

**Escuela Superior Politécnica del Litoral**  
**Facultad de Ingeniería en Electricidad y Computación**

Prototipado Programable y Simulación Virtual de Redes SDN  
**Proyecto Integrador**

Previo a la obtención del Título de:  
**Ingeniero en Telecomunicaciones**

Presentado por:

**Shirley Sharon Medina Magües**

**Joel Fabian Tenezaca Mawyin**

GUAYAQUIL – ECUADOR

Año: 2023

## Dedicatoria

---

A quienes han estado incondicionalmente a mi lado, con profunda gratitud dedico este logro a quienes han sido mi fuerza y mi guía a lo largo de este camino.

A mis abuelos, quienes tanto en el cielo como en la tierra han sido fuente constante de inspiración. Su amor, y sabiduría me han guiado y enseñado a perseverar y trabajar muy duro, enfrentando cada desafío con determinación.

A mis padres, su sacrificio, amor y fuerza han sido la base de toda mi formación personal, académica y espiritual, son mi guía, mi ejemplo y la razón de mi ser.

A mis hermanos y tíos, quienes han sido mi apoyo incondicional en cada paso. Han sido mi fortaleza en los momentos más difíciles.

A mis amigos, que se han convertido en una extensión de mi familia, gracias por acompañarme y compartir lágrimas, risas y preocupaciones, por brindarme palabras de aliento y ayudar a seguir adelante. Tener su amistad es un tesoro invaluable.

Este proyecto es un testimonio del apoyo y presencia de cada uno de ustedes. Gracias infinitas, su influencia en mi vida perdurará para siempre en mi corazón.

*Shirley Sharon Medina Magües*

## **Dedicatoria**

---

El presente proyecto lo dedico a quienes han estado apoyándome en este largo trayecto brindándome fuerzas para seguir adelante.

A mis padres, cuyas enseñanzas las llevo presente en mi día a día y me demuestran a diario que tanto el sacrificio como el esfuerzo son importantes para seguir adelante mejorando en cada paso.

A mis hermanas, quienes brindan un enfoque distinto en mi día a día y me observan como un ejemplo a seguir.

A mis amigos, que en cada momento me brindan su apoyo y aliento para vencer cada obstáculo que se presente.

***Joel Fabian Tenezaca Mawyin***

## Agradecimientos

---

Quiero expresar mi sincero agradecimiento a mis profesores, especialmente al Mgtr. Fernando Vásquez y al Ph.D. Francisco Novillo, quienes han guiado y enriquecido mi formación con su experiencia y conocimientos. Al Ing. Aristóteles Amat por todo su apoyo y guía. Docentes cuyos consejos, orientación, amistad y confianza han sido invaluableles en esta trayectoria.

A mi familia, que con su apoyo emocional y su aliento constante me han motivado a perseverar y poder lograr mis objetivos.

A mis amigos y compañeros que compartieron este camino conmigo, sus ideas y opiniones han sido aportaciones importantes para mí.

A mi compañero Joel, que con su participación y contribuciones hemos podido llevar a cabo este proyecto.

Finalmente, quiero agradecer a la M.Sc. Verónica Soto y la Ph.D. María Antonieta Álvarez, que gracias a su paciencia y guía este proyecto se pudo llevar a cabo de la mejor manera posible.

*Shirley Sharon Medina Magües*



## **Agradecimientos**

---

Mis más sinceros agradecimientos a mis padres y hermanas que me han apoyado durante todo este proceso de aprendizaje y me han dado fuerzas para seguir adelante.

A mi tutora por guiarme en este proyecto, por su paciencia y comentarios críticos que han sido fundamentales llevar este trabajo a su mejor versión.

También le agradezco a mis amigos que con sus palabras de aliento y creencia en mí me han impulsado a superar cada obstáculo.

Finalmente, expreso mi gratitud a mi compañera de este trabajo por su contribución y dedicación de este.

***Joel Fabian Tenezaca Mawyin***

## Declaración Expresa

---

Nosotros Shirley Sharon Medina Magües y Joel Fabian Tenezaca Mawyin acordamos y reconocemos que la titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, información no divulgada y cualquier otro derecho o tipo de Propiedad Intelectual que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada durante el desarrollo de su trabajo de titulación, incluyendo cualquier derecho de participación de beneficios o de valor sobre titularidad de derechos, pertenecerán de forma total, perpetua, exclusiva e indivisible a LA ESPOL, sin limitación de ningún tipo. Se deja además expresa constancia de que lo aquí establecido constituye un “previo acuerdo”, así como de ser posible bajo la normativa vigente de transferencia o cesión a favor de la ESPOL de todo derecho o porcentaje de titularidad que pueda existir.

Sin perjuicio de lo anterior los alumnos firmantes de la presente declaración reciben en este acto una licencia de uso gratuita e intransferible de plazo indefinido para el uso no comercial de cualquier investigación, desarrollo tecnológico o invención realizada durante el desarrollo de su trabajo de titulación, sin perjuicio de lo cual deberán contar con una autorización previa expresa de la ESPOL para difundir públicamente el contenido de la investigación, desarrollo tecnológico o invención.

Así también autorizamos expresamente a que la ESPOL realice la comunicación pública de la obra o invento, por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual.

Guayaquil, 20 de septiembre de 2023.



---

**Shirley Sharon Medina Magües**



---

**Joel Fabian Tenezaca Mawyin**

## **Evaluadores**

---

**Ph.D. María Antonieta Álvarez Villanueva**

Profesor de Materia

---

**M.Sc. Verónica Alexandra Soto Vera**

Tutor de proyecto

## **Resumen**

Las empresas necesitan redes de comunicación más flexibles y eficientes para satisfacer las necesidades de sus aplicaciones. Las redes definidas por software (SDN) ofrecen una solución a este problema, ya que separan el control de la red de la infraestructura física. Este proyecto se centra en el diseño de un prototipo programable de red SDN utilizando simuladores de redes virtuales. El proyecto tiene un enfoque práctico y experimental para que los estudiantes puedan comprender las redes SDN y sus escenarios de implementación. El prototipo se basa en el software Mininet, que permite crear entornos de red SDN realistas. Los estudiantes pueden utilizar el prototipo para experimentar con diferentes configuraciones y escenarios de tráfico. Finalmente, el proyecto ha sido un éxito en su objetivo de proporcionar a los estudiantes un primer acercamiento a las redes SDN.

**Palabras Clave:** Redes, escalabilidad, software, emulador, mininet

## **Abstract**

*Enterprises and their interconnectivity needs have been growing rapidly. Traffic and network quality demands have increased, making dynamic network conditions challenging for conventional network architectures. This is how Software Defined Networking (SDN) is proposed as an innovative solution that can address these new needs for flexibility and efficiency in communication networks. This project focuses on the design of a programmable prototype SDN network, using virtual network simulators to expand the resources available for the implementation, manipulation and testing of the network with a focus on its application to the academic environment. This project has been developed with a practical and experimental approach to understand SDN networks and implementation scenarios. Two scenarios were proposed for the development of practices, where using virtualization and programming techniques, using Mininet software, to create a realistic environment where students can design and experiment with their configurations and those proposed in each guide, to analyze the impact of the different changes and traffic behavior in this type of environments controlled with tools such as Wireshark. Finally, it was possible to propose environments in which students can understand SDN networks and thus have a first approach to this technology that is constantly evolving.*

**Keywords:** *Networking, scalability, software, emulator, mininet*

## Índice General

<b>Resumen .....</b>	<b>I</b>
<b>Abstract .....</b>	<b>II</b>
<b>Índice General .....</b>	<b>III</b>
<b>Índice de figuras.....</b>	<b>V</b>
<b>Índice de tablas.....</b>	<b>VI</b>
<b>Abreviaturas.....</b>	<b>VII</b>
<b>Capítulo 1 .....</b>	<b>8</b>
<b>1.1 Introducción.....</b>	<b>9</b>
<b>1.2 Descripción del problema .....</b>	<b>9</b>
<b>1.3 Justificación del problema.....</b>	<b>10</b>
<b>1.4 Objetivos .....</b>	<b>11</b>
1.4.1 Objetivo General.....	11
1.4.2 Objetivos Específicos .....	11
<b>1.5 Marco Teórico .....</b>	<b>11</b>
1.5.1 Redes definidas por software (SDN) .....	11
1.5.2 Tecnologías y protocolos relacionados con SDN .....	13
1.5.3 Diseño de prototipos programables de redes definidas por software.....	14
<b>Capítulo 2 .....</b>	<b>16</b>
<b>2.1 Diseño de prototipo programable de red SDN.....</b>	<b>17</b>
2.1.1 Etapas del desarrollo.....	17
2.1.2 Definición de escenarios y requerimientos específicos de la red.....	17
2.1.2.1 Escenarios de red .....	18
2.1.2.2 Requerimientos específicos.....	19
2.1.3 Análisis de software de virtualización utilizado .....	20
2.1.4 Protocolos de control utilizados en la programación de la red SDN. ....	22
<b>Capítulo 3 .....</b>	<b>24</b>
<b>3.1 Implementación y pruebas .....</b>	<b>25</b>
3.1.1 Ambiente del desarrollo.....	25
3.1.2 Implementación del Prototipo SDN.....	25
3.1.3 Integración y configuración .....	26
3.1.4 Pruebas y resultados .....	27
<b>Capítulo 4 .....</b>	<b>31</b>
<b>4.1 Conclusiones y Recomendaciones .....</b>	<b>32</b>
4.1.1 Conclusiones.....	32
4.1.2 Recomendaciones .....	32

<b>Referencias .....</b>	<b>33</b>
<b>Anexos.....</b>	<b>36</b>
Anexo 1: Guía de instalación .....	36
Anexo 2: Práctica 1 .....	60
Anexo 3: Práctica 2.....	74

## Índice de figuras

Figura 1: Comparativa entre la arquitectura tradicional y la arquitectura SDN[11] .....	12
Figura 2: Arquitectura general de las redes SDN[8]. .....	12
Figura 3: Diagrama de flujo del desarrollo del diseño e implementación de una red SDN. ....	17
Figura 4: Topología para el escenario "Small Office" .....	18
Figura 5: Topología para el escenario "Centro Educativo pequeño" .....	18
Figura 6: Requerimientos específicos para el diseño del prototipo de red SDN. ....	19
Figura 7: Comparación del desempeño entre algunas herramientas[19] .....	21
Figura 8: Topología del escenario SOHO en miniedit. ....	26
Figura 9: Topología del escenario Centro educativo pequeño en miniedit. ....	27
Figura 10: Visualización en tiempo real por medio de Wireshark para la topología SOHO. ....	28
Figura 11: Visualización en tiempo real por medio de Wireshark para la topología SOHO. ....	29
Figura 12: Visualización en tiempo real por medio de Wireshark para la topología Centro Educativo Pequeño. ....	29
Figura 13: Visualización en tiempo real por medio de Wireshark para la topología Centro Educativo Pequeño. ....	30



## Índice de tablas

Tabla 1: Comparación de simuladores y emuladores de red en un contexto educativo[20].....	21
Tabla 2: Ejemplos de controladores[15] .....	23

## Abreviaturas

Término	Definición
<b>ESPOL</b>	Escuela Superior Politécnica del Litoral
<b>SDN</b>	Siglas en inglés para Redes Definidas por <i>Software</i> (Software Defined Network)
<b>OF</b>	Protocolo OpenFlow
<b>API</b>	Interfaces de Programación de Aplicaciones
<b>VPC</b>	Nube Virtual Privada
<b>TI</b>	Tecnología de la Información
<b>TCP</b>	Protocolo de Control de Transmisión
<b>MPLS</b>	Siglas en inglés para Conmutación de Etiquetas Multiprotocolo (Multiprotocol Label Switching)
<b>QoS</b>	Siglas en inglés para Calidad del Servicio (Quality of Service)
<b>ODL</b>	Protocolo OpenDayLight
<b>VoIP</b>	Voz sobre Protocolo de Internet
<b>VLAN</b>	Siglas en inglés para Red de Área Local Virtual (Virtual Local Area Network)
<b>SOHO</b>	Siglas en inglés de Oficina Pequeña (Small Office/ Home Office)
<b>IP</b>	Internet Protocol
<b>OMNET</b>	Operation and Maintenance New Equipment Training
<b>IoT</b>	Siglas en inglés de Internet de las Cosas (Internet of Things)
<b>WiFi</b>	Wireless Fidelity
<b>OVSDB</b>	Protocolo de Gestión de Base de Datos Open vSwitch

## **Capítulo 1**

## **1.1 Introducción**

Las necesidades de interconectividad de las empresas han crecido vertiginosamente en los últimos años, esto debido al avance tecnológico tanto en el hardware como en el software. Construir una red para un negocio o emprendimiento es necesario, y dado que su diseño e implementación muchas veces resulta tedioso y complicado por falta de conocimiento, las redes físicas han ido evolucionando en su manera de ser construidas. Con la tecnología actual, las empresas deben mantenerse competitivas para que sus sistemas de tráfico de datos se encuentren alineados con sus necesidades, este recurso debe a su vez, mantenerse a la par con los avances tecnológicos que son cada vez mayores, ya que las redes de telecomunicaciones son una parte crucial del desarrollo y crecimiento de las empresas[1].

