl-mdsal:** proporciona una capa de abstracción de datos que permite a los desarrolladores manipular y gestionar datos de forma programática, independientemente delprotocolo subyacente.
- **odl-yangtools-common:** proporciona una serie de utilidades y herramientas para trabajar con modelos YANG.
- **odl-dlux-core:** proporciona una interfaz de usuario web que permite a los usuarios gestionar y monitorizar su red SDN a través de un navegador web.
- **odl-dlux-node:** permite a los usuarios visualizar información detallada sobre los nodos en su red SDN, incluyendo información sobre los puertos, los flujos de tráfico, los controladores asociados, etc.
- **odl-dlux-yangui:** proporciona una interfaz de usuario web que permite a los usuarios editar y validar archivos YANG a través de un navegador web.
- **odl-dlux-yangvisualizer:** permite a los usuarios visualizar los modelos de datos YANG de manera gráfica, lo que facilita la comprensión y la exploración de la

estructurade los modelos de datos.

B.4. Conexión OpenDaylight - Open Flow Manager

El proceso para configurar OFM implica la instalación de algunas herramientas específicas en el sistema operativo Linux. Para comenzar, se instala `npm`, una herramienta que proporcionaun entorno de ejecución para JavaScript, un lenguaje esencial para el proyecto OFM. Además, es necesario instalar `curl`, una herramienta enfocada en la transferencia de datos.

```
1    sudo apt install -y
2    npmsudo apt install
```

Extracto de código B.18: Comandos instalación herramientas npm y curl.

Para instalar OFM, es necesario contar con la herramienta Node.js, que proporciona el entorno de ejecución para los scripts de OFM escritos en JavaScript. Para obtener Node.js, se utilizael comando `curl` para descargar el código fuente e instalarlo. Esto se logra ingresando las siguientes líneas en la terminal:

```
1    curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
2    sudo apt-get install -y nodejs-legacy
```

Extracto de código B.19: Comandos instalación NodeJs.

Después de instalar las dependencias requeridas para ejecutar OFM, se procede a obtener el software desde su repositorio correspondiente. Para ello, se utiliza el comando `git` para clonar el proyecto y crear una copia local del mismo. Esto se realiza ingresando el siguiente comando en la terminal:

```
1    git clone https://github.com/CiscoDevNet/OpenDaylight-Openflow-
     App
```

Extracto de código B.20: Enlace repositorio OFM.

Una vez clonado el proyecto y accediendo al directorio descargado, se realiza una modificación en uno de los scripts generados, ubicado en la ruta `/ofm/src/common/config/`. En concreto, se necesita reemplazar el valor del campo correspondiente a la dirección IP a la cualOFM apuntará. Esta modificación se realiza en la línea 7 del Extracto de código B.21.

La interacción entre OpenDaylight y OpenFlow Manager resulta fundamental para el correcto desempeño de una red definida por software. OpenDaylight ejerce como el controlador centralizado de SDN y asume la responsabilidad de supervisar y coordinar la red. Por su parte, OpenFlow Manager constituye un elemento interno de OpenDaylight que interactúa de manera directa con los dispositivos de red que son compatibles con el protocolo OpenFlow.

En síntesis, la relación entre OpenDaylight y OpenFlow Manager posibilita que OpenDaylight tenga un control efectivo y eficiente sobre los dispositivos de red que son compatibles con OpenFlow. Esto contribuye a otorgar una mayor versatilidad y agilidad en la administración de redes en el contexto de un entorno de red definida por software.

A continuación, se detalla el archivo de configuración final de esta conexión:

```
define(['angularAMD'], function(ng)
{
     'use strict';


    var config = angular.module('config', [])
        .constant('ENV', {
              baseURL:
              "http://IP_CONTROLADOR_ODL:",
              adSalPort: "8181",
              mdSalPort : "8181",
              ofmPort :  "8181",
              configEnv : "ENV_DEV",
              odlUserName:
              'odlUserName',
              odlUserPassword:
              'odlUserPassword',getBaseURL :
              function(salType){
                  if(salType!==undefined){
```

```
                        var urlPrefix = "";
                        if(this.configEnv==="ENV_DEV")
                        {
                            urlPrefix = this.baseURL;
                        }else{
                            urlPrefix = window.location.protocol
                                +"//"+window.location.hostname+":";
                        }

                        if(salType==="AD_SAL"){
                            return urlPrefix + this.adSalPort;
                        }else if(salType==="MD_SAL"){
                            return  urlPrefix + this.mdSalPort;
                        }else if(salType==="CONTROLLER"){
                            return  urlPrefix + this.ofmPort;
                        }
                    }
                    return "";
                }
            });
        return config;
    });
```

Extracto de código B.21: Archivo de configuración OFM.

Para finalizar la configuración de OFM, se procede a instalar el cliente de la herramienta `Grunt`. Grunt es una herramienta que automatiza diversas tareas relacionadas con JavaScript. Parainstalarla, simplemente se debe ejecutar el siguiente comando en la terminal:

```
    sudo npm install -g grunt-cli
```

Extracto de código B.22: Instalación del cliente grunt.

C. Código en python del servidor NAC

C.1. Obtención de topología SDN

```
# -*- coding: utf-8 -*-
"""
Autores: [Erick Pérez P. - Mateo Molina Y.]
Fecha: [14/06/2023]
Descripcion: [El presente codigo ha sido desarrollado y compilado con
    la version 3.8 de python. El codigo tiene como objetivo la obtención
     de la topologia del controlador SDN, a traves de consultas HTTP
     GET al controlador OpenDaylight y a partir de los resultados
     obtenidos trabajar con archivos tipo JSON de los cuales empleando
     bucles se extrae la información necesaria para elaborar dos tablas,
     la primera haciendo referencia a la topologia conformada por los
     switches OvS y mientras que la segunda hace referencia a la
     topologia conformadapor hosts. Se hace uso funciones que ayudan a
     la correcta impresionde los datos en la tablas.]

Licencia: [Este código es de libre distribucion con fines académicos,
           investigativos.]
"""

import requests
import re
from requests.auth import HTTPBasicAuth

#Funcion que detecta si es un switch OpenFlow
def es_switch(palabra):
    return bool(re.match("^o", palabra))
#Funcion que detecta si es un host
def es_host(host_name):
    return bool(re.match("^h", host_name))

#funcion principal para la obtencion de la topología
def muestra_topo(controller_ip,username,password):
```

```
    # Direccion URL para solicitud HTTP GET de la topologia a ODL
    url = f'http://{controller_ip}:8181/restconf/operational/network-
        topology:network-topology/topology/flow:1/'
# Solicitud HTTP GET para obtencion de informacion de la topologia
    respuesta_http = requests.get(url, auth=HTTPBasicAuth(username,
        password))
    puertos_topo_sw=[] # puertos OvS para topologia de switches
    nodos_topo_sw=[] # nombres switches OvS para toopologia de
switches
    puertos_topo_host=[] # puertos OvS para topologia de hosts
nodos_topo_host=[] # nombres switches OvS para toopologia de hosts
    host_topo_host=[] # Direcciones MAC de los hosts detectados en la
        topologia
    data = respuesta_http.json() # Respuesta de ODL con informacion
        deacerca de la topologia
    ban=1 # Bandera para la impresion de resultados
    # Otencion de la topología de los Switches OvS en base a la
        respuesta HTTP
    # mediante bucles
    for sw in data['topology'][0]['link']:
        source_nodeSW = sw['source']['source-node'] # Nombre switch
            origen
        source_tpSW = sw['source']['source-tp'] # Nombre del puerto
            deswith origen
        dest_nodeSW = sw['destination']['dest-node'] # Nombre switch
            destino
        dest_tpSW = sw['destination']['dest-tp']#Nombre del puerto de
             swith origen
        # Impresion en consola de la topologia obtenida
        if ban==1:
            print("\n|=============================================|")
            print("|=============== TOPOLOGIA SW ================|")
            print("|=============================================|")
            print("| Switch Org  |  Puerto Org | Conectado a| Switch
                Dst |  Puerto Dst  |")
            print("|.............|")
```

```
        # Uso de funciones de ayuda para la correcta impresion de
            resultados
        if es_switch(source_nodeSW) and es_switch(source_tpSW) and
            es_switch(dest_tpSW) and es_switch(dest_nodeSW):
            puertos_topo_sw.append(source_tpSW)
            nodos_topo_sw.append(source_nodeSW)
            print(f'|{source_nodeSW} | {source_tpSW} | {dest_nodeSW}
|
                {dest_tpSW} |')
        ban=ban+1
  print("|================================================|\n")
   ban=1
# Otencion de la topología de los Hosts en base a la respuesta HTTP
   # mediante bucles
   for host in data['topology'][0]['link']:
        source_node = host['source']['source-node'] # Nombre del
             switchdonde esta conectado cada host
        source_tpHost = host['source']['source-tp'] # Nombre del
             puertode switch origen donde esta conectado cada host
        dest_nodeHost = host['destination']['dest-node'] # Dirección
            MAC host
        if ban==1:
            print("\n|=========================================|")
            print("|============ TOPOLOGIA HOSTS ==============|")
            print("\n|=========================================|")
            print("| Switch Org  |  Puerto Org |Conectado a|Hosts|")
            print("|.....|")
        # Uso de funciones de ayuda para la correcta impresion de
            resultados
        if es_switch(source_node) and es_switch(source_tpHost) and
            es_host(dest_nodeHost):
            source_node=host['source']['source-node']
            source_tpHost = host['source']['source-tp']
            dest_nodeHost = host['destination']['dest-node']
            print(f'|{source_node}|{source_tpHost}| dest_nodeHost}|')
```

```
75              # Se agregan en listas el nombre y puerto de los
                    switchesjunto
76              # con las direcciones MAC de los hosts detectados
77              nodos_topo_host.append(source_node)
78              puertos_topo_host.append(source_tpHost)
79              host topo host.append(dest nodeHost)
80
81          ban=ban+1
82      print("|================================================|\n"
83      )
84      #retornan las listas con el nombre y puerto de los switches
85      junto# con las direcciones MAC de los hosts detectados
```

Extracto de código C.1: Función para la obtención de la topologia de ODL.

C.2. Funciones complementarias

```
1  # -*- coding: utf-8 -*-
2  """
3  Autores: [Erick Pérez P. - Mateo Molina Y.]
4  Fecha: [14/06/2023]
5  Descripcion: [El presente codigo ha sido desarrollado y compilado con
       la version 3.8 de python. El codigo tiene como objetivo la
       implementacion de funciones de ayuda para que el servidor NAC
       funcione de manera adecuada. Las funciones son: ordena_macs,
       buscar_usuarios, asigna_departamento, asigna_vlan, crea_matriz,
       put_vlan.]
6
7  Licencia: [Este código es de libre distribucion para fines academicos
8              investigativos.]
9
10 """
11 import requests
12 from requests.auth import HTTPBasicAuth
13
14 # Ordena las direcciones MAC detectadas en el topologia
```

```
def ordena_macs(mac):
    macs = []
    for i in range(len(mac)):
        macs.append(mac[i][5:])
    return macs

# Valida la existencia de las direcciones MAC detectadas. Tienen como entrada las MACs ordenadas y devuelve un vector con los usuarios con en el mismo orden que las direcciones MAC
def buscar_usuarios(macs):
    with open("macsDB.txt", "r") as file:
        usuarios_macs = [linea.strip().split() for linea in file]
    resultados = []

    for mac in macs:
        for usuario, mac_usuario in usuarios_macs:
            if mac == mac_usuario:
                resultados.append(usuario)
    return resultados

# Valida los departamentos a los que pertenece cada usuario. Tiene como entrada la lista de usuarios y devuelve un vector con los departamnetos a los que que pertenece cada usuario, en el mismo orden que los usuarios
def asigna_departamento(user_list):
    with open("departamentos.txt") as file:
        db = [line.strip().split() for line in file]

    result = []
    for user in user_list:
        department = None
        for entry in db:
            if user == entry[1]:
                department = entry[0]
                break
        result.append(department)
```

```
46      return result
47
48  # Valida las VLANs a los que pertenece cada departamento. Tiene como
        entrada la lista de departamentos y devuelve un vector con las
        VLANs a las que que pertenece cada departamento, en el mismo orden
        que los departamentos
49  def asigna_vlan(dep_list):
50      with open("grupo_vlan.txt") as file:
51          db = [line.strip().split() for line in file]
52
53      result = []
54      for dep in dep_list:
55          vlan = None
56          for entry in db:
57              if dep == entry[1]:
58                  vlan = entry[0]
59                  break
60          result.append(vlan)
61      return result
62
63  # Crea la matriz NAC con toda la informacion validada
64  def crea_matriz(nodos,puertos,macs,user_per_mac,dep_per_user,
        vlan_per_dep):
65      matriz=[]
66      for i in range(len(nodos)):
67          matriz.append([nodos[i] , puertos[i], macs[i], user_per_mac[i],
68                         dep_per_user[i], vlan_per_dep[i]])
69      print("\n
            |============================================================|")
70      print("|================= RESUMEN DE TOPOLOGIA ================|")
71      for i in range(len(matriz)):
72          print(matriz[i])
73      print("|============================================================|\
            n")
74      return matriz
75
```

```
76 # Asigna las VLANs a los usuarios de la matriz NAC mediante el comando
       PUT y con la publicacion de un archivo tipo JSON con el protocolo
       802.1q
77 def put_vlan(controller_ip, username, password, node_id, port_number1,
       mac,
78               usuario, departamento, vlan_id):
79     id_flow1_port1=usuario+'-'+departamento+'-'+vlan_id
80     url1 = f'http://{controller_ip}:8181/restconf/config/opendaylight-
           inventory:nodes/node/{node_id}/table/0/flow/{id_flow1_port1}'
81     flowp1={
82         "flow": [
83                 {
84                     "table_id": 0,
85                     "id": id_flow1_port1,
86                     "priority": 1145,
87                     "hard-timeout": 0,
88                     "idle-timeout": 0,
89                     "match": {
90                         "in-port": port_number1,
91                         "vlan-match": {
92                             "vlan-id": {
93                                 "vlan-id": vlan_id,
94                                 "vlan-id-present": "true"
95                             }
96                         }
97                     },
98                     "instructions": {
99                         "instruction": [
100                             {
101                                 "order": 0,
102                                 "apply-actions": {
103                                     "action": [
104                                         {
105                                             "order": 0,
106                                             "push-vlan-action": {
107                                                 "ethernet-type": 33024,
```

```
                                                "tag": vlan_id,
                                                "pcp": 7,
                                                "cfi": 0,
                                                "vlan-id": vlan_id
                                            }
                                        }
                                    ]
                                }
                            }
                        ]
                    }
                }
            ]

    }


    response_vlanp1 = requests.put(url1, json=flowp1, auth=
        HTTPBasicAuth(username, password))


    if response_vlanp1.status_code == 200:
        print(f'VLAN {id_flow1_port1} agregada con éxito')

    else:
        print(f'Error al agregar VLAN {id_flow1_port1}')
```

Extracto de código C.2: Función de ayuda de servidor NAC (helpers.py).

C.3. Función principal

```
# -*- coding: utf-8 -
*-"""
Autores: [Erick Pérez P. - Mateo Molina Y.]
Fecha: [14/06/2023]
Descripcion: [El presente codigo ha sido desarrollado y compilado
```

```
    la version 3.8 de python. El codigo tiene como objetivo la
    implementacion de un servidor de acceso a la red, NAC, a través
    de consultas GET al controlador OpenDaylight y a partir de los
    resultados obtenidos trabajar con archivos tipo JSON. Se hace uso
    funciones de ayuda que realizan los procesos de obtencion del
    estado de la topologia y valicacion de usuarios basados en
    archivos de texto con informacion de direcciones MAC, usuarios,
    departamentos yVLANs.]

Licencia: [Este código es de libre distribucion para fines academicos e
           investigativos.]

"""
import topologia
import helpers
import time

ip_controlador = '192.168.18.21'     # Direccion IP del controlador ODL
username = 'admin'                   # Usuario del controlador ODL
password = 'admin'                   # Contraseña del controlador ODL
matriz_aux = []                      # Matriz NAC

while True:
    #================================================================
    #================================================================
    #recolecta la informacion actual de la topologia
    puertos,nodos,mac = topologia.muestra_topo(ip_controlador,username,
        password)

    # ordena las macs detectadas de la topologia en una matriz
    macs = helpers.ordena_macs(mac)

    # compara macs de hosts contectados con la DB y los relaciona con
        su user
    user_per_mac = helpers.buscar_usuarios(macs)

```

```
33      # compara users detectados con la DB y los relaciona con su
            departamento
34      dep_per_user = helpers.asigna_departamento(user_per_mac)
35
36      # compara departamentos de users con la DB y los relaciona con su
            VLAN
37      vlan_per_dep = helpers.asigna_vlan(dep_per_user)
38
39      # crea una matriz con toda la informacion recolectada de la
            topologia y DB
40      matriz = helpers.crea_matriz(nodos, puertos, macs,user_per_mac,
41                                   dep_per_user, vlan_per_dep)
42
43      # compara cambios en la topologia creada
44      if matriz_aux != matriz:
45          # asigna VLANs dinamicamente con la informacion de la matriz
46
47          for i in range(len(matriz)):
48              helpers.put_vlan(ip_controlador, username, password,
                    matriz[i][0],
49                               matriz[i][1], matriz[i][2], matriz[i
                                     ][3],
50                               matriz[i][4], matriz[i][5])
51      else:
52          print("NO EXISTEN CAMBIOS EN LA TOPOLOGIA")
53
54      matriz_aux = matriz
55      #================================================================
56      #================================================================
57      time.sleep(20)
```

Extracto de código C.3: Función principal de servidor NAC (main.py).