Debido al aumento del tráfico de red y las exigencias de calidad en los últimos diez años, los requisitos de las redes han cambiado rápidamente, generando una demanda por una mayor eficiencia en la consecución de objetivos de extremo a extremo[2]. Las condiciones dinámicas de la red a las que el mundo se dirige actualmente son complicadas para las arquitecturas de red convencionales, porque no cuentan con una fluidez a la hora de articular los datos. "Software Defined Networks" (SDN) es un nuevo concepto de implementación de redes que permite agilizar los procesos de flujos de datos para que sean adaptables a cada necesidad particular[3].

Las SDN tienen como principal característica separar y optimizar la estructura del hardware de la estructura del software, lo que proporciona una gestión dinámica de la red y más flexibilidad al manipularla. Además, permite una administración y personalización de red a la medida de las necesidades del usuario, por lo que al simularla en ambientes virtuales es de gran utilidad para la comprensión de su funcionamiento[4]. En conclusión, este proyecto se enfoca en el diseño e implementación de un prototipo programable de una red definida por software que será orientada al ámbito académico.

## **1.2 Descripción del problema**

Las redes de comunicación de datos generalmente están compuestas por dispositivos finales interconectados, conocidos como hosts, que utilizan una infraestructura de red compartida. Esta infraestructura incluye elementos de conmutación como enrutadores y switches, así como enlaces de comunicación para la transmisión de datos entre los hosts. Sin embargo, estos enrutadores y switches suelen ser sistemas cerrados con interfaces de control limitadas y específicas del fabricante, lo que dificulta la evolución de la infraestructura de red una vez implementada. Actualizar protocolos existentes o introducir nuevos protocolos y servicios se

convierte en un obstáculo significativo en las redes actuales, incluyendo el acceso a Internet, que es una red de redes[5].

Mantener las redes actualizadas y acorde a las necesidades de cada empresa, es primordial, por ello se dedican grandes esfuerzos e importantes inversiones en el diseño e implementación de actualizaciones en sus redes físicas[6]. En los últimos años, debido al constante crecimiento del tráfico de red y las exigencias en cuanto a calidad, las redes han experimentado cambios muy rápidos y significativos [2]. Pero debido a esta velocidad a la que crece la demanda, las arquitecturas de red tradicionales que son estáticas se quedan atrás por la dinámica que ahora requiere la red[3]. En este sentido, resulta útil contar con un diseño de red que se adapte rápidamente a los requisitos cambiantes y optimice el uso de los recursos disponibles. Esto permite simplificar y agilizar el proceso de prueba de protocolos y servicios de red, que suele ser complejo y costoso debido al equipamiento necesario para simular redes de telecomunicaciones.

### **1.3 Justificación del problema**

Mantener las redes actualizadas y adaptadas a las necesidades cambiantes de las empresas es de gran importancia y tomando en cuenta que las arquitecturas de red tradicionales están limitadas en su capacidad de respuesta y eficiencia, la necesidad de adoptar soluciones más flexibles y dinámicas es primordial, por lo que con el fin de permitir que las redes sean adaptables, ha surgido un nuevo modelo de red denominado "Redes definidas por software"[3].

La SDN se desarrolló con el objetivo de fomentar la innovación y permitir un control programable sencillo del flujo de datos en la red. La separación del hardware de envío de la lógica de control facilita la implementación de nuevos protocolos y aplicaciones, la visualización y gestión directa de la red, y la consolidación de diversos dispositivos intermedios en un control de software. En lugar de aplicar políticas y ejecutar protocolos en dispositivos dispersos, la red se reduce a un hardware de envío "simple" y al controlador de red (o controladores) responsable(s) de la toma de decisiones[5].

La SDN es una tecnología emergente que permite la gestión centralizada y programable de redes de telecomunicaciones para así tener más adaptabilidad y optimizar los recursos, lo que puede mejorar la eficiencia y reducir los costos de gestión de redes[7]. Por esto, el diseño de un prototipo programable de red SDN se plantea como una solución, ya que los ingenieros de redes pueden programar y configurar la red de manera centralizada, lo que conlleva una reducción en el tiempo y los costos asociados a la gestión de redes[5].

## 1.4 Objetivos

### 1.4.1 Objetivo General

- Diseñar un prototipo programable de una red SDN utilizando simuladores de redes virtuales que amplíe los recursos disponibles para la implementación, manipulación y pruebas de red.

### 1.4.2 Objetivos Específicos

- Investigar los fundamentos teóricos y conceptuales de las SDN, y las plataformas de emulación de red de datos para el diseño de un prototipo programable de una red SDN.
- Diseñar una arquitectura programable de red para la realización de pruebas de configuración y control de la red SDN emulada.
- Analizar ventajas del prototipo de red SDN programable con redes tradicionales basándose en el costo y tiempo de implementación aplicado al ámbito académico.

## 1.5 Marco Teórico

En esta sección se presentan los conceptos básicos de las redes definidas por software (SDN), se explica sus beneficios y enumera los protocolos y tecnologías utilizados en estas, además de una breve descripción del diseño de prototipos programables.

### 1.5.1 Redes definidas por software (SDN)

Las redes definidas por software (SDN) son una tecnología, que como su nombre lo indica, utilizando controladores basados en software o interfaces de programación de aplicaciones (API), permiten la gestión de una red para administrar el enrutamiento principal en una red específica [8]. Cuando se habla de SDN se conoce que es posible la separación del plano de control con el plano de datos. El plano de control se refiere a los procesos que realiza una red para dirigir el tráfico de esta, estableciendo rutas de red y comunicando los protocolos a utilizarse; por otro lado, el plano de datos se basa en los datos que circulan por la red [9].

SDN fue desarrollada a partir de la necesidad de automatizar, optimizar y escalar las redes, ya sean de un centro de datos a nivel empresarial, una nube virtual privada (VPC) o un servicio en general; lo que les da un enfoque centralizado para la gestión de la infraestructura de red [10].

En la figura 1 se realiza una comparativa en donde se observa cómo en una red convencional, tanto la transferencia de paquetes de datos como el enrutamiento principal ocurren

en un mismo dispositivo de hardware. Sin embargo, con la tecnología SDN se establece un método para controlar el enrutamiento de los datos de manera separada de la infraestructura de hardware subyacente [8].

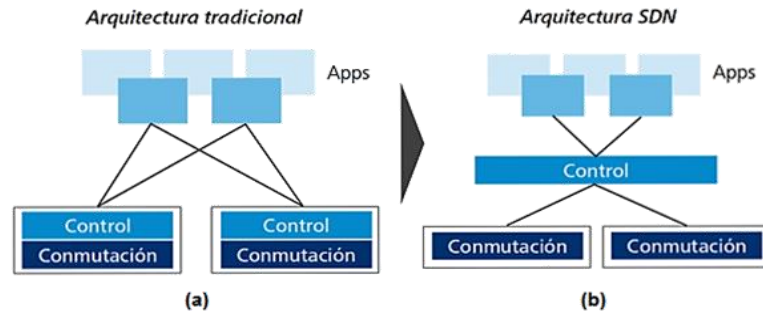


Figura 1: Comparativa entre la arquitectura tradicional y la arquitectura SDN[11] .

Las SDN presentan una arquitectura típica como la presentada en la figura 2, donde se muestran: las aplicaciones que son las que comunican las solicitudes de información sobre la red; los controladores que hacen uso de la información para decidir el proceso de enrutamiento de los paquetes de datos; y los dispositivos de red, los cuales reciben la información del controlador y ejecutan las acciones de envío, procesamiento y seguimiento de datos[12].

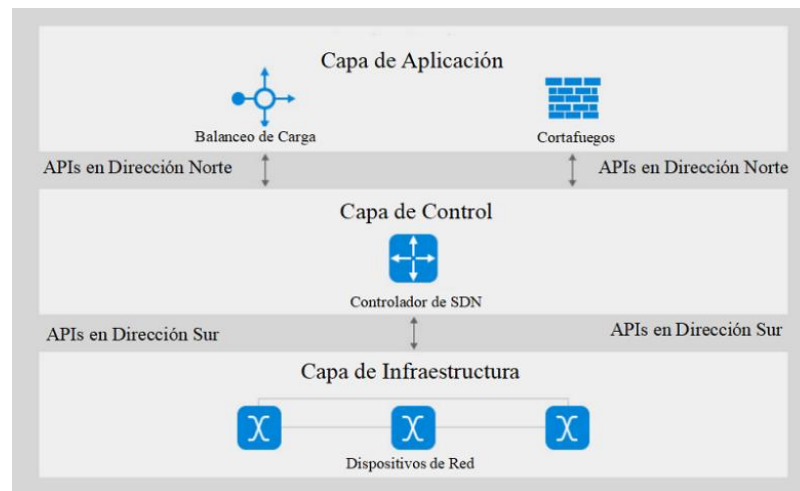


Figura 2: Arquitectura general de las redes SDN[8].

En la arquitectura de las redes SDN, el controlador juega un papel crucial al integrar los tres componentes o capas. Esta integración se divide en dos direcciones: norte y sur. La dirección norte se refiere a la integración entre el controlador y las aplicaciones, donde el controlador proporciona una interfaz para que las aplicaciones se comuniquen con él y accedan a la funcionalidad de la red SDN. Por otro lado, la dirección sur se refiere a la integración entre el controlador y los dispositivos de red, permitiendo al controlador gestionar y controlar el comportamiento de los dispositivos dentro de la red SDN. En sí, el controlador es el elemento

central en la arquitectura de las redes definidas por software, facilitando la integración de las aplicaciones en dirección norte y los dispositivos de red en dirección sur [8].

SDN brinda grandes beneficios para los operadores de red y TI mediante la visibilidad y automatización de la red, lo cual incluye:

- Mediante la programación se logra automatizar las configuraciones de red con el fin de obtener mayor escalabilidad y fiabilidad.
- Mayor flexibilidad al cambiar la operación que se lleva a cabo en la red con el fin de habilitar una aplicación o anular una tarea.
- Centralización en la visibilidad de una topología de red o en sus elementos, incluyendo el funcionamiento de toda la infraestructura de red [10].

Las redes SDN reducen la complejidad mediante la separación de sus planos, a su vez aumentan la seguridad y la escalabilidad de la automatización. Adicionalmente, las redes SDN aceleran el plazo de comercialización con la implementación de aplicaciones y servicios [13].

### ***1.5.2 Tecnologías y protocolos relacionados con SDN***

El protocolo más reconocido para la implementación de redes definidas por software es OpenFlow (OF). Este protocolo permite centralizar las decisiones relacionadas con la trayectoria de los datos o paquetes, lo que permite una programación autónoma e independiente del equipo central y los switches individuales en el centro de datos. OF facilita la comunicación entre el switch y el controlador, mejorando así la eficacia de la gestión centralizada de la red.

El protocolo OF es un estándar en la arquitectura de las redes SDN, el cual define la comunicación entre un controlador y un dispositivo de red. Este protocolo es normalmente utilizado por el controlador SDN y el switch, donde el controlador toma la información de las aplicaciones y las transforma en entradas de flujo para alimentar al conmutador. Además, OF se puede utilizar para supervisar las estadísticas de conmutadores y puertos al momento de gestionar la red [14].

En las redes SDN, el control y la inteligencia de la red se centralizan en un controlador SDN. Los switches SDN, que son dispositivos de conmutación de paquetes, reciben instrucciones del controlador para reenviar el tráfico. El controlador también se utiliza para configurar el enrutamiento y proporcionar nuevas funcionalidades a las aplicaciones a través de interfaces de programación de aplicaciones (API).

Un controlador SDN se describe de manera general como un sistema de software, o colección de sistemas, los cuales ofrecen:



- Gestionar el estado de la red, mediante una base de datos.
- Un modelo de datos sobre las relaciones entre los recursos que se gestionan, las políticas y otros servicios brindados por el controlador.
- Seguridad sobre el protocolo de control de transmisión (TCP) con la conexión entre el controlador y los elementos de la red.
- Protocolo basado en estándares (OpenFlow), para la obtención del estado de la red impulsada por las aplicaciones de los elementos de red [15].

Para la virtualización de redes se es necesario de un controlador SDN, la cual permita a los administradores la creación de redes virtuales de manera dinámica. La virtualización permite un completo aislamiento entre los segmentos de red, lo que es de utilidad con respecto al tema de seguridad, por ejemplo, al aislar los datos generados por un grupo de usuarios da paso a ejecutar estas acciones sin generar tráfico en la red. Esto es con el fin de tener una red virtual eficiente de manera que se configure de manera centralizada, con un total aislamiento entre otras redes, y con una configuración automatizada [15].