D. Configuración de red en usuarios

A continuación, desde la Figura D.1 hasta la Figura D.8 se ilustra las configuraciones de red delas interfaces de los hosts conectadas a los conmutadores OvS. También se puede verificar las direcciones MAC de cada uno de los hosts, mismas que coinciden con lo indicado en la Figura 4.17.

```
erick@erick:~$ ifconfig enp3s0
enp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.30  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::212a:297f:f13c:61d1  prefixlen 64  scopeid 0x20<link>
        ether c8:d3:ff:75:53:8d  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 4543 (4.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 83  bytes 9107 (9.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

erick@erick:~$
```

Figura D.1: Configuración de red en user1.

```
hp@erickUbuntu:~$ ifconfig eno1
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.31  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::aa6e:979c:699:49ca  prefixlen 64  scopeid 0x20<link>
        ether 10:e7:c6:dd:4f:07  txqueuelen 1000  (Ethernet)
        RX packets 296  bytes 20512 (20.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 160  bytes 29412 (29.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

hp@erickUbuntu:~$
```

Figura D.2: Configuración de red en user2.

```
Adaptador de Ethernet Conexión de área local:

   Sufijo DNS específico para la conexión. . : example.org
   Descripción . . . . . . . . . . . . . . . : JMicron PCI Express Fast Ethernet Adapter
   Dirección física. . . . . . . . . . . . . : 00-90-F5-B6-BE-37
   DHCP habilitado . . . . . . . . . . . . . : sí
   Configuración automática habilitada . . . : sí
   Vínculo: dirección IPv6 local. . . : fe80::5024:c884:a4e7:fe07%7(Preferido)
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.2.30(Preferido)
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Concesión obtenida. . . . . . . . . . . . : domingo, 21 de mayo de 2023 11:15:59
   La concesión expira . . . . . . . . . . . : domingo, 21 de mayo de 2023 11:25:57
   Puerta de enlace predeterminada . . . . . : 192.168.2.1
   Servidor DHCP . . . . . . . . . . . . . . : 192.168.2.21
   IAID DHCPv6 . . . . . . . . . . . . . . . : 234918133
   DUID de cliente DHCPv6. . . . . . . . . . : 00-01-00-01-19-F8-63-19-00-90-F5-B6-BE-37
   Servidores DNS. . . . . . . . . . . . . . : 8.8.8.8
   NetBIOS sobre TCP/IP. . . . . . . . . . . : habilitado
```

Figura D.3: Configuración de red en user3.

```
Adaptador de Ethernet Conexión de área local:

   Sufijo DNS específico para la conexión. . :
   Descripción . . . . . . . . . . . . . . . : Controladora Gigabit Ethernet Atheros AR8131 PCI-E (NDIS 6.20)
   Dirección física. . . . . . . . . . . . . : F0-BF-97-8F-B9-8A
   DHCP habilitado . . . . . . . . . . . . . : no
   Configuración automática habilitada . . . : sí
   Vínculo: dirección IPv6 local. . . : fe80::800:b1ff:4778:fc57%11(Preferido)
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.2.31(Preferido)
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . :
   IAID DHCPv6 . . . . . . . . . . . . . . . : 300990359
   DUID de cliente DHCPv6. . . . . . . . . . : 00-01-00-01-1F-1C-03-9E-F0-BF-97-8F-B9-8A
   Servidores DNS. . . . . . . . . . . . . . : fec0:0:0:ffff::1%1
                                               fec0:0:0:ffff::2%1
                                               fec0:0:0:ffff::3%1
   NetBIOS sobre TCP/IP. . . . . . . . . . . : habilitado
```

Figura D.4: Configuración de red en user4.

```
Adaptador de Ethernet Ethernet:

   Sufijo DNS específico para la conexión. . :
   Descripción . . . . . . . . . . . . . . . : Realtek PCIe GbE Family Controller
   Dirección física. . . . . . . . . . . . . : 00-2B-67-DF-A3-D8
   DHCP habilitado . . . . . . . . . . . . . : no
   Configuración automática habilitada . . . : sí
   Vínculo: dirección IPv6 local. . . : fe80::31e8:e60d:603a:89bb%24(Preferido)
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.3.33(Preferido)
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.3.254
   IAID DHCPv6 . . . . . . . . . . . . . . . : 100674407
   DUID de cliente DHCPv6. . . . . . . . . . : 00-01-00-01-26-AD-09-15-00-2B-67-DF-A3-D8
   Servidores DNS. . . . . . . . . . . . . . : 8.8.8.8
   NetBIOS sobre TCP/IP. . . . . . . . . . . : habilitado
```

Figura D.5: Configuración de red en user5.

```
acer@acer-pc:~$ ifconfig enp1s0
enp1s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.34  netmask 255.255.255.0  broadcast 192.168.3.255
        ether e8:9a:8f:e2:95:f9  txqueuelen 1000  (Ethernet)
        RX packets 602  bytes 60875 (60.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 129  bytes 22770 (22.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

acer@acer-pc:~$
```

Figura D.6: Configuración de red en user6.

```
Ethernet adapter Ethernet:

   Connection-specific DNS Suffix  . : example.org
   Description . . . . . . . . . . . : Killer E2600 Gigabit Ethernet Controller
   Physical Address. . . . . . . . . : 60-18-95-30-09-CD
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::5c46:a2b4:582a:a633%6(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.4.31(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Sunday, May 21, 2023 12:00:15 PM
   Lease Expires . . . . . . . . . . : Sunday, May 21, 2023 12:10:15 PM
   Default Gateway . . . . . . . . . : 192.168.4.1
   DHCP Server . . . . . . . . . . . : 192.168.4.42
   DHCPv6 IAID . . . . . . . . . . . : 106961045
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-28-57-3D-F6-60-18-95-30-09-CD
   DNS Servers . . . . . . . . . . . : 8.8.8.8
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Figura D.7: Configuración de red en user7.

```
Adaptador de Ethernet Ethernet:

   Sufijo DNS específico para la conexión. . : example.org
   Descripción . . . . . . . . . . . . . . . : Realtek PCIe GbE Family Controller
   Dirección física. . . . . . . . . . . . . : 28-D2-44-FF-54-C2
   DHCP habilitado . . . . . . . . . . . . . : sí
   Configuración automática habilitada . . . : sí
   Vínculo: dirección IPv6 local. . . : fe80::fc3a:9f30:5b06:94c6%9(Preferido)
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.4.30(Preferido)
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Concesión obtenida. . . . . . . . . . . . : domingo, 21 de mayo de 2023 16:59:57
   La concesión expira . . . . . . . . . . . : domingo, 21 de mayo de 2023 17:09:57
   Puerta de enlace predeterminada . . . . . : 192.168.4.1
   Servidor DHCP . . . . . . . . . . . . . . : 192.168.4.41
   IAID DHCPv6 . . . . . . . . . . . . . . . : 136892996
   DUID de cliente DHCPv6. . . . . . . . . . : 00-01-00-01-2A-AE-C7-FA-28-D2-44-FF-54-C2
   Servidores DNS. . . . . . . . . . . . . . : 8.8.8.8
   NetBIOS sobre TCP/IP. . . . . . . . . . . : habilitado

C:\Users\Mateo>
```

Figura D.8: Configuración de red en user8.

E. Pruebas de Funcionamiento

E.1. VLANs sobre red sin SDN

A continuación, en las Figuras E.1, E.2, E.3 y E.4 se ilustran las configuraciones de los conmutadores de red, tarjetas RPi con OvS. Donde se observa la asignación de VLANs en los puertos respectivos.

```
pi@raspberrypi:~/openvswitch-2.17.5 $ sudo ovs-vsctl show
8fe35be0-4348-45bc-9c46-8e5b8c8dca40
    Bridge br0
        Port br0
            Interface br0
                type: internal
        Port eth0
            tag: 100
            Interface eth0
        Port eth2
            tag: 100
            Interface eth2
        Port eth1
            tag: 100
            Interface eth1
pi@raspberrypi:~/openvswitch-2.17.5 $
```

Figura E.1: Configuración OvS sobre RPi 1.

```
pi@raspberrypi:~/openvswitch-2.17.5 $ sudo ovs-vsctl show
82bf1eea-7451-42a0-add7-1510638dad75
    Bridge br0
        Port br0
            Interface br0
                type: internal
        Port eth2
            tag: 200
            Interface eth2
        Port eth0
            tag: 200
            Interface eth0
        Port eth1
            tag: 200
            Interface eth1
pi@raspberrypi:~/openvswitch-2.17.5 $
```

Figura E.2: Configuración OvS sobre RPi 2.

```
pi@raspberrypi:~/openvswitch-2.17.6 $ sudo ovs-vsctl show
b08fdb85-18c4-4042-b77f-e5f9555ad4a2
    Bridge br0
        Port br0
            Interface br0
                type: internal
        Port eth2
            tag: 300
            Interface eth2
        Port eth0
            tag: 300
            Interface eth0
        Port eth1
            tag: 300
            Interface eth1
pi@raspberrypi:~/openvswitch-2.17.6 $
```

Figura E.3: Configuración OvS sobre RPi 3.

```
pi@raspberrypi:~/openvswitch-2.17.5 $ sudo ovs-vsctl show
16f7cbe6-0243-4ca1-9dba-4df05f9c8480
    Bridge br0
        Port eth2
            tag: 400
            Interface eth2
        Port br0
            Interface br0
                type: internal
        Port eth1
            tag: 400
            Interface eth1
        Port eth0
            tag: 400
            Interface eth0
pi@raspberrypi:~/openvswitch-2.17.5 $
```

Figura E.4: Configuración OvS sobre RPi 4.

```
erick@erick:~$ ping -c 4 192.168.1.31
PING 192.168.1.31 (192.168.1.31) 56(84) bytes of data.
64 bytes from 192.168.1.31: icmp_seq=1 ttl=64 time=4.04 ms
64 bytes from 192.168.1.31: icmp_seq=2 ttl=64 time=1.27 ms
64 bytes from 192.168.1.31: icmp_seq=3 ttl=64 time=1.27 ms
64 bytes from 192.168.1.31: icmp_seq=4 ttl=64 time=1.19 ms

--- 192.168.1.31 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 1.194/1.942/4.039/1.210 ms
erick@erick:~$
```

Figura E.5: Ping desde user 1 a user 2 en VLAN 100.

A continuación, en las Figuras E.7 y E.8 se ilustra la comunicación lograda entre 2 clientespertenecientes a la VLAN 400, con el uso de paquetes ICMP.

```
hp@erickUbuntu:~$ ping -c 4 192.168.1.30
PING 192.168.1.30 (192.168.1.30) 56(84) bytes of data.
64 bytes from 192.168.1.30: icmp_seq=1 ttl=64 time=1.04 ms
64 bytes from 192.168.1.30: icmp_seq=2 ttl=64 time=0.999 ms
64 bytes from 192.168.1.30: icmp_seq=3 ttl=64 time=1.70 ms
64 bytes from 192.168.1.30: icmp_seq=4 ttl=64 time=0.730 ms

--- 192.168.1.30 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 0.730/1.119/1.708/0.360 ms
hp@erickUbuntu:~$
```

Figura E.6: Ping desde user 2 a user 1 en VLAN 100.