### ***1.5.3 Diseño de prototipos programables de redes definidas por software***

Dentro del desarrollo de una red SDN se tiene la elaboración de un despliegue de esta red aplicando el protocolo MPLS o Conmutación de Etiquetas Multiprotocolo y a su vez la generación de políticas de calidad de servicio o QoS para servicios de telefonía IP. Dicho proyecto presenta una topología tipo estrella con una combinación de redes SDN con las redes tradicionales; donde estas redes tradicionales son integradas mediante un middleware (Neutrón), los servicios de Voz sobre IP (VoIP) se almacenan en una nube privada y el controlador SDN se basa en el software OpenDayLight (ODL). Para el protocolo MPLS se realiza la simulación de la red en Mininet, la cual se configura desde el controlador ODL (OpenDayLight), para la configuración de los routers y la red local con sus respectivas subredes se llevaron a cabo en OpenStack [16].

Otro ejemplo de las redes SDN que se han implementado a lo largo del tiempo es el análisis de esta red llevado a cabo en el campus sur de la Universidad Politécnica Salesiana; donde se evaluó que la red era discontinua debido a tener el plano de control y el plano de datos en un mismo dispositivo. Para solución de este problema se separó estos planos y se implementó un controlador SDN con el protocolo OpenFlow; las pruebas llevadas a cabo para la solución fueron de manera virtual mediante simulación de una infraestructura SDN [17].

Dentro de la implementación de las SDN se tiene también la implementación de estas como prácticas de laboratorio en la carrera de Telecomunicaciones, con el fin de brindar los conocimientos básicos y la manera adecuada de implementarla; donde se utilizó Mininet como herramienta de simulación de redes y Ryu como controlador. La primera práctica se basa en la instalación de los respectivos programas y la prueba de su funcionamiento, la siguiente ya se brinda una topología donde se le aplica un firewall a la interfaz del router; otra práctica es centrada en la configuración del router para luego subir de nivel con la topología de red e implementar VLANs, con el fin de aislar el tráfico de tenants en la red propuesta [18].

## **Capítulo 2**

## 2.1 Diseño de prototipo programable de red SDN

En el presente capítulo se describe la metodología utilizada para diseñar y desarrollar el prototipo de una red SDN.

### 2.1.1 Etapas del desarrollo

En esta sección, se presentan las diferentes etapas del desarrollo de las redes SDN.

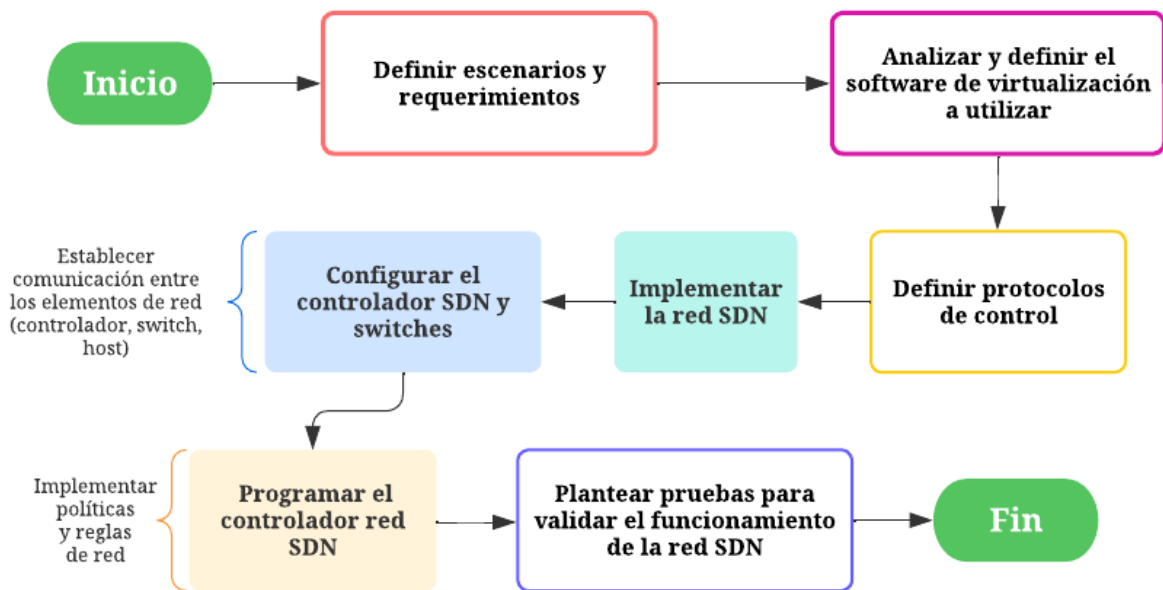


Figura 3: Diagrama de flujo del desarrollo del diseño e implementación de una red SDN.

Como se observa en la figura 3, las etapas de desarrollo comienzan con la definición de los escenarios, topologías y requerimientos. Luego, según las necesidades de estos, se puede escoger el software de virtualización más adecuado. Después, se definen los protocolos a utilizar. Una vez que se ha definido toda la base en el capítulo 3, se detalla el proceso a realizar para la implementación de la red SDN, realizar las configuraciones y programar el controlador, para finalmente realizar pruebas de funcionamiento como validación.

### 2.1.2 Definición de escenarios y requerimientos específicos de la red

Para la implementación de la red SDN se presentan dos escenarios a trabajar, tomando en cuenta los requerimientos específicos de la red.

## Escenarios de red

Se define dos escenarios principales en los cuales se implementan el prototipo de red SDN. Estos presentan características y desafíos diferentes que nos permiten evaluar si el prototipo es viable y efectivo en distintos entornos.

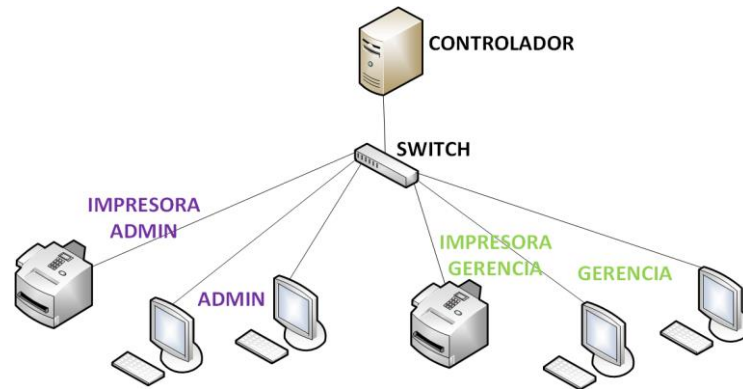


Figura 4: Topología para el escenario "Small Office"

El **escenario de SOHO** (por sus siglas en inglés Small office/Home Office) lo podemos observar en la figura 4, consiste en un entorno de red localizado en una pequeña oficina, donde se interconectan dispositivos periféricos, computadoras, impresoras, etc. Se espera que este proporcione una gestión simplificada y centralizada de la red, permita la segmentación de la red en áreas funcionales distintas y ofrezca servicios de calidad diferenciada.

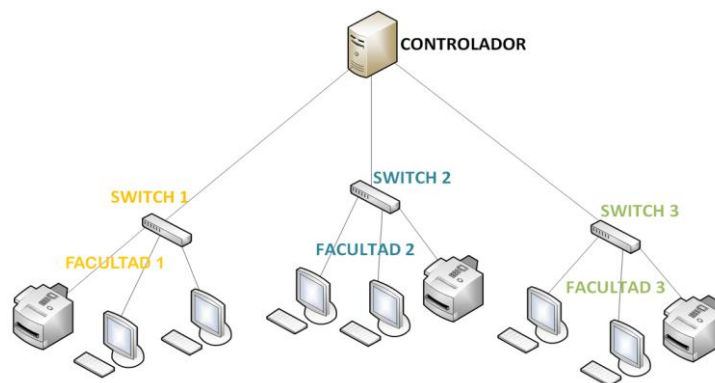


Figura 5: Topología para el escenario "Centro Educativo pequeño"

Por otro lado, en la figura 5 podemos ver la topología para el **escenario de red de un centro educativo pequeño** implica la implementación de una red a mayor escala que el anterior escenario, en donde se interconectan aulas, bibliotecas, laboratorios, salas comunes y otros espacios educativos. Aquí, el prototipo de red SDN debe ofrecer una administración centralizada de la red, permitir la implementación de políticas de acceso a Internet y proporcionar un control eficiente del tráfico de red para garantizar una experiencia de aprendizaje óptima.

### 2.1.2.1 Requerimientos específicos

Los requerimientos específicos para el diseño del prototipo de red SDN se desglosan en la figura 6 con los siguientes aspectos:

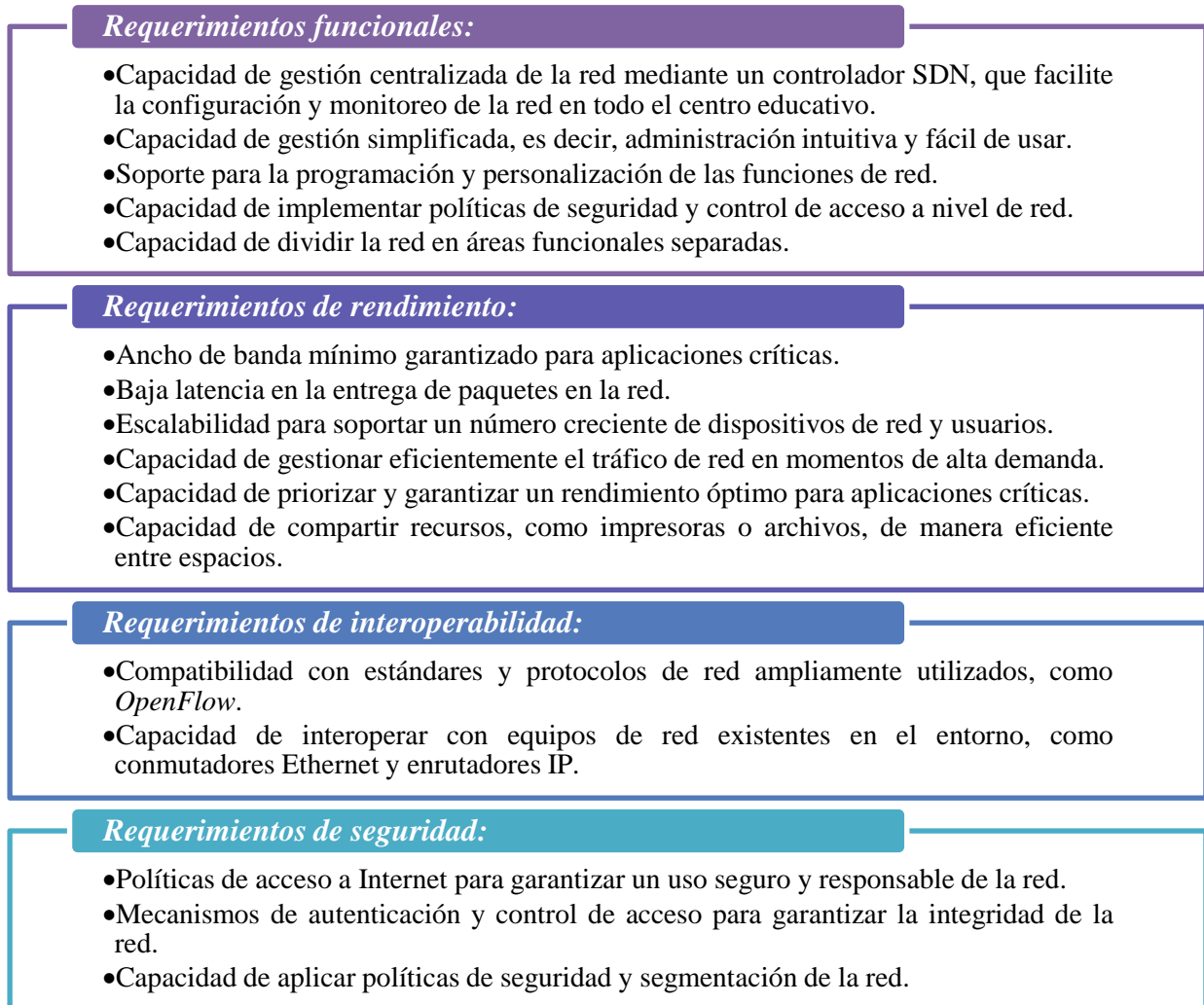


Figura 6: Requerimientos específicos para el diseño del prototipo de red SDN.

Estos dos escenarios resaltan la necesidad de un prototipo de red SDN capaz de proporcionar una gestión centralizada, admitir la segmentación de la red, brindar servicios premium diferenciados, hacer cumplir las políticas de seguridad, garantizar un rendimiento óptimo e integrarse con la interoperabilidad de los equipos de red existentes. Estos prototipos también deben cumplir requisitos específicos relacionados con el ancho de banda, la latencia, la escalabilidad, la gestión del tráfico, el uso compartido de recursos y la seguridad para satisfacer las necesidades del entorno de red respectivo.

### **2.1.3 Análisis de software de virtualización utilizado**

En la actualidad existen muchos *softwares* de virtualización, cada uno con sus ventajas y desventajas, características que se adaptan a las distintas necesidades que se presenten. Para este proyecto se decidió realizar pruebas con los softwares de virtualización OMNeT++ y Mininet.

#### **OMNeT++**

OMNeT++ es un simulador utilizado para modelar el tráfico de una red de telecomunicaciones, los protocolos y otros sistemas de hardware con el fin de evaluar el rendimiento de sistemas complejos. Dentro del tema de redes SDN nos brinda una mayor cantidad de codificación un tanto compleja y a su vez una forma gráfica de la misma para una perfecta simulación; pero requiere de un mayor rendimiento y almacenamiento del equipo dado a los paquetes de herramientas que presenta, por lo que lo hace de gran utilidad para representar topologías a nivel industrial debido a la complejidad de estas. OMNeT++ es un simulador muy útil a gran escala.

#### **Mininet**

Mininet es un emulador de red de telecomunicaciones de código abierto que permite la simulación de switches, routers, enlaces y terminales de una manera sencilla. Este emulador es utilizado con el fin de facilitar la enseñanza al momento de implementar una red como en el caso de la red SDN, y dado a su codificación sencilla e implementación de ejemplos que brinda el emulador se desarrolla un fácil entendimiento para los estudiantes. Mininet no demanda gran cantidad de recursos por lo que es una herramienta ligera, pero de igual manera permite la manipulación de redes complejas. Adicionalmente, es compatible con varias versiones por lo que facilita el uso de esta herramienta en ambientes educativos y en investigaciones sobre las redes.

Los investigadores, desarrolladores y estudiantes estudian el comportamiento de los sistemas en tiempo real mediante la realización de simulaciones, con el fin de obtener flexibilidad en términos de repetibilidad, escalabilidad y estabilidad. En la figura 7 se presenta la comparación del desempeño entre algunas herramientas, donde se muestra que OMNeT++ es menos realista en comparación a las herramientas de Mininet y PlanetLab; por otro lado, PlanetLab es comúnmente utilizado en proyectos, pero presenta un costo muy elevado y su complejidad aumenta considerablemente al momento de manipular esta herramienta [19].

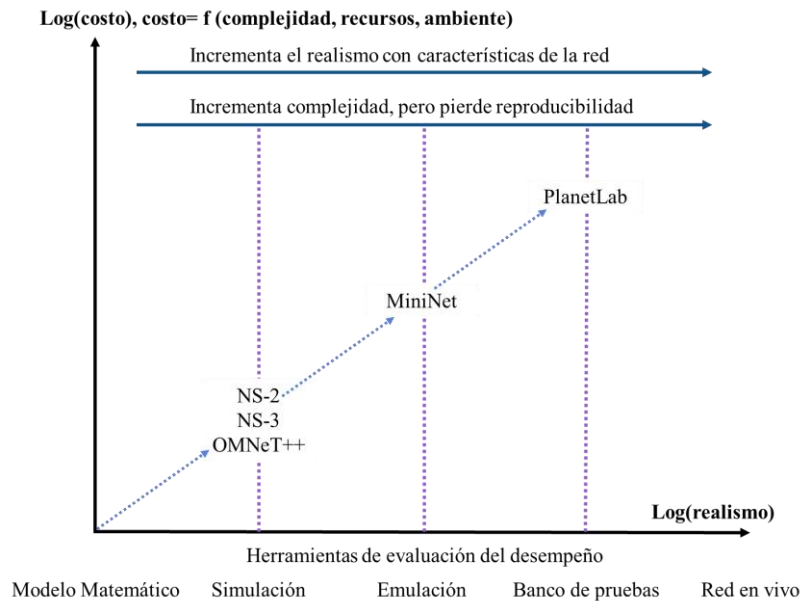


Figura 7: Comparación del desempeño entre algunas herramientas[19]

Al comparar simuladores y emuladores en el contexto académico (tabla 1) se puede notar que simuladores, como OMNeT++, Cisco Packet Tracer o NS3, y emuladores, como GNS3, NetSim o CORE, no pueden representar un ambiente real en un contexto de aprendizaje, ya que no soportan herramientas de análisis como Wireshark [20].

Tabla 1: Comparación de simuladores y emuladores de red en un contexto educativo[20]

	SIMULADORES			EMULADORES			
	OmNet++	Cisco Packet Tracer 7.0	NS3	GNS3	NetSim	CORE	Mininet
<i>Código abierto</i>	✓	✗	✓	✓	✗	✗	✓
<i>Lenguaje de Programación</i>	C++	JavaScript/CSS	C++/Python	Python	Java	Python	Python
<i>Última Actualización</i>	2017	2018	2018	2018	2018	2018	2017
<i>Adecuado para redes inalámbricas</i>	✓	✓	✓	✓	✓	✓	✓
<i>Soporta tráfico 6lowPAN para dispositivos IoT</i>	✗	✗	✓	✓	✓	✗	✓
<i>Soporta tráfico SDN</i>	✗	✗	✓	✓	✗	✗	✓
<i>Apoyo de la comunidad científica</i>	✓	✗	✓	✓	✗	✓	✓
<i>Emula interfaces de red reales</i>	✗	✓	✗	✗	✓	✓	✓
<i>Interfaz fácil de usar</i>	✓	✓	✗	✓	✓	✓	✓
✓ ✗	+1	-1	+3	+7	+1	+1	+7



En la tabla 1 se muestra una comparativa de los simuladores y emuladores, en esta se muestra las actualizaciones e interfaces que utilizan para el desarrollo de una red en específico y cada uno de estos presentan un lenguaje de programación que varían entre Python, Java y C++, algunos son de código abierto que son a los que se apunta. Dentro de los emuladores de código abierto destaca Mininet, que es un emulador en tiempo real que está basado en OpenFlow y SDN por lo que proporciona un entorno y configuraciones de fácil uso, permite la creación dinámica de redes, con host emulados, que en Linux trabajan con núcleos y switches que soportan el protocolo OpenFlow, lo que permite que tengan un gran nivel de flexibilidad [20].

Tomando en cuenta tanto la figura 7 como la tabla 1, se puede valorar las características de los emuladores presentados y ya que se plantea escenarios y topologías con finalidad académica, se llegó a la conclusión que el emulador que más se alinea a las necesidades del proyecto es Mininet.

#### ***2.1.4 Protocolos de control utilizados en la programación de la red SDN.***

**OpenFlow** es el primer protocolo, creado para redes SDN, en permitir la comunicación entre el plano de control y el plano de datos, es decir, facilita el envío de datos entre el controlador SDN y los routers o switches de la red. OpenFlow tiene sus respectivos controladores SDN y se lo describe como un sistema de software que ofrece la gestión del estado de la red, la cual implica una base de datos que sirve como repositorio para la información. Adicionalmente, estos controladores ofrecen una sesión de control segura sobre el Protocolo de Control de Transmisión (TCP) y un conjunto de APIs; todo esto es basado en el protocolo estándar.

**Protocolo de Gestión de Base de Datos *Open vSwitch (OVSDB)***, este protocolo es una configuración de OpenFlow encargado de la gestión de las operaciones realizadas de los conmutadores; tales como las interfaces y las configuraciones de políticas para la calidad de servicio. Por otro lado, Open vSwitch es un dispositivo virtual cuya función es la de un switch específico para las redes SDN con el fin de automatizar la red.

Al implementar Open vSwitch se tiene un grupo de administradores y controladores, los cuales brindan información de la configuración que se tiene al servidor de la base de datos de switch; a su vez los controladores utilizan OpenFlow para comunicar los detalles que se tiene al enviar los paquetes por la red a través de estos switches [15] [21] [22].

**Tabla 2: Ejemplos de controladores[15]**

	<i>Floodlight</i>	<i>OpenDayLight (ODL)</i>	<i>NOX</i>	<i>POX</i>	<i>Ryu</i>
<i>OpenFlow</i>	Versión 1.0	Versión 1.0	Versión 1.0	Versión 1.0	Versiones 1.0, 1.2, 1.3 y extensiones
<i>Virtualización</i>	<i>Mininet y Open vSwitch</i>	<i>Mininet y Open vSwitch</i>	<i>Mininet y Open vSwitch</i>	<i>Mininet y Open vSwitch</i>	<i>Mininet y Open vSwitch</i>
<i>Lenguaje de programación</i>	<i>Java</i>	<i>Java6</i>	<i>C++</i>	<i>Python</i>	<i>Python</i>
<i>Plataformas</i>	<i>Linux, Mac OS, Windows</i>	<i>Linux, Mac OS, Windows</i>	<i>Linux</i>	<i>Linux, Mac OS, Windows</i>	<i>Linux</i>
<i>Interfaz gráfica</i>	<i>Web</i>	<i>Web</i>	<i>Python+, QT4</i>	<i>Python+, QT4, Web</i>	<i>Web</i>

En la tabla 2 se presentan varios ejemplos de controladores con el detalle de las versiones de OpenFlow con la que son compatibles; siendo el controlador Ryu el que más versiones abarca. Dentro de los softwares de virtualización Mininet destaca en todos los controladores; y en los lenguajes de programación solo NOX utiliza C++, el resto se varía entre Java y Python. Adicionalmente, se presenta que la plataforma o sistema operativo de Linux es compatible con cada uno de los controladores y de igual manera la interfaz gráfica de la Web.

Para el uso de una máquina virtual es necesario del sistema operativo Linux, que está presente en todos los controladores mostrados, por lo que se establece una mayor prioridad en la versión del protocolo OpenFlow y en el lenguaje de programación más compatible como lo es Python; dentro de la tabla 2 solo dos controladores utilizan este lenguaje el cual son POX y Ryu.

## **Capítulo 3**

### **3.1 Implementación y pruebas**

En el presente capítulo se describe el proceso de implementación y pruebas de un prototipo de red SDN. Este proceso se divide en las siguientes etapas: Ambiente del desarrollo, implementación del prototipo, configuración del prototipo y pruebas de funcionalidad.

#### ***3.1.1 Ambiente del desarrollo***

El prototipo es desarrollado utilizando el software de virtualización Mininet, este emulador permite diseñar la topología desde su editor gráfico llamado Miniedit.py así como trabajar desde consola por medio de códigos. Estos prototipos serán desarrollados de las dos maneras para permitir la comprensión de las redes definidas por software. La instalación del software y los componentes necesario para llevar a cabo las topologías se encuentran en el anexo 1 denominado “Guía de instalación”.

#### ***3.1.2 Implementación del Prototipo SDN***

El prototipo de red SDN está compuesto por:

##### **Escenario 1: SOHO**

- 1 controlador
- 1 switch
- 6 hosts

##### **Escenario 2: Centro educativo pequeño**

- 1 controlador
- 3 switches
- 9 hosts

Ambos escenarios presentan características similares ya que trabajan con el protocolo OpenFlow, que en el software mininet ofrece la gestión del estado de la red a través de su controlador SDN. Además, se utilizan switches (1 en el escenario SOHO y 3 en el escenario del centro educativo) que permiten conectar los hosts respectivos de cada topología. En ambos escenarios los hosts representan 1 impresora y 2 usuarios (por área/departamento/facultad), estos pueden escalarse según la necesidad de la topología que se plantee.

Para evitar problemas de red al momento de escalar, se han utilizado VLANs y segmentación de red para dividir la red en áreas, departamentos o facultades. Esto evita que dos departamentos utilicen la misma IP para un host, lo que podría bloquear la red o perder datos.

### 3.1.3 Integración y configuración

Para la integración y configuración se utiliza el emulador mininet y con la ayuda de miniedit se puede observar gráficamente la representación de los equipos a utilizar en los escenarios propuestos, además en la consola de mininet se visualizan los elementos agregados y las configuraciones realizadas al ejecutar los diagramas mostrados. Como se presenta la figura 8a en la que se tiene la topología SOHO implementada en miniedit y en la figura 8b las respectivas configuraciones en la consola de mininet.