```
C:\Users\Mateo>ping 192.168.4.30

Pinging 192.168.4.30 with 32 bytes of data:
Reply from 192.168.4.30: bytes=32 time=2ms TTL=128
Reply from 192.168.4.30: bytes=32 time=1ms TTL=128
Reply from 192.168.4.30: bytes=32 time=1ms TTL=128
Reply from 192.168.4.30: bytes=32 time=1ms TTL=128

Ping statistics for 192.168.4.30:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\Users\Mateo>
```

Figura E.7: Ping desde cliente 1 a cliente 2 en VLAN 400.

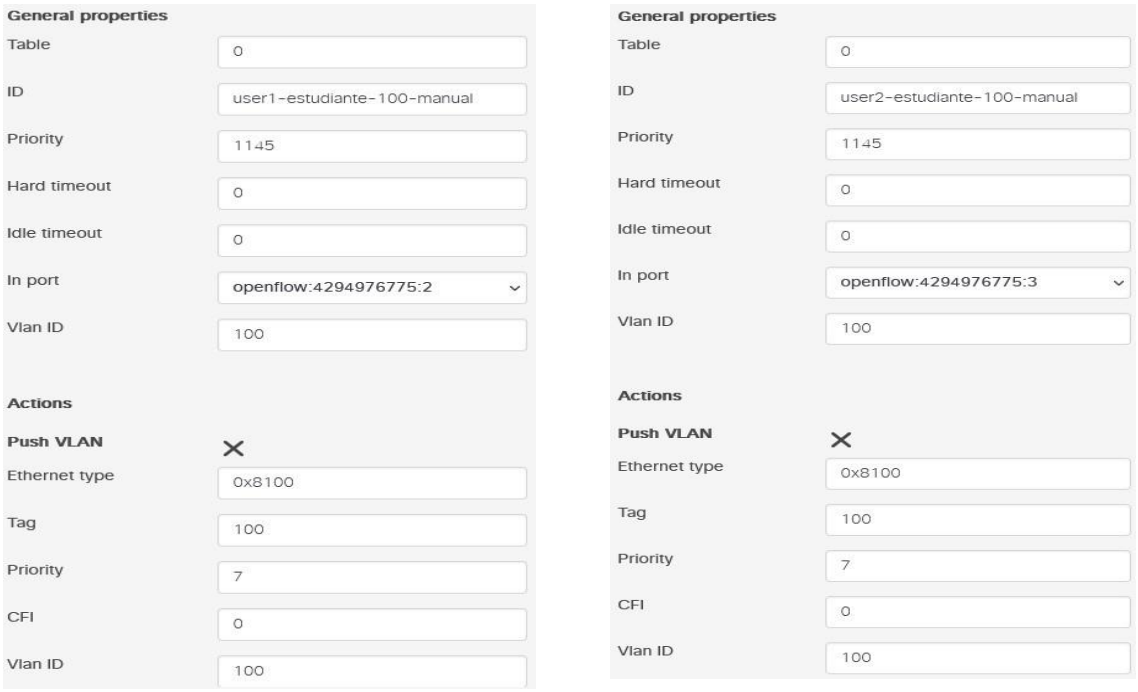
```
C:\Users\Mateo>ping 192.168.4.31

Haciendo ping a 192.168.4.31 con 32 bytes de datos:
Respuesta desde 192.168.4.31: bytes=32 tiempo=1ms TTL=128
Respuesta desde 192.168.4.31: bytes=32 tiempo=1ms TTL=128
Respuesta desde 192.168.4.31: bytes=32 tiempo=2ms TTL=128
Respuesta desde 192.168.4.31: bytes=32 tiempo=2ms TTL=128

Estadísticas de ping para 192.168.4.31:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 1ms, Máximo = 2ms, Media = 1ms

C:\Users\Mateo>
```

Figura E.8: Ping desde cliente 2 a cliente 1 en VLAN 400.

E.2. VLANs sobre red con SDN

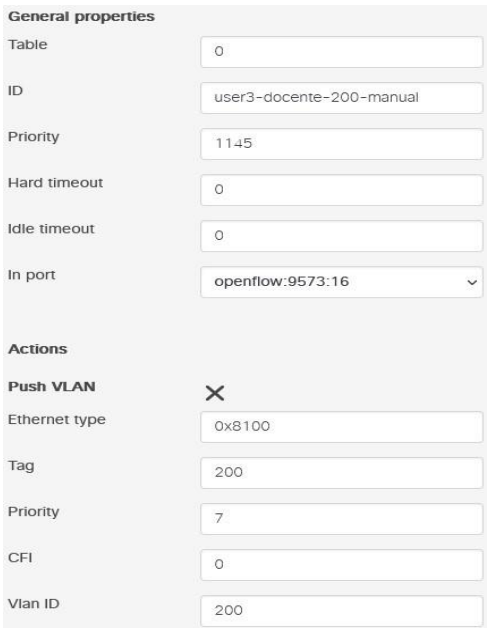
E.2.1. Asignación estática de VLANs

En este apartado se presenta los resultados obtenidos a partir de la asignación de VLANs empleando la herramienta OpenFlow Manager. Donde, luego de realizar la configuración y creación de flujos que contienen las reglas para la asignación de VLANs a los hosts (FigurasE.9(a) a E.12(b)).

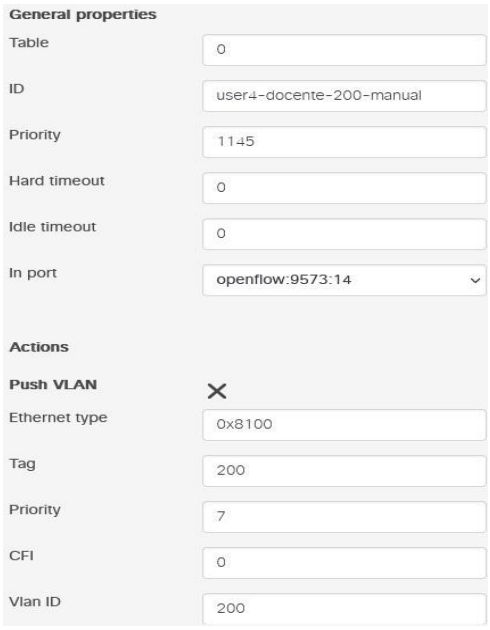


((a)) Puerto 2. ((b)) Puerto 3.

Figura E.9: Asignación estática de VLAN 100 sobre de RPi 1.

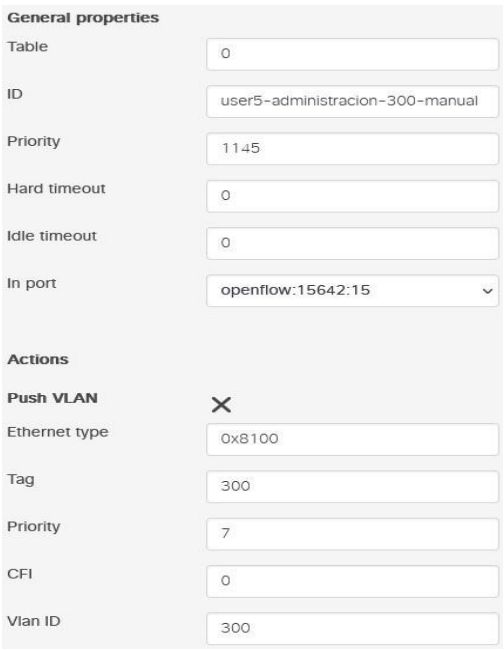


((a)) Puerto 2.

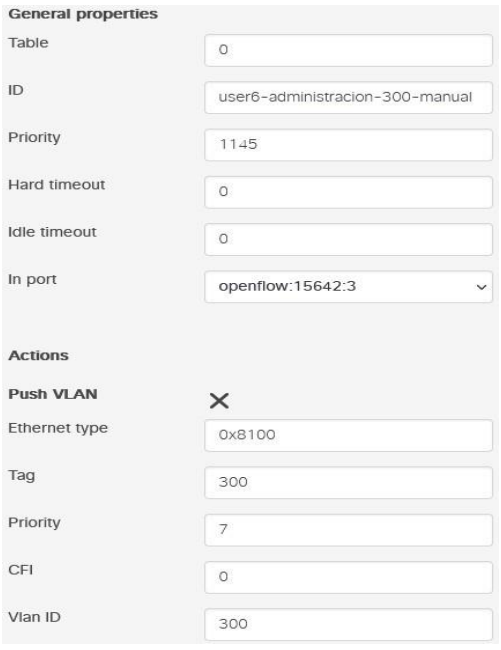


((b)) Puerto 3.

Figura E.10: Asignación estática de VLAN 200 sobre de RPi 2.

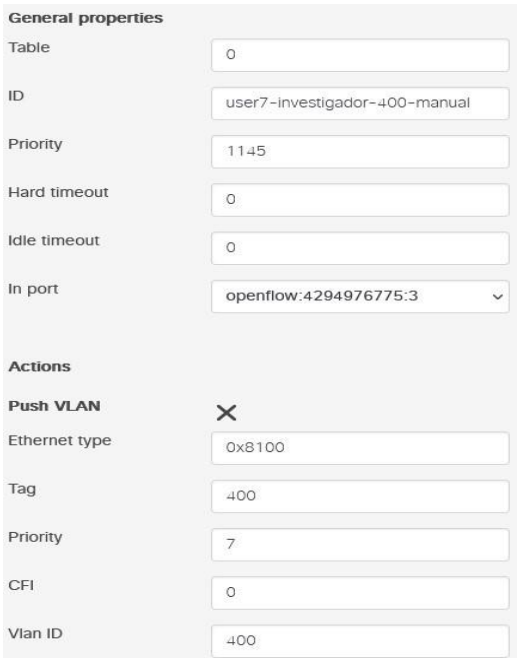


((a)) Puerto 2.

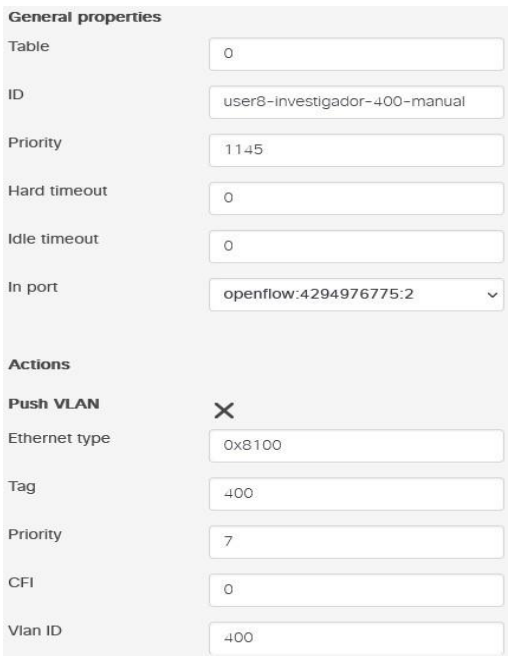


((b)) Puerto 3.

Figura E.11: Asignación estática de VLAN 300 sobre de RPi 3.



((a)) Puerto 2.



((b)) Puerto 3.

Figura E.12: Asignación estática de VLAN 400 sobre de RPi 4.

E.2.2. Asignación dinámica de VLANs

A continuación, en las Figuras E.13, E.14 y E.15 se ilustran los resultados obtenidos al correrel programa que crea y configura el servidor NAC. Se trata de mensajes en la terminal del IDESpyder3, mismos que se usan como mensajes log a lo largo del funcionamiento del servidor.