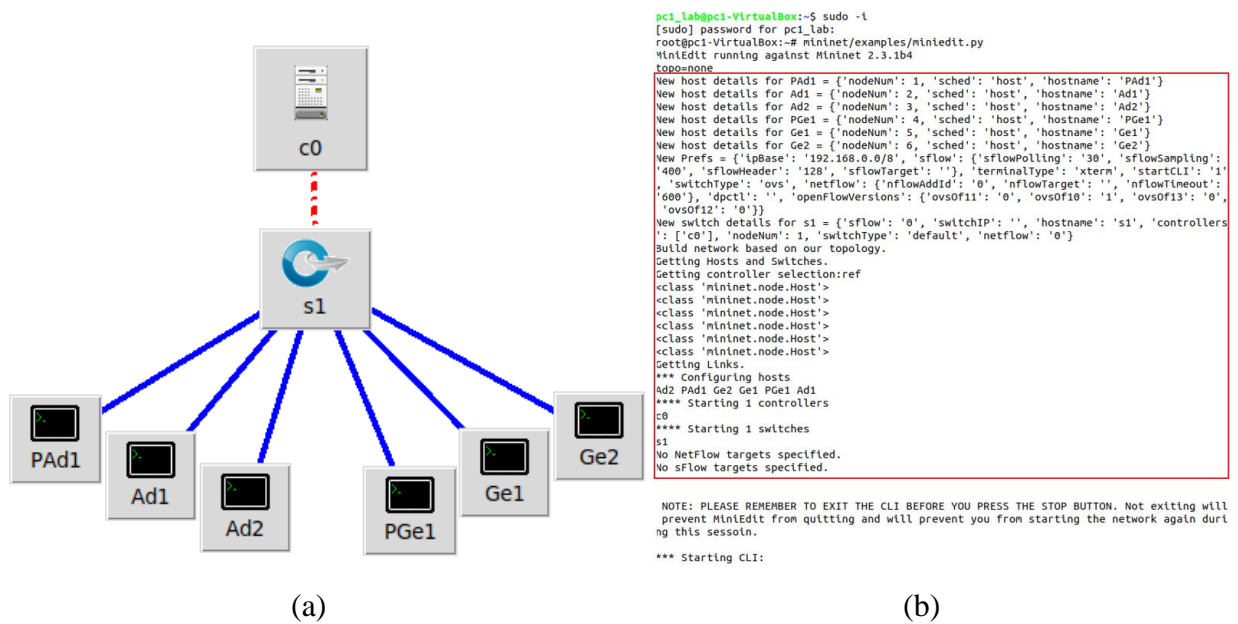


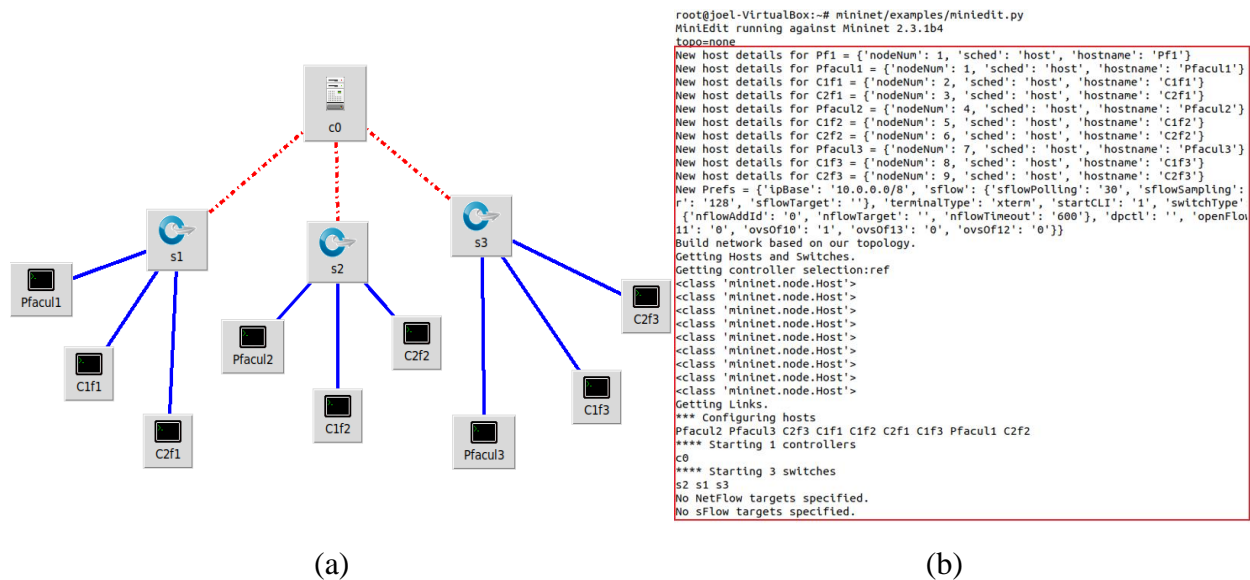
Figura 8: (a) Topología del escenario SOHO en miniedit. (b) Vista de consola de mininet al momento de implementar la topología en miniedit.

En donde,

- c0 corresponde al controlador
- s1 corresponde al switch
- PAd1 y PGe1 son las impresoras de Administración y Gerencia
- Ad1, Ad2, Ge1 y Ge2 son los hosts que pertenecen a Administración y Gerencia respectivamente

Estos elementos y sus configuraciones básicas se evidencian en la vista de consola de la figura 8b.

En la figura 9a se tiene la topología Centro Educativo Pequeño implementada en miniedit y en la figura 9b las respectivas configuraciones en la consola de mininet.



**Figura 9: (a) Topología del escenario Centro educativo pequeño en miniedit. (b) Vista de consola de mininet al momento de implementar la topología en miniedit.**

En donde,

- c0 corresponde al controlador
- s1, s2 y s3 corresponden a los switches
- Pfacul1, Pfacul2 y Pfacul3 son las impresoras de las 3 facultades
- C1f1, C2f2, C1f2, C2f2, C1f3 y C2f3 son los hosts que pertenecen a las facultades 1, 2 y 3 respectivamente.

Estos elementos y sus configuraciones básicas se evidencian en la vista de consola de la figura 9b.

### 3.1.4 Pruebas y resultados

Una vez llevada a cabo la implementación y la conexión entre los equipos correspondientes se realiza la configuración de estos con respecto a las IPs y las VLANs para lograr una buena intercomunicación entre estos.

Para la comunicación que se tiene entre los hosts de cada VLANs se utiliza el protocolo “Open vSwitch” en los switches utilizados en las topologías con el fin de que exista comunicación con el controlador, que posee la configuración respectiva, para llevar a cabo la comunicación entre

hosts por medio de las VLANs implementadas. Cabe mencionar que para las redes SDN el controlador trabaja con el protocolo OpenFlow para que las configuraciones específicas en cada dispositivo se pueda llevar a cabo de manera correcta en la virtualización de estas.

Dentro de cada configuración en los hosts se toma en cuenta lo escalable de cada red SDN con el fin de permitir cambios una vez realizada la implementación, es decir, la capacidad de aumentar más dispositivos a la red pensando en el aumento de tamaño de cada red. De igual manera en el caso de eliminar algún dispositivo, la escalabilidad de esta permite la eliminación o cambio sin perder la comunicación de la red.

En el caso de aumentar un nuevo grupo de dispositivos, se implementa una nueva VLAN con los equipos que se requieran o exijan, de esta manera se puede personalizar ambas topologías a gusto y se da un ejemplo de una implementación física de una red SDN de manera virtual.

La implementación y configuraciones para cada escenario se encuentran en el anexo 2 y 3 denominados: “Práctica 1” y “Práctica 2”. Dentro de las prácticas mencionadas se encuentran las instrucciones y procesos para realizar las pruebas de ping, evaluaciones de los nodos y pruebas en tiempo real mediante Wireshark como se muestra en las figuras 10 y 11 para la topología SOHO, visualizando los protocolos que presentan los hosts y controlador.

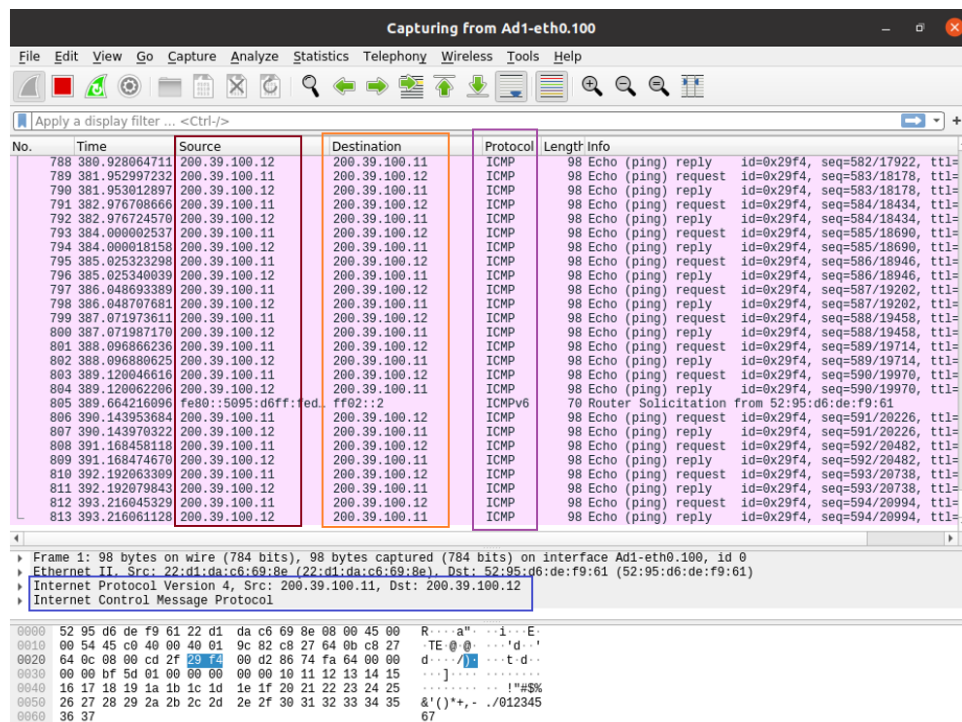


Figura 10: Visualización en tiempo real por medio de Wireshark para la topología SOHO.

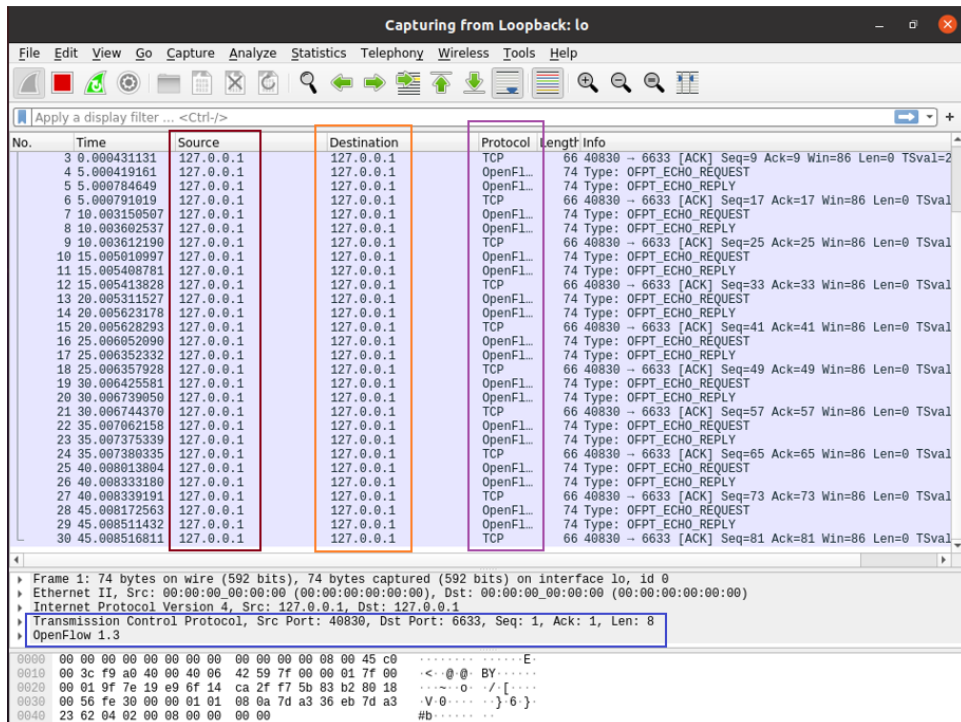


Figura 11: Visualización en tiempo real por medio de Wireshark para la topología SOHO.

De igual manera se visualizan las pruebas de conectividad en tiempo real para la topología Centro Educativo Pequeño mencionando los protocolos de comunicación entre hosts y el protocolo del controlador (Figura 12 y 13).

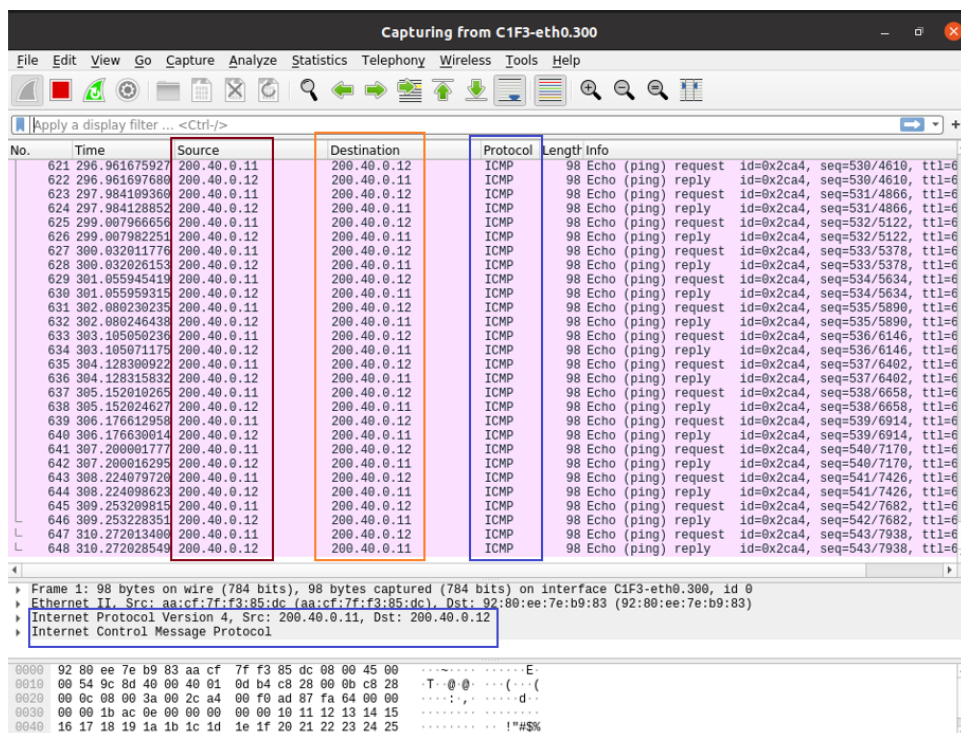


Figura 12: Visualización en tiempo real por medio de Wireshark para la topología Centro Educativo Pequeño.