```
|============================================================|
|========================= TOPOLOGIA SW =====================|
|============================================================|
| Nodo Org  |  Puerto Org | Conectado a| Nodo Dst |  Puerto Dst  |
|------------------------------------------------------------|
|openflow:4294976775 | openflow:4294976775:1 | openflow:4294976906 | openflow:4294976906:3 |
|openflow:15642 | openflow:15642:3 | openflow:4294976906 | openflow:4294976906:5 |
|openflow:968934587319 | openflow:968934587319:1 | openflow:4294976906 | openflow:4294976906:4 |
|openflow:4294976906 | openflow:4294976906:2 | openflow:9573 | openflow:9573:13 |
|openflow:9573 | openflow:9573:13 | openflow:4294976906 | openflow:4294976906:2 |
|openflow:4294976906 | openflow:4294976906:5 | openflow:15642 | openflow:15642:3 |
|openflow:4294976906 | openflow:4294976906:3 | openflow:4294976775 | openflow:4294976775:1 |
|openflow:4294976906 | openflow:4294976906:4 | openflow:968934587319 | openflow:968934587319:1 |
|============================================================|
```

Figura E.13: Información de la topología de conmutadores.

```
|============================================================|
|====================== TOPOLOGIA HOSTS =====================|
|============================================================|
| Nodo Org  |  Puerto Org | Conectado a  |        Hosts       |
|------------------------------------------------------------|
|openflow:968934587319 | openflow:968934587319:3 | host:10:e7:c6:dd:4f:07 |
|openflow:15642 | openflow:15642:15 | host:00:2b:67:df:a3:d8 |
|openflow:968934587319 | openflow:968934587319:2 | host:c8:d3:ff:75:53:8d |
|openflow:9573 | openflow:9573:16 | host:f0:bf:97:8f:b9:8a |
|openflow:9573 | openflow:9573:14 | host:00:90:f5:b6:be:37 |
|openflow:15642 | openflow:15642:3 | host:b8:27:eb:da:28:db |
|openflow:4294976775 | openflow:4294976775:2 | host:28:d2:44:ff:54:c2 |
|openflow:4294976775 | openflow:4294976775:3 | host:60:18:95:30:09:cd |
|============================================================|
```

Figura E.14: Información de la topología de hosts.

```
|============================================================|
|=================== RESUMEN DE TOPOLOGIA ===================|
['openflow:968934587319', 'openflow:968934587319:3', '10:e7:c6:dd:4f:07', 'user2', 'estudiante', '100']
['openflow:15642', 'openflow:15642:15', '00:2b:67:df:a3:d8', 'user5', 'administracion', '300']
['openflow:968934587319', 'openflow:968934587319:2', 'c8:d3:ff:75:53:8d', 'user1', 'estudiante', '100']
['openflow:9573', 'openflow:9573:16', 'f0:bf:97:8f:b9:8a', 'user4', 'docente', '200']
['openflow:9573', 'openflow:9573:14', '00:90:f5:b6:be:37', 'user3', 'docente', '200']
['openflow:15642', 'openflow:15642:3', 'b8:27:eb:da:28:db', 'user6', 'administracion', '300']
['openflow:4294976775', 'openflow:4294976775:2', '28:d2:44:ff:54:c2', 'user8', 'investigador', '400']
['openflow:4294976775', 'openflow:4294976775:3', '60:18:95:30:09:cd', 'user7', 'investigador', '400']
|============================================================|
```

Figura E.15: Matriz NAC.

Además, en las Figura E.16 se ilustra los flujos agregados a la SDN observados desde OFM.Estos flujos hacen referencia a la asignación de VLANs desde el servidor NAC.

Flow name	ID	Table ID	Device	Device type	Device name	Operational	Actions
	user						
[id:user6-administracion-300, table:0]	user6-administracion-300	0	openflow:15642	Open vSwitch	None	ON DEVICE	
[id:user5-administracion-300, table:0]	user5-administracion-300	0	openflow:15642	Open vSwitch	None	ON DEVICE	
[id:user7-investigador-400, table:0]	user7-investigador-400	0	openflow:4294976775	Open vSwitch	None	ON DEVICE	
[id:user8-investigador-400, table:0]	user8-investigador-400	0	openflow:4294976775	Open vSwitch	None	ON DEVICE	
[id:user4-docente-200, table:0]	user4-docente-200	0	openflow:9573	Open vSwitch	None	ON DEVICE	
[id:user3-docente-200, table:0]	user3-docente-200	0	openflow:9573	Open vSwitch	None	ON DEVICE	

Figura E.16: Flujos en OFM.

F. Guía de procedimiento general para la actualización de asignación de VLANs de manera estática a dinámica en una red de campus universitaria con SDN.

Guía de procedimiento general para la actualización de asignación deVLANs de manera estática a dinámica en una red de campus universitaria con SDN.

Guía de procedimiento general para la actualizaciónde asignación de VLANs de manera estática a dinámica en una red de campus universitaria con SDN	Pag.: 1
	Año: 2023

INTRODUCCIÓN

El propósito de la elaboración de la presente guía de procedimientos es establecer un conjunto de pasos claros para el proceso de asignación de VLANs, realizado por el Departamento de Tecnologías y Comunicación en las redes de campus. Este documento ayudará a automatizar las configuraciones de red mediante la actualización de equipos de hardware y software en la red de campus.

La guía proporciona una referencia general y accesible que permite a los administradores de red comprender y seguir los procedimientos para asignar dinámicamente VLANs a usuarios conectados a red de campus de manera efectiva. Esto permite una mayor productividad y menor margen de error en las configuraciones realizadas por los administradores de red. Además, promueve la uniformidad en las prácticas de este tipo, lo que en un futuro puede facilitar la capacitación de nuevos usuarios y la colaboración entre diferentes equipos.

OBJETIVO DE LA GUÍA

Establecer acciones, actividades y procedimientos generales necesarios para la actualización del proceso de asignación de VLANs de manera dinámica en redes de campus universitarios con SDN.

ALCANCE DE LA GUÍA

La presente guía de procedimiento es de carácter académico que servirá como apoyo a los administradores de red que ejecuten funciones de automatización de procesos en entornos universitarios basados en SDN, específicamente en la asignación dinámica de VLANs.

RESPONSABLES

- Departamento de Tecnologías de la Información y Comunicación (Administradores de red).

MATERIALES, EQUIPOS Y SISTEMAS

- Adaptadores USB – Ethernet (de ser necesario).
- Cables Ethernet.
- Tarjetas Raspberry Pi / Conmutadores OpenFlow.
- Framework Open vSwitch (OvS).
- Controlador SDN OpenDaylight (ODL).
- Framework Open Flow Manager (OFM).
- Servidor de Control de Acceso a la Red (NAC).

Guía de procedimiento general para la actualizaciónde asignación de VLANs de manera estática a dinámica en una red de campus universitaria con SDN.	Pag.: 5
	Año: 2023
PROCEDIMIENTOS	
ASIGNACION ESTÁTICA DE VLANS SIN SDN	

Actores del proceso:

- Administrador de redes.

Herramientas:

- Conmutadores y/o tarjetas RPi (OvS).

No.	Responsable	Descripción de la actividad
1	Administrador de red	Configurar e iniciar el servidor DHCP en cada uno de los conmutadores y/o tarjetas RPi.
2	Administrador de red	Identificar al usuario conectado al conmutador y/o tarjeta RPi mediante la dirección MAC del equipo (en equipo de usuario).
3	Administrador de red	Asignar a cada usuario su grupo correspondiente (estudiante, docente, administrador, investigador, etc.) en base a los archivos locales.
4	Administrador de red	Ingresar los comandos vía terminal de los conmutadores para la asignación de VLANs a los diferentes usuarios.
5	Administrador de red	Verificar la configuración realizada en el conmutador mediante línea de comandos.
6	Administrador de red	Comprobar la comunicación de los equipos dentro de la VLAN correspondiente mediante un intercambio de paquetes ICMP (ping).

PROCEDIMIENTOS

ASIGNACION ESTÁTICA DE VLANS EMPLEANDO SDN

Actores del proceso:

- Administrador de redes.

Herramientas:

- Conmutadores y/o tarjetas RPi (OvS).
- Controlador SDN OpenDaylight (ODL).
- Framework Open Flow Manager (OFM).

Descripción de actividades:

No.	Responsable	Descripción de la actividad
1	Administrador de red	Iniciar el controlador ODL junto con el framework OFM.
2	Administrador de red	Configurar e iniciar el servidor DHCP en cada uno de los conmutadores y/o tarjetas RPi.
3	Administrador de red	Configurar e iniciar el protocolo OpenFlow en cada uno de los conmutadores y/o tarjetas RPi.
4	Administrador de red	Comprobar la construcción de la topologíadentro de la interfaz gráfica del controlador.
5	Administrador de red	Identificar a cada usuario conectado a la redmediante la dirección MAC del equipo, detectada por el controlador ODL (en controlador ODL).

6	Administrador de red	Asignar a cada usuario al grupo correspondiente (estudiante, docente, administrador, investigador, etc.) en base a los archivos locales. .
7	Administrador de red	Configurar y asignar los flujos dentro de OFM para asignar VLANs a cada usuario dependiendo al grupo al que pertenece.