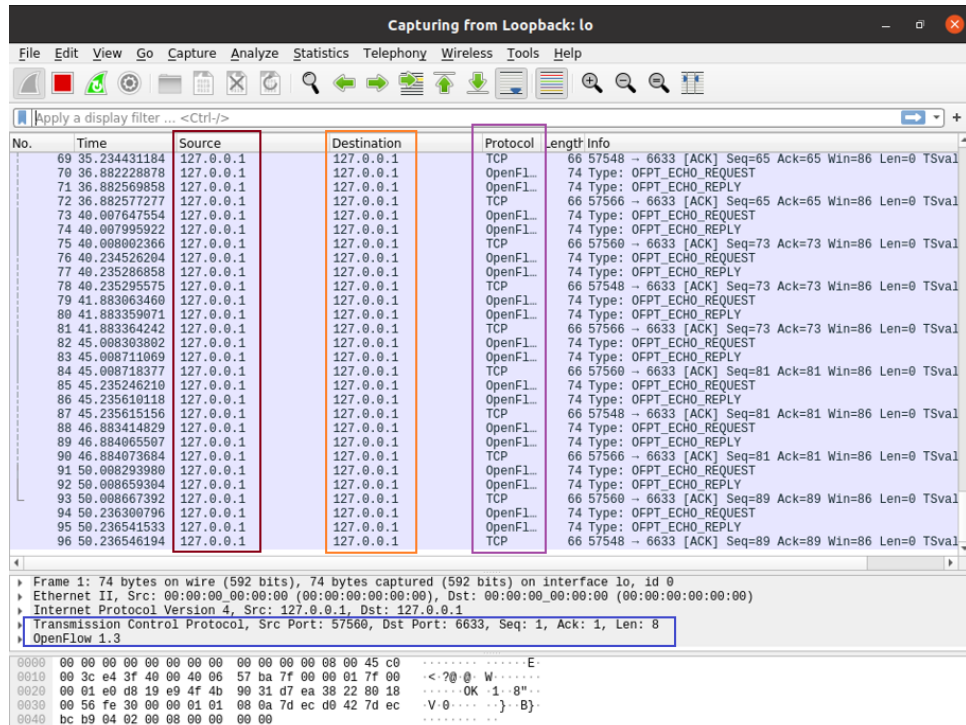


Figura 13: Visualización en tiempo real por medio de Wireshark para la topología Centro Educativo Pequeño.

Para ambas prácticas se realizan pruebas en tiempo real. Estas pruebas se ejecutan desde el controlador o desde uno de los hosts. A partir de estos puntos, se visualiza qué protocolos se utilizan, la comunicación entre qué dispositivos está ocurriendo y el tiempo que tarda. El objetivo de estas pruebas es comprender cómo se realiza la comunicación en una red definida por software y cómo interactúan los dispositivos y demás elementos con el controlador SDN.

## **Capítulo 4**

## **4.1 Conclusiones y Recomendaciones**

### **4.1.1 Conclusiones**

- La investigación realizada permitió comprender y definir los puntos principales respecto a las redes definidas por software, además debido a la existencia de varios simuladores y emuladores, se logró seleccionar un emulador con facilidad de armar y configurar topologías de redes SDN en un ambiente académico.
- Se definieron 2 topologías para ser emuladas dentro del laboratorio, una básica para una primera aproximación al tema llamada “Oficina pequeña” y una más grande llamada “Centro educativo pequeño”. Estas topologías planteadas se desarrollaron dentro del ambiente proporcionado por el software “Mininet” y su graficador llamado “Miniedit”, se pudo trabajar con controladores, switches OpenFlow y hosts, en los cuales se realizaron las configuraciones pertinentes y las pruebas planteadas para verificar el funcionamiento de la red.
- La emulación de redes muy cercanas a la realidad, con funciones aplicables sin gastos en hardware e implementaciones, permite a los estudiantes tener un acercamiento a la industria con el fin de experimentar, practicar y realizar pruebas de funcionamiento de redes SDN de manera sencilla y parecida a la implementación en la vida real; si bien las redes tradicionales aún se mantienen en muchas áreas, las redes definidas por software poco a poco están ganando espacio y las empresas las están adoptando por lo que es importante tener una experiencia previa que permita a los estudiantes estar más capacitados y preparados para los nuevos retos.

### **4.1.2 Recomendaciones**

- Es importante tomar en cuenta que el uso de los emuladores y simuladores representan consumo de recursos de computadora, especialmente si se necesita crear máquinas virtuales por lo que se recomienda contar con equipos con capacidad de al menos 520Gb de almacenamiento y 8Gb de memoria RAM para soportar la instalación y el uso de estos softwares.
- Las prácticas deben tener instrucciones detalladas para evitar que los estudiantes presenten problemas al momento de ejecutar la red, así mismo estos deben seguir paso a paso las mismas para no presentar errores de funcionamiento tanto en el emulador como en la red.

## Referencias

- Miguel Fabricio Bone Andrade, Jaime Darío Rodríguez Vizuete, Sandra María Sosa
- 1] Calero, and Luis Alfonso Núñez Freire, "Perspectivas de aplicación e investigación en Software Defined Networking SDN," *Conciencia Digital*, vol. 4, pp. 121-133, enero - marzo 2021.
- Mosab Hamdan et al., "A comprehensive survey of load balancing techniques in
- 2] software-defined network," *Journal of Network and Computer Applications*, vol. 174, enero 2021. [Online].  
<https://www.sciencedirect.com/science/article/abs/pii/S1084804520303222>
- Juan Camilo Correa Chica, Jenny Cuatindioy Imbachi, and Juan Felipe Botero Vega,
- 3] "Security in SDN: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, junio 2020. [Online].  
<https://www.sciencedirect.com/science/article/abs/pii/S1084804520300692>
- Miller Ramírez Giraldo and Ana María López Echeverry, "Redes de datos definidas
- 4] por software - SDN, arquitectura, componentes y funcionamiento," *Journal de Ciencia e Ingeniería*, vol. 10, pp. 55-61, agosto 2018. [Online].  
<https://jci.uniautonomo.edu.co/2018/2018-7.pdf>
- Bruno Nunes Astuto, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and
- 5] Thierry Turetli, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," enero 2014. [Online]. <https://inria.hal.science/hal-00825087v5/document>
- Würzburg. (2022, agosto) How Software Defined Networking (SDN) Continues to
- 6] Change Telco Network Management. [Online].  
<https://www.infosim.net/stablenet/blog/how-sdn-changes-telco-network-management/>
- Diego Kreutz et al., "Software-Defined Networking: A Comprehensive Survey,"
- 7] *Proceedings of the IEEE*, vol. 103, pp. 14-76, enero 2015. [Online].  
<https://arxiv.org/pdf/1406.0440.pdf>
- FS Community. (2022, junio) Redes Definidas por Software (SDN): Tipos, Ventajas
- 8] y Aplicaciones. [Online]. <https://community.fs.com/es/blog/software-defined-networking-sdn-types-advantages-and-applications.html>

- Cloudflare. (2023) ¿Qué es una red definida por software (SDN)? [Online].  
 9] <https://www.cloudflare.com/es-es/learning/network-layer/what-is-sdn/>
- John Apostolopoulos. (2018, junio) Why is intent-based networking good news for  
 10] software-defined networking? [Online]. <https://blogs.cisco.com/analytics-automation/why-is-intent-based-networking-good-news-for-software-defined-networking>
- Marco Antonio Calderón Ureña, Alejandro Sandoval Gonzalez, Esteban Umaña  
 11] Coto, and Julio Córdoba Retana. Uso de las redes definidas por software en el rendimiento integral de las telecomunicaciones como solución a la escalabilidad y la gestión de los recursos informáticos. [Online].  
<https://repositorio.ulacit.ac.cr/bitstream/handle/123456789/10582/REF-1640290637-1.pdf?sequence=1&isAllowed=y>
- VMware. (2023) Redes definidas por software. [Online].  
 12] <https://www.vmware.com/es/topics/glossary/content/software-defined-networking.html#:~:text=La%20arquitectura%20t%C3%ADpica%20de%20SDN%20se%20divide%20en,informaci%C3%B3n%20del%20controlador%20sobre%20d%C3%B3nde%20mover%20los%20datos.>
- Cisco. (2023) Redes definidas por software. [Online].  
 13] [https://www.cisco.com/c/es\\_mx/solutions/software-defined-networking/overview.html](https://www.cisco.com/c/es_mx/solutions/software-defined-networking/overview.html)
- Willies Ogola. (2021, febrero) Introduction to OpenFlow Protocol. [Online].  
 14] <https://www.section.io/engineering-education/openflow-sdn/#what-is-the-openflow-protocol>
- Alejandro García Centeno, Carlos Manuel Rodríguez Vergel, Caridad Anías  
 15] Calderón, and Frank Camilo Casmartíño Bondarenko, "Controladores SDN, elementos para su selección y evaluación.," *Revista Telemática*, vol. 13, septiembre 2014. [Online].  
<http://revistatelematica.cujae.edu.cu/index.php/tele>
- Mesías Mauricio Fernández Mora and Roberth Fernando Ulloa Banegas, Despliegue  
 16] de una red SDN aplicando el protocolo MPLS y generando políticas de QoS para servicios de telefonía IP., 2016.
- Milton Fidel Chamorro Armijos and Jhostin Andrés López Vallejo, "Análisis de  
 17] implementación de una red SDN en el campus sur de la Universidad Politécnica," Quito, 2022.

- Randa Lamallam. Propuesta de entorno y prácticas de laboratorio para tecnologías
- 18] SDN. [Online]. [https://oa.upm.es/49291/1/PFC\\_RANDA\\_LAMALLAM.pdf](https://oa.upm.es/49291/1/PFC_RANDA_LAMALLAM.pdf)
- Sohaib Manzoor, Yachao Yin, Yayu Gao, Xiaojun Hei, and Wenqing Cheng. (2020,
- 19] agosto) A Systematic Study of IEEE 802.11 DCF Network Optimization from Theory to Testbed. [Online]. [https://www.researchgate.net/publication/343782769\\_A\\_Systematic\\_Study\\_of\\_IEEE\\_802\\_11\\_DCF\\_Network\\_Optimization\\_from\\_Theory\\_to\\_Testbed/figures?lo=1](https://www.researchgate.net/publication/343782769_A_Systematic_Study_of_IEEE_802_11_DCF_Network_Optimization_from_Theory_to_Testbed/figures?lo=1)
- Cristina Alcaraz, Antonio Ortega, and Rodrigo Roman. (2018, octubre) The Role of
- 20] Software-Defined Networks for Practical Learning in the Engineering Areas. [Online]. [https://www.researchgate.net/publication/328591250\\_The\\_Role\\_of\\_Software-Defined\\_Networks\\_for\\_Practical\\_Learning\\_in\\_the\\_Engineering\\_Areas](https://www.researchgate.net/publication/328591250_The_Role_of_Software-Defined_Networks_for_Practical_Learning_in_the_Engineering_Areas)
- Dayana, Cáceres, Carlos Guzmán. (2019) Análisis del desempeño de redes definidas
- 21] por software (SDN) frente a redes con arquitectura TCP/IP. [Online]. <https://1library.co/document/zkwrn3ez-analisis-desempeno-redes-definidas-software-frente-redes-arquitectura.html>
- Michelle McNickle. (2014, septiembre) Cinco protocolos SDN que no son
- 22] OpenFlow. [Online]. <https://www.computerweekly.com/es/cronica/Cinco-protocolos-SDN-que-no-son-OpenFlow>
- Idimad 360. (2020, julio) Evolución de las telecomunicaciones. [Online].
- 23] <https://aselcom.com/blog/actualidad/evolucion-de-las-telecomunicaciones>
- Martin Fransman, "Evolution of the Telecommunications Industry into the internet
- 24] age," *The International Handbook on Telecommunication Economics*, enero 2021. [Online]. <https://www.pearsonhighered.com/assets/samplechapter/0/1/3/0/0130281360.pdf>

# Anexos

## Anexo 1: Guía de instalación

# Guía de Instalación

## Instalación de VirtualBox y Mininet

### Objetivos:

- Instalar el software para emular máquinas virtuales - Virtual Box.
- Instalar el software emulador de redes - Mininet.
- Familiarizarse con la máquina virtual y el simulador de red instalado.

### Marco teórico:

#### Virtual Box

VirtualBox es un software de código abierto para virtualizar la arquitectura informática x86. Actúa como un hipervisor, creando una VM (máquina virtual) donde el usuario puede ejecutar otro SO (sistema operativo).