8	Administrador de red	Comprobar la comunicación de los equipos dentro de la VLAN correspondiente mediante un intercambio de paquetes ICMP (ping).

PROCEDIMIENTOS

ASIGNACION DINÁMICA DE VLANS EMPLEANDO SDN

Actores del proceso:

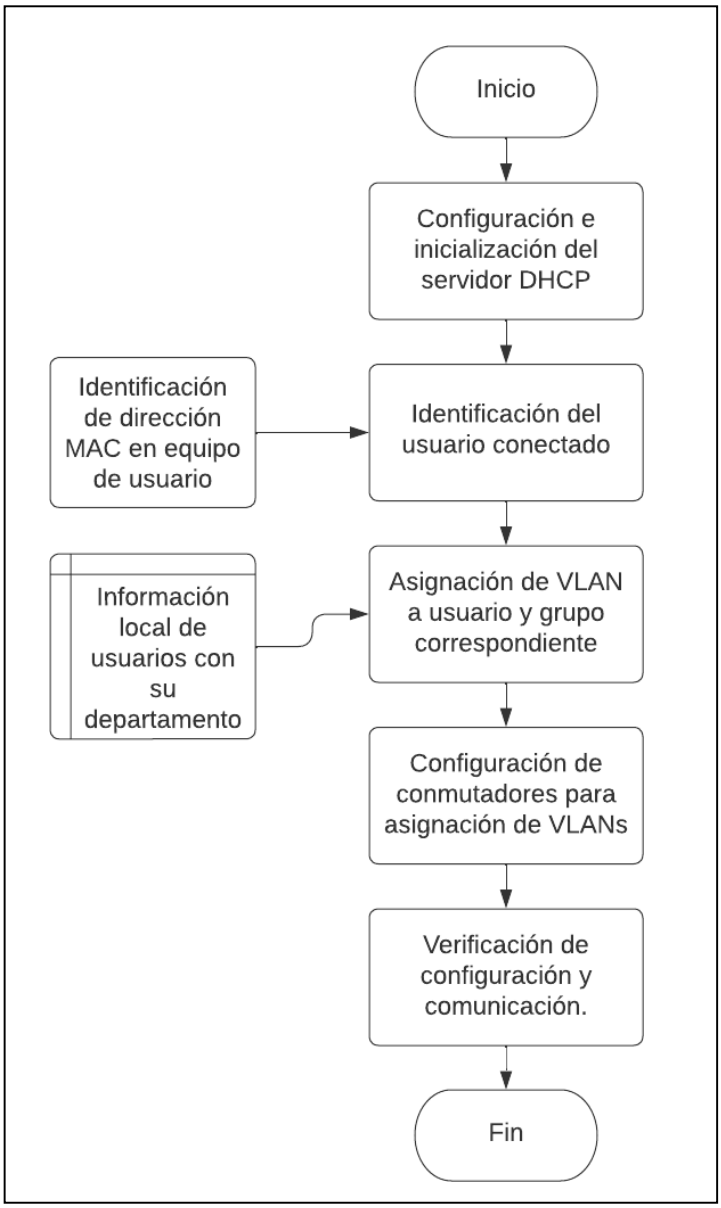
- Administrador de redes.

Herramientas:

- Conmutadores y/o tarjetas RPi (OvS).
- Controlador SDN OpenDaylight (ODL).
- Framework Open Flow Manager (OFM).
- Servidor de control de acceso a la red (NAC).

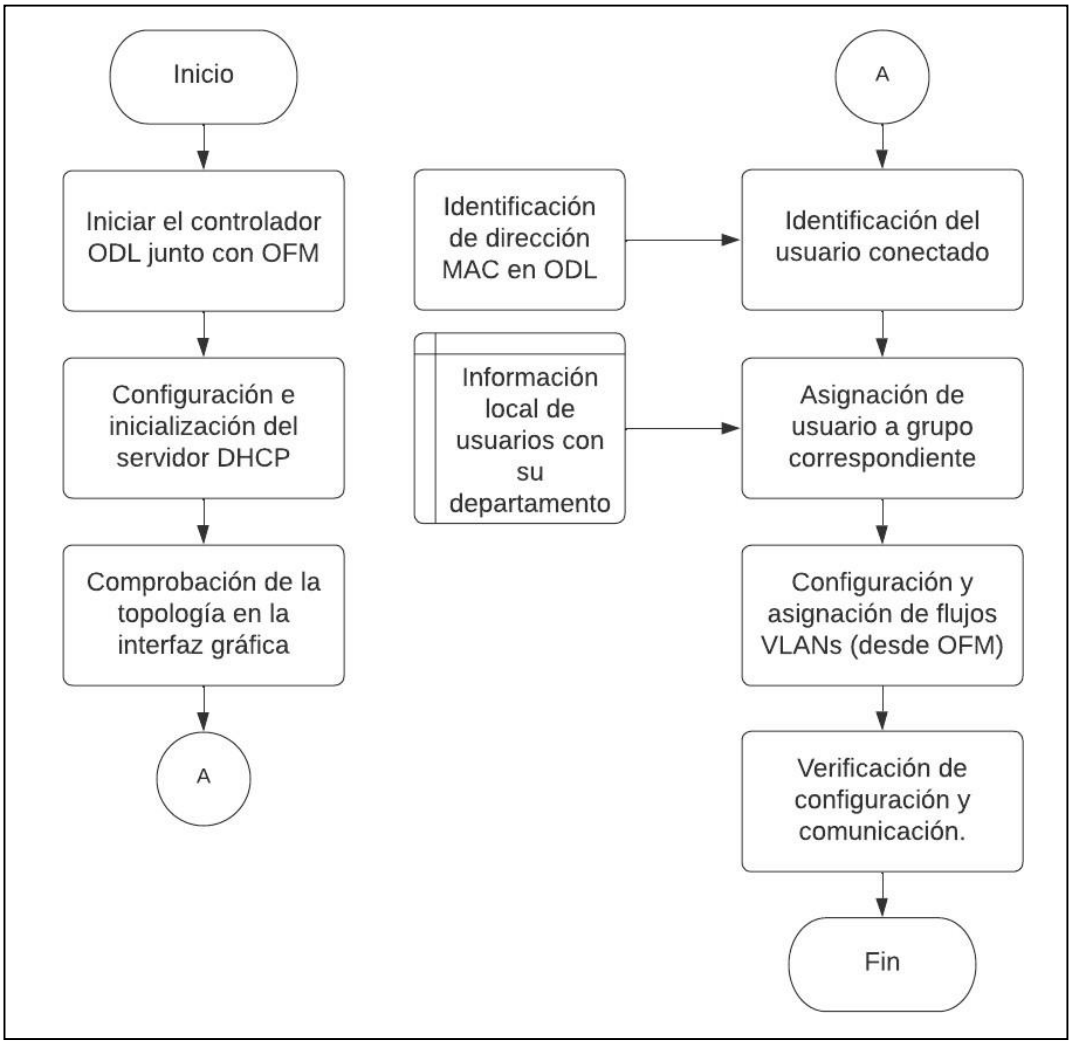
No.	Responsable	Descripción de la actividad
1	Administrador de red	Iniciar el controlador ODL junto con el framework OFM.
2	Administrador de red	Configurar e iniciar el servidor DHCP en cada uno de los conmutadores y/o tarjetas RPi.
3	Administrador de red	Configurar e iniciar el protocolo OpenFlow en cada uno de los conmutadores y/o tarjetas RPi.
4	Administrador de red	Inicio del servidor NAC
4	Servidor NAC	Asignación dinámica de VLANs a cada usuario de la red.
5	Administrador de red	Comprobar la comunicación de los equipos dentro de la VLAN correspondiente mediante un intercambio de paquetes ICMP (ping).

Guía de procedimiento general para la actualizaciónde asignación de VLANs de manera estática a dinámica en una red de campus universitaria con SDN.	Pag.: 9 Año: 2023
DIAGRAMAS DE FLUJO	
ASIGNACION ESTÁTICA DE VLANS SIN SDN	



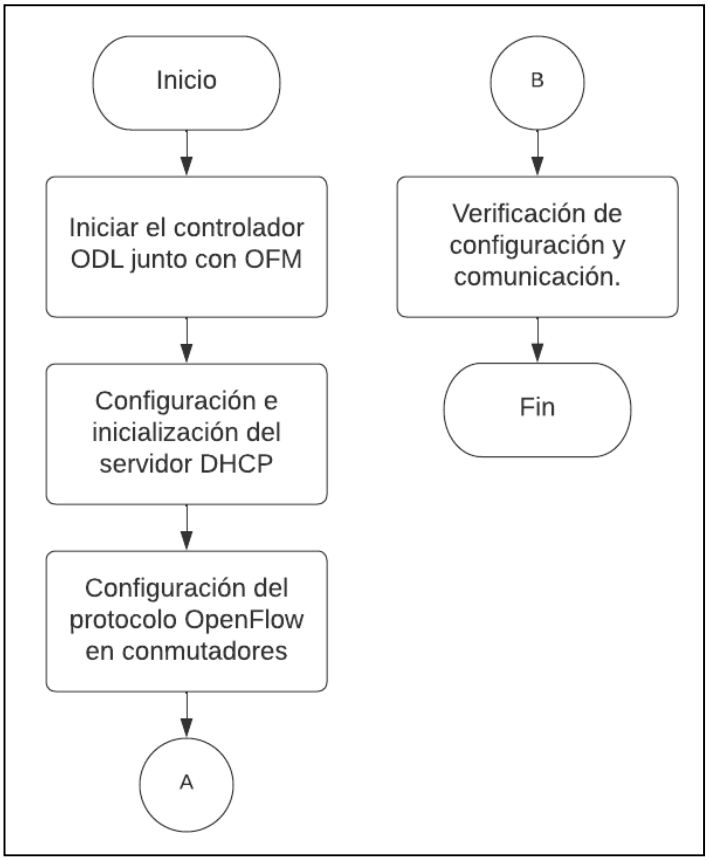
DIAGRAMAS DE FLUJO

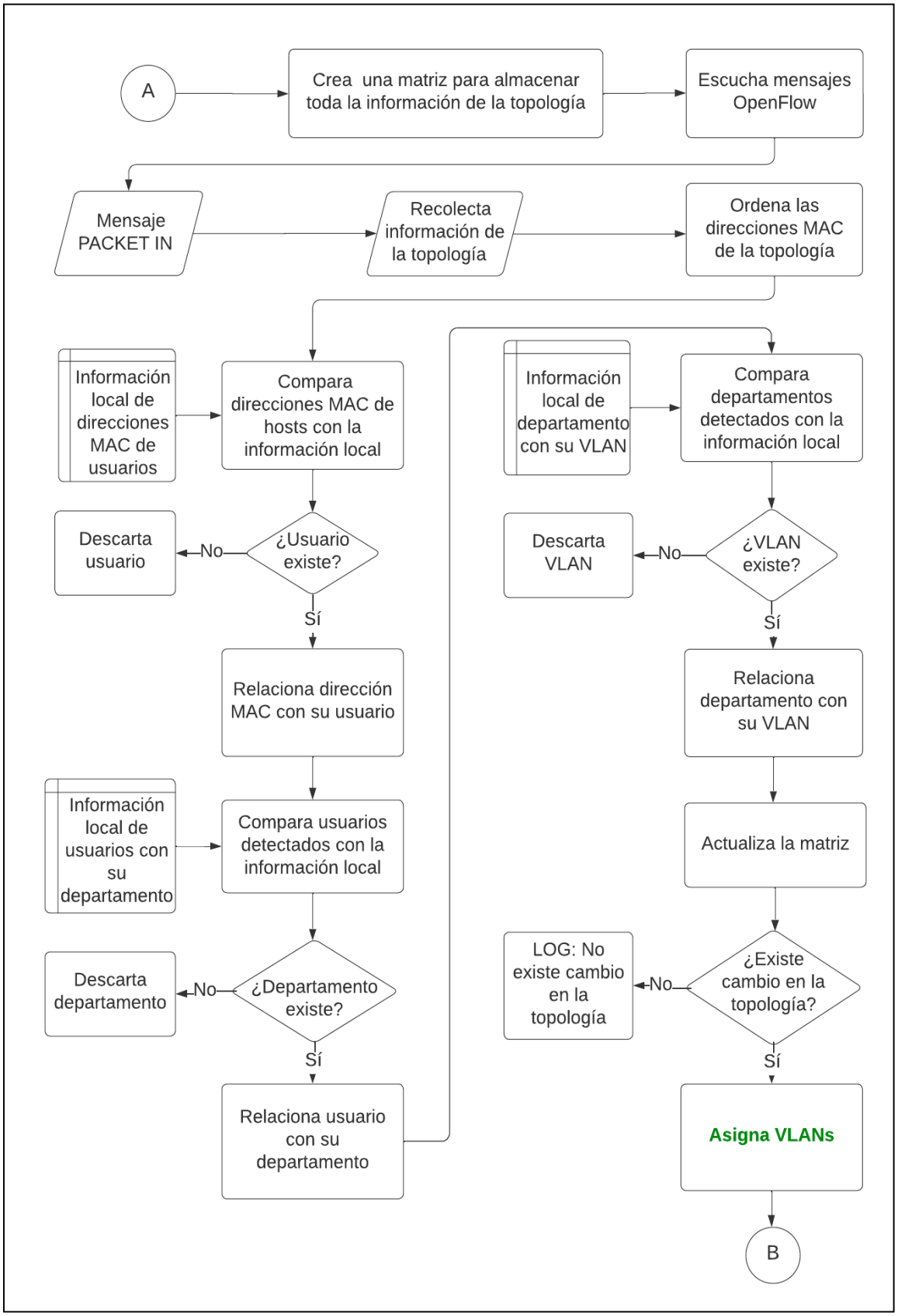
ASIGNACION ESTÁTICA DE VLANS EMPLEANDO SDN



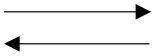
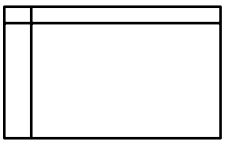
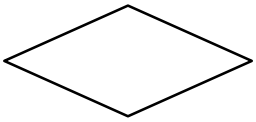
DIAGRAMAS DE FLUJO

ASIGNACION DINÁMICA DE VLANS EMPLEANDO SDN




A
Crea una matriz para almacenar toda la información de la topología
Escucha mensajes OpenFlow
Mensaje PACKET IN
Recolecta información de la topología
Ordena las direcciones MAC de la topología
Información local de direcciones MAC de usuarios
Compara direcciones MAC de hosts con la información local
Descarta usuario
No
¿Usuario existe?
Sí
Relaciona dirección MAC con su usuario
Información local de usuarios con su departamento
Compara usuarios detectados con la información local
Descarta departamento
No
¿Departamento existe?
Sí
Relaciona usuario con su departamento
Información local de departamento con su VLAN
Compara departamentos detectados con la información local
Descarta VLAN
No
¿VLAN existe?
Sí
Relaciona departamento con su VLAN
Actualiza la matriz
LOG: No existe cambio en la topología
No
¿Existe cambio en la topología?
Sí
Asigna VLANs
B

SIMBOLOGÍA

Nombre	Símbolos	Detalle
Líneas de flujo		Muestra la dirección y el sentido del flujo del proceso, conectando los símbolos.
Terminador		Información o acciones para comenzar procesos o para mostrar el resultado en el final del mismo
Proceso		Representa una etapa del proceso.
Archivo local		Simboliza un archivo local.
Decisión		Representa el punto del proceso donde se debe tomar una decisión. Dos fechas que salen del símbolo muestran la dirección del proceso en función de la decisión.
Datos		Elementos que alimentan y segeneran en el procedimiento.

Referencias

[1] A. Mendiola, J. Astorga, E. Jacob, y M. Higuero, “A Survey on the Contributions of Software-Defined Networking to Traffic Engineering,” *IEEE Communications Surveys and Tutorials*, vol. 19, num. 2, pp. 918–953, 2017.

[2] P. Goransson, C. Black, y T. Culver, *Software defined networks: a comprehensive approach*. Morgan Kaufmann, 2016.

[3] R. Jain y S. Paul, “Network virtualization and software defined networking for cloud computing: A survey,” *IEEE Communications Magazine*, vol. 51, num. 11, pp. 24–31, 2013.

[4] Moumita, “The IEEE 802 1Q Standard,” jul 2020. [En línea]. Disponible: https://www.tutorialspoint.com/the-ieee-802-1q-standard

[5] V. Mohan, “Raspberry Pi Models Comparison,” oct 2021. [En línea]. Disponible: https://raspberryexpert.com/raspberry-pi-models-comparison/

[6] M. Koerner y O. Kao, “Mac based dynamic vlan tagging with openflow for wlan access networks,” *Procedia Computer Science*, vol. 94, pp. 497–501, 2016.

[7] R. R. Zebari, S. Zeebaree, K. Jacksi, y H. M. Shukur, “E-business requirements for flexibility and implementation enterprise system: A review,” *International Journal of Scientific and Technology Research*, vol. 8, num. 11, pp. 655–660, 2019.

[8] H. Sufiev y Y. Haddad, “A dynamic load balancing architecture for sdn,” in *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*. IEEE, 2016, pp. 1–3.

[9] A. Prajapati, A. Sakadasariya, y J. Patel, “Software defined network: Future of networking,” in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2018, pp. 1351–1354.

[10] D. Perepelkin y I. Tsyganov, “Sdn cluster constructor: software toolkit for structures segmentation of software defined networks,” in *2019 XVI International Symposium”Problems of Redundancy in Information and Control Systems”(REDUNDANCY)*. IEEE, 2019, pp. 195–198.

[11] E. J. Huamanñahui Ñaccha, “Diseño de un sistema de autenticación de usuarios 802.1 con asignación de vlan dinámica,” 2010.

[12] W. L. Valles Rios, “Diseño de una red sdn para brindar una gestion centralizada de las configuraciones y una adecuada gestion de crecimiento de los nodos a la red de un operador de servicio.”

[13] H. ÇOTUK, A. ÖMERCİOĞLU, y N. ERGİNÖZ, “Ieee 802.1 x, radius and dynamic vlan assignment.”

[14] V.-G. Nguyen y Y.-H. Kim, “Sdn-based enterprise and campus networks: a case of vlan management,” *Journal of Information Processing Systems*, vol. 12, num. 3, pp. 511–524, 2016.

[15] N. Ravi, “Design and implementation of an sdn based authentication and separation mechanism for wifi users.”

[16] D. Gonzalez, C. Mellado, K. Waltam, y A. Lara, “Low-cost SDN switch comparison: Zodiac FX and raspberry PI,” in *Proceedings - 4th Jornadas Costarricenses de Investigacion en Computacion e Informatica, JoCICI 2019*. Institute of Electrical and Electronics Engineers Inc., aug 2019.

[17] B. Babayiit, S. Karakaya, y B. Ulu, “An implementation of software defined network with Raspberry Pi,” in *26th IEEE Signal Processing and Communications Applications Conference, SIU 2018*. Institute of Electrical and Electronics Engineers Inc., jul 2018, pp. 1–4.

[18] Y. Yamasaki, Y. Miyamoto, J. Yamato, H. Goto, y H. Sone, “Flexible access management system for campus VLAN based on openflow,” *Proceedings - 11th IEEE/IPSJ International Symposium on Applications and the Internet, SAINT 2011*, pp. 347–351, 2011.