El sistema operativo en el que se ejecuta VirtualBox se denomina sistema operativo "host/anfitrión". El sistema operativo que se ejecuta en la máquina virtual se denomina sistema operativo "guest/invitado". VirtualBox es compatible con Windows, Linux o macOS como sistema operativo host.

Al configurar una máquina virtual, el usuario puede especificar cuántos núcleos de CPU y cuánto RAM y espacio en disco se deben dedicar a la máquina virtual. Cuando la máquina virtual se está ejecutando, se puede "pausar". La ejecución del sistema se congela en ese momento y el usuario puede reanudar su uso más tarde.

Los sistemas operativos invitados compatibles con VirtualBox incluyen:

- Windows 10, 8, 7, XP, Vista, 2000, NT y 98.
- Distribuciones de Linux basadas en
  - Linux kernel 2.4 y posteriores
  - Ubuntu
  - Debian
  - OpenSUSE
  - Mandriva/Mandrake
  - Fedora
  - RHEL
  - Arch Linux.
- Solaris y OpenSolaris.
- macOS X Server Leopard y Snow Leopard.



- OpenBSD y FreeBSD.
- MS-DOS.
- OS/2.
- QNX.
- BeOS R5.
- Haiku.
- React OS.

#### Mininet

Mininet es un emulador de red de telecomunicaciones de código abierto el cual permite la simulación de switches, routers, enlaces y terminales de una manera sencilla. Este emulador es utilizado con el fin de facilitar la enseñanza al momento de implementar una red como en el caso de la red SDN, y dado a su codificación sencilla e implementación de ejemplos que brinda el emulador se desarrolla un fácil entendimiento para los estudiantes.

Mininet no demanda gran cantidad de recursos por lo que es una herramienta ligera, pero de igual manera permite la manipulación de redes complejas. Adicionalmente, es compatible con varias versiones por lo que facilita el uso de esta herramienta en ambientes educativos y en investigaciones sobre las redes.

Algunas de las características de Mininet se enumeran a continuación:

- **Escalabilidad:** Puede trabajar en una red con cientos de nodos (conmutadores y hosts).
- **Adecuado para investigación y desarrollo:** Mininet es muy popular entre la comunidad de investigadores de todo el mundo. El código que desarrolla y ejecuta en mininet se puede probar en un sistema de red real con cambios mínimos.
- **API de Python:** el código de mininet es casi en su totalidad código de Python. • Admite topologías personalizadas • En comparación con el hardware: económico y reconfigurable.
- Para saber más sobre mininet, navegue a: <https://www.mininet.org/overview/>

#### Equipos requeridos:

- Laptop o PC de versión de 32bits o 64bits
- Conexión a Internet
- Tamaño de memoria RAM entre 3GB y 8GB
- Disco duro virtual libre recomendado entre 10GB y 15 GB





## Instrucciones

### Instalación de Virtual Box

- Entrar en la página oficial de VirtualBox

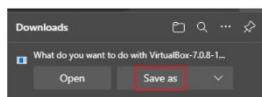
<https://www.virtualbox.org/wiki/Downloads>

- Dar click en Windows Hosts



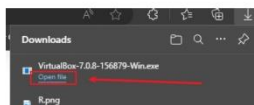
Captura 1: Captura de pantalla de la página de descarga de VirtualBox.

- Se abrirá un mensaje de descarga, dar click en Guardar o Save y esperar que termine la descarga.



Captura 2: Captura para guardar la descarga de VirtualBox.

- Dar click en Open File o Abrir



Captura 3: Descarga de VirtualBox.

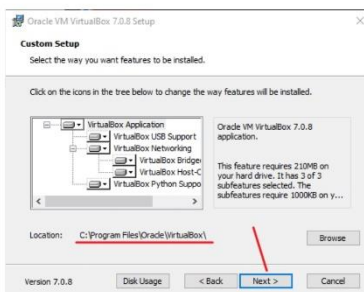
- Saldrá la ventana de instalación y dar click en Next



Captura 4: Ventana de instalación de VirtualBox.

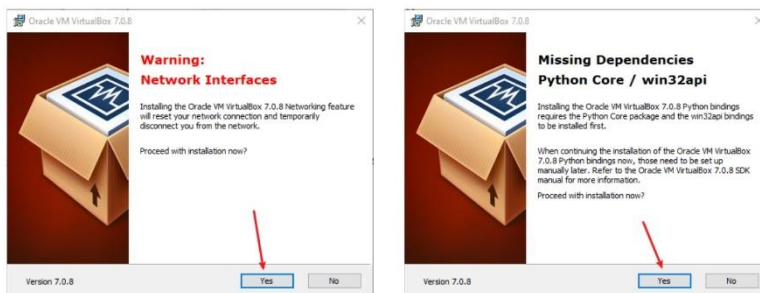


- Se abrirá la pantalla de Setup. Puede elegir si quiere cambiar de ubicación el programa a instalar o dejarlo por Default. En este caso lo dejaremos por Default y damos clic en Next.



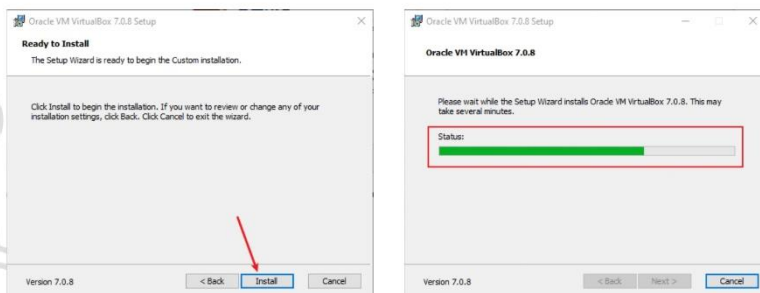
Captura 5: Pantalla de Configuraciones de instalación del VirtualBox.

- Damos click en Yes.

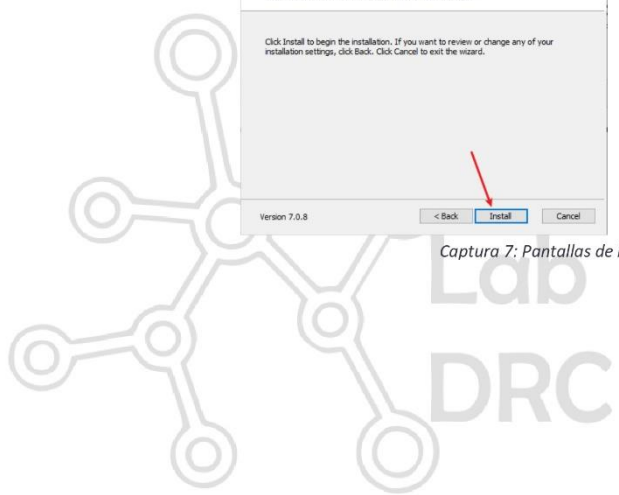


Captura 6: Pantalla de Avisos para proceder a la instalación de VirtualBox.

- Finalmente, damos clic en Install y esperamos que se instale (esto puede tomar unos minutos).



Captura 7: Pantallas de instalación de VirtualBox.



- Una vez finalizada la instalación sale la pantalla de Instalación completa, aquí tenemos una casilla que al dejar activada abrirá automáticamente el programa, en este caso la dejaremos sin marcar y damos clic en Finish.



Captura 8: Pantalla de instalación completa del VirtualBox.

### Creación de máquina virtual con Ubuntu

1. Entrar en el enlace a la página de descarga

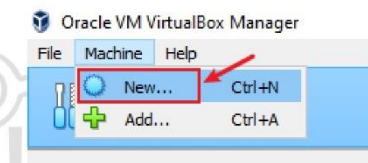
[Ubuntu 20.04.6 LTS \(Focal Fossa\)](#)

2. Descargar la imagen del software del sistema de Ubuntu. Abrir la carpeta en la que se descargó para tenerlo a mano.

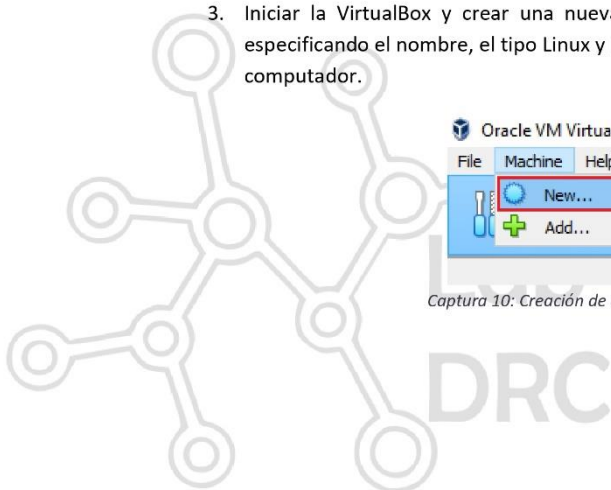


Captura 9: Pantalla de descarga de la página oficial de Ubuntu.

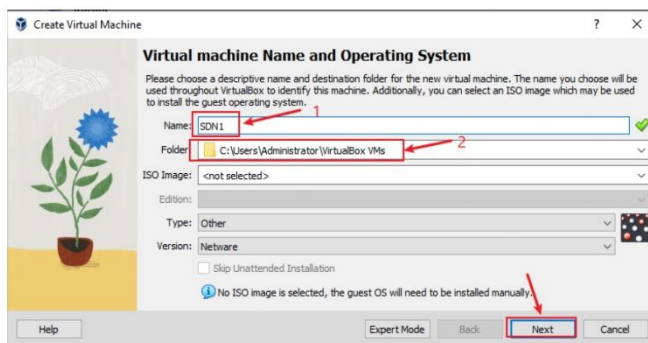
3. Iniciar la VirtualBox y crear una nueva máquina virtual (VM por sus siglas en inglés) especificando el nombre, el tipo Linux y la versión dependiendo de las características de su computador.



Captura 10: Creación de una nueva máquina virtual.

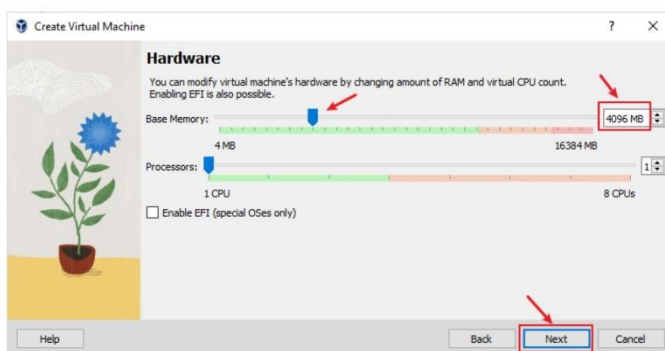


- a. 1. Asignar el nombre sin espacios. 2. Recomendamos dejar la carpeta por default. y clic en Next/Siguiente.



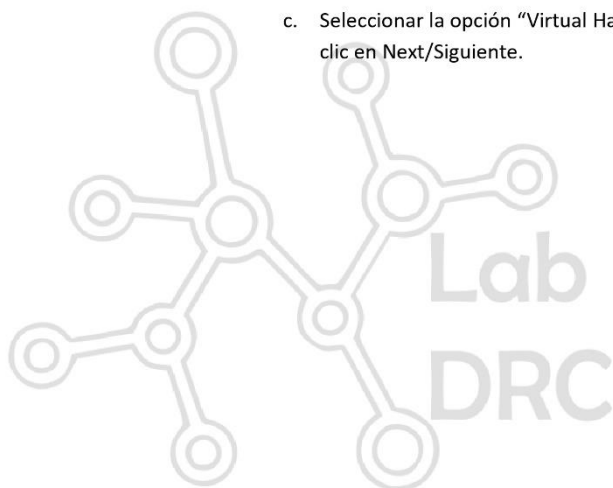
Captura 11: Proceso de la creación de la máquina virtual SDN1.

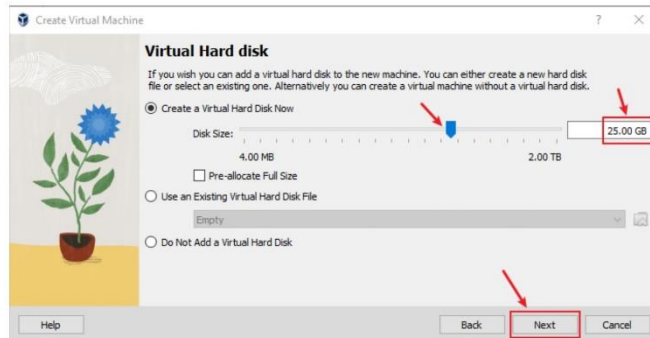
- b. Especificar el tamaño de la memoria RAM. Se recomienda asignar 4096MB de RAM mínimo y luego clic en Next/Siguiente.



Captura 12: Proceso de la creación de la máquina virtual UbuntuSDN1.

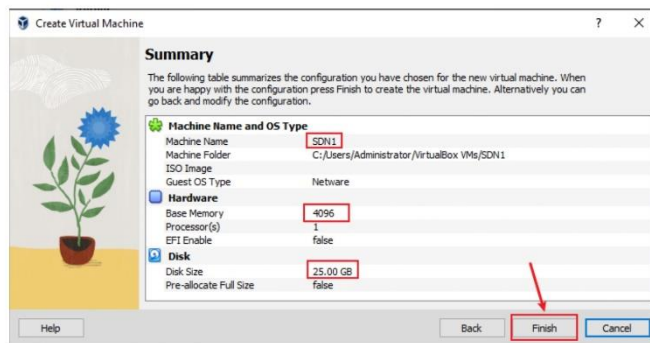
- c. Seleccionar la opción "Virtual Hard Disk" con capacidad entre 10Gb a 25Gb y luego clic en Next/Siguiente.





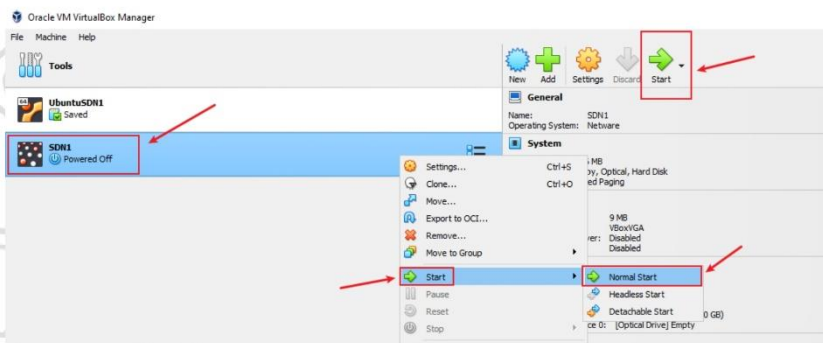
Captura 13: Proceso de la creación de la máquina virtual UbuntuSDN1.

- d. Revisamos y confirmamos que toda la información sea la correcta, damos clic en Finish/Finalizar.



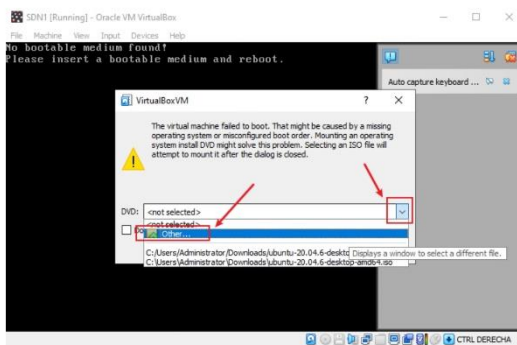
Captura 14: Proceso de la creación de la máquina virtual SDN1.

4. Iniciar la VM, hay dos formas de hacerlo. Seleccionar la VM y dar clic al botón de Start de la derecha o dar clic derecho sobre la VM y seleccionar Start-Normal Start.

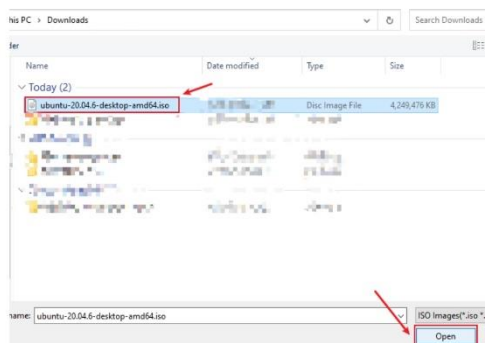


Captura 15: VM creada SDN1.

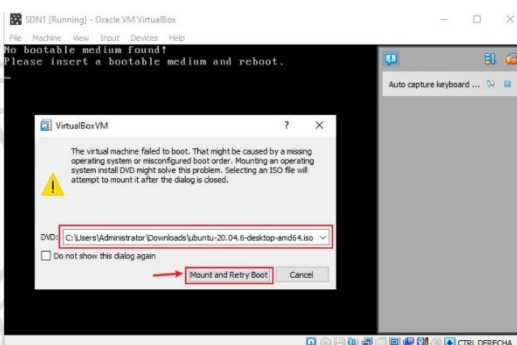
- En la siguiente pantalla nos pedirá elegir el sistema operativo que descargamos en el punto 1. Damos clic en la flecha de la derecha y luego en otros (captura 16), aparecerá una pantalla en la que buscaremos la ruta en que se descargó la imagen ISO del sistema operativo y damos clic en open/abrir (captura 17) para luego dar clic en Mount and Retry Boot (captura 18) para reiniciar la VM.



Captura 16: Pantalla para agregar el sistema operativo a la VM creada.



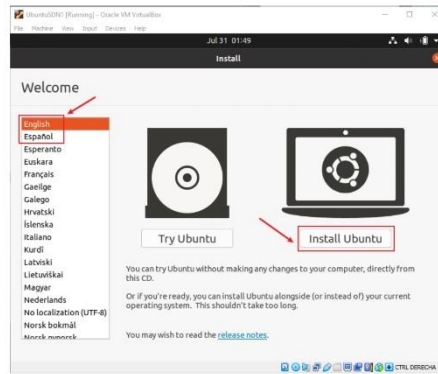
Captura 17: Proceso de selección del sistema operativo descargado en el punto 1.





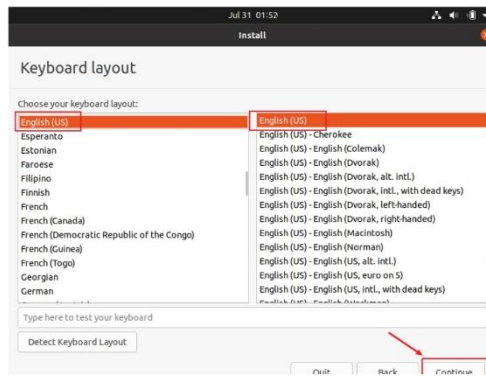
Captura 18: Proceso de selección del sistema operativo descargado en el punto 1.

6. Elegimos el lenguaje de instalación de preferencia y luego damos clic en Install/Instalar Ubuntu.



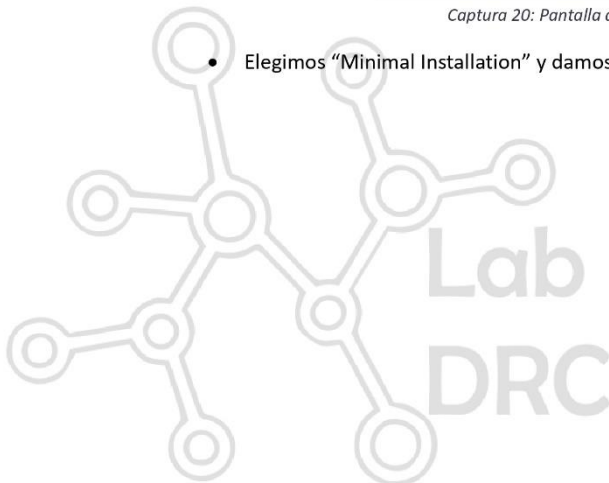
Captura 19: Pantalla de instalación de Ubuntu.

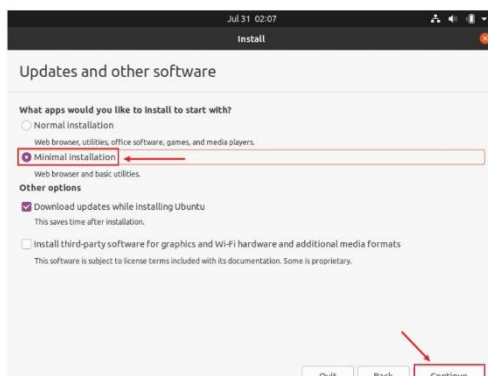
7. Elegimos el lenguaje para el teclado y damos clic en Continue/Continuar.



Captura 20: Pantalla de instalación de Ubuntu.

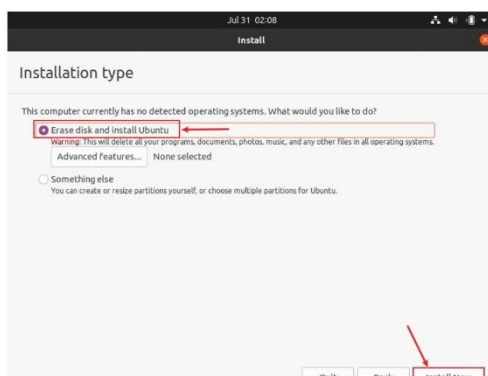
- Elegimos "Minimal Installation" y damos a Continue/Continuar.





Captura 21: Pantalla de update and other software

- Seleccionamos “Erase disk and install Ubuntu” y damos click a Install Now/Instalar ahora.

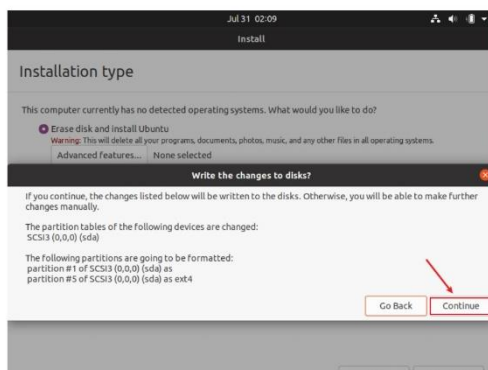


Captura 22: Pantalla de tipo de instalación

- Nos pregunta si queremos escribir los cambios en el disco, damos clic en Continue/Continuar.







Captura 23: Pantalla de escritura en disco

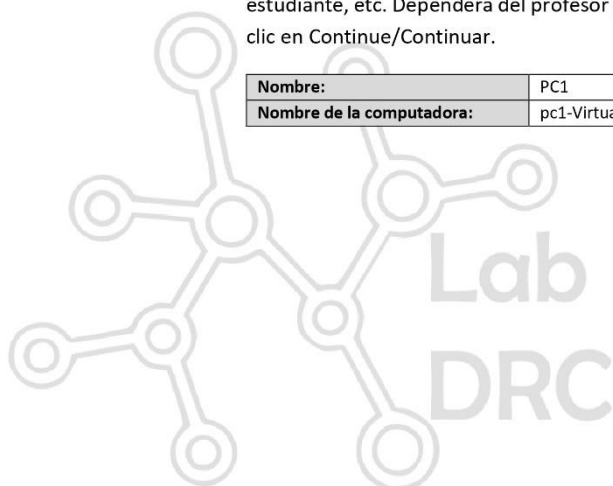
- Seleccionamos la región en la que nos encontramos, en este caso, seleccionamos "Guayaquil" y damos clic en Continue/Continuar.

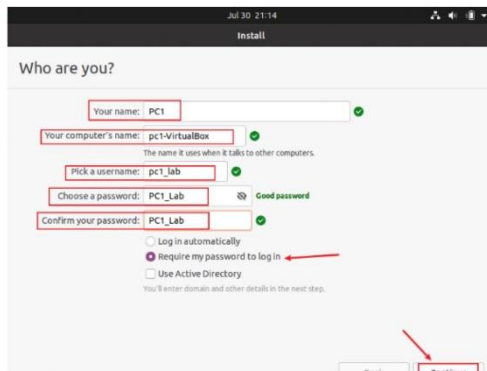


Captura 24: Pantalla de selección de región

- Ahora vamos a crear el nuevo usuario, este puede ser uno genérico, uno por máquina, por estudiante, etc. Dependerá del profesor guía, pero aquí se plantea uno de ejemplo y damos clic en Continue/Continuar.

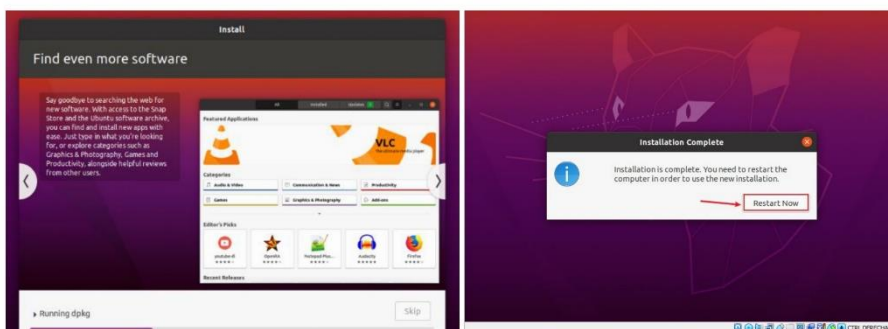
<b>Nombre:</b>	PC1	<b>Nombre de usuario:</b>	pc1_lab
<b>Nombre de la computadora:</b>	pc1-VirtualBox	<b>Contraseña:</b>	PC1_Lab





Captura 25: Registro de usuario para la VM

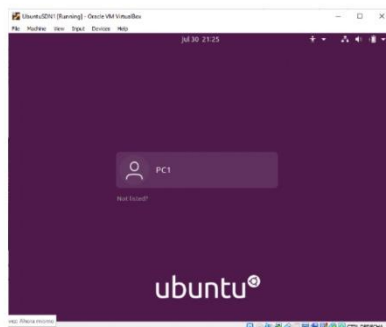
- Esperamos que se instale y le damos a reiniciar VM.



Captura