[19] A. C. Jaramillo, R. Alcivar, J. Pesantez, y R. Ponguillo, “Cost effective test-bed for comparison of sdn network and traditional network,” in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2018, pp. 1–2.

[20] J. F. Guano Viscarra, “Prototipo de una SDN utilizando herramientas Open-Source,” Escuela Politécnica Nacional, Quito, Tech. Rep., 2017.

[21] S. E. Nazareno Arroyo, “Diseño e implementación de un prototipo sd-wan basado en raspberry pi,” Ph.D. dissertation, Universidad de Guayaquil. Facultad de Ciencias Matemáticas y Físicas , 2019.

[22] A. Hameed y M. Wasim, “On the study of SDN for emulating virtual lans,” *2019 8th International Conference on Information and Communication Technologies, ICICT 2019*, pp. 162–167, 2019.

[23] A. Gelberger, N. Yemini, y R. Giladi, “Performance analysis of Software-Defined Networking (SDN),” *Proceedings - IEEE Computer Society’s Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS*, pp. 389–393, 2013.

[24] H. Babbar y S. Rani, “Performance evaluation of QoS metrics in software defined networking using ryu controller,” *IOP Conference Series: Materials Science and Engineering*, vol. 1022, num. 1, 2021.

[25] J. Vora, S. Kaneriya, S. Tanwar, y S. Tyagi, “Performance Evaluation of SDN based Virtualization for Data Center Networks,” *Proceedings - 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages, IoT-SIU 2018*, num. August 2019, 2018.

[26] H. E. Egilmez, S. T. Dane, K. T. Bagci, y A. M. Tekalp, “Openqos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks,” in *Proceedings of the 2012 Asia Pacific signal and information processing association annual summit and conference*. IEEE, 2012, pp. 1–8.

[27] D. R. Rodríguez Herlein, C. A. Talay, C. N. González, y L. A. Marrone, “Explorando las redes definidas por software (sdn),” in *XXII Workshop de Investigadores en Ciencias de la Computación (WICC 2020, El Calafate, Santa Cruz).*, 2020.

[28] C.-S. Li y W. Liao, “Software defined networks,” *IEEE Communications Magazine*, vol. 51, num. 2, pp. 113–113, 2013.

[29] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, y S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, num. 1, pp. 14–76, 2015.

[30] ITU-T, “Y.3300: Framework of software-defined networking,” 2014. [En línea]. Disponible: https://www.itu.int/rec/T-REC-Y.3300/es

[31] W. Stallings, *Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud*. Addison-Wesley Professional, 2015.

[32] C. DeCusatis, *Handbook of fiber optic data communication: a practical guide to optical networking*. Academic Press, 2013.

[33] ONF, “Software-Defined Networking (SDN) Definition - Open Networking Foundation.” [En línea]. Disponible: https://opennetworking.org/sdn-definition/

[34] E. Haleplidis, K. Pentikousis, S. Denazis, J. Salim, D. Meyer, y O. Koufopavlou, “Rfc 7426: Software-defined networking (sdn): layers and architecture terminology,” *Internet Research Task Force (IRTF)*, 2015.

[35] A. Rawal, “Introduction to OpenFlow Protocol,” feb 2021. [En línea]. Disponible: https://www.section.io/engineering-education/openflow-sdn/#basics-of-openflow

[36] OpenDaylight, “Controller — Controller 6.0.1-SNAPSHOT documentation,” 2022. [En línea]. Disponible: https://docs.opendaylight.org/projects/controller/en/latest/dev-guide.html#overview

[37] J. B. Gibson, N. Lewis, M. A. Adena, y R. Wilson, “Selection for Ethanol Tolerance in Two Populations of Drosophila Melanogaster Segregating Alcohol Dehydrogenase Allozymes,” *Australian Journal of Biological Sciences*, vol. 32, num. 3, pp. 387–398, 1979.

[38] M. McCauley, “POX Wiki - Open Networking Lab - Confluence,” *Openflow.Stanford.Edu*, pp. 1–57, 2016. [En línea]. Disponible: https://openflow.stanford.edu/display/ONL/POX+Wiki

[39] A. OpenDaylight, “Visibility and Control - OpenDaylight,” 2023. [En línea]. Disponible: https://www.opendaylight.org/use-cases/visibility-and-control

[40] K. W. Kurose, J. F., Ross, “REDES DE COMPUTADORAS Un enfoque descendente,” *PEARSON Educación*, vol. 5, p. 793, 2010. [En línea]. Disponible: http://dialnet.unirioja.es/servlet/dcart?info=link&codigo=2741660&orden=170694

[41] P. Garimella, Y.-W. E. Sung, N. Zhang, y S. Rao, “Characterizing vlan usage in an operational network,” in *Proceedings of the 2007 SIGCOMM workshop on Internet network management*, 2007, pp. 305–306.

[42] X. Sun, Y.-W. Sung, S. D. Krothapalli, y S. G. Rao, “A systematic approach for evolving vlan designs,” in *2010 Proceedings IEEE INFOCOM*. IEEE, 2010, pp. 1–9.

[43] A. D. E. L. Protocolo, “UIT-T,” 2005.

[44] Cisco, “Inter − Switch Link and IEEE 802 . 1Q Frame Format,” Tech. Rep., 2006.

[45] H. Packard, “VLAN fundamentals,” 2014. [En línea]. Disponible: https://techhub.hpe.com/eginfolib/networking/docs/switches/5500hi/5998-5328_l2-lan_cg/content/377991789.htm

[46] A. Shereya, “Overview of VLAN Trunking and Encapsulation,” jul 2020. [En línea]. Disponible: https://www.section.io/engineering-education/vlan-trunking/

[47] Juniper, “What is 802.1X Network Access Control (NAC)?” [En línea]. Disponible: https://www.juniper.net/us/en/research-topics/what-is-802-1x-network-access-control.html

[48] Juniper Networks, “Learn About 802 . 1X Network Access Control (NAC),” pp. 1–13, 2016.

[49] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, y S. Shenker, “Extending Networking into the Virtualization Layer,” *8th ACM Workshop on Hot Topics inNetworks*, vol. VIII, p. 6, 2009. [En línea]. Disponible: http://www.icsi.berkeley.edu/pubs/networking/extendingnetworking09.pdf

[50] Linux Foundation, “What is Open vSwitch?” pp. 1–2, 2016. [En línea]. Disponible: https://docs.openvswitch.org/en/latest/intro/what-is-ovs/

[51] L. Foundation, “Why Open vSwitch?” [En línea]. Disponible: https://docs.openvswitch.org/en/latest/intro/why-ovs/#the-mobility-of-state

[52] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, y M. Casado, “The design and implementation of open vSwitch,” in *Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2015*, 2015, pp. 117–130.

[53] M. Kalpana, M. T. Student, y R. Dist, “Online monitoring of water quality using raspberry pi3 model b,” *International Journal of Innovative Technology and Research*, vol. 4, num. 6, pp. 4790–4795, 2016.

[54] W. Harrington, *Learning raspbian*. Packt Publishing Ltd, 2015.

[55] D. D. Dinu y M. Togan, “Dhcp server authentication using digital certificates,” in *2014 10th International Conference on Communications (COMM)*. IEEE, 2014, pp. 1–6.

[56] R. Droms, “Automated configuration of tcp/ip with dhcp,” *IEEE Internet Computing*, vol. 3, num. 4, pp. 45–53, 1999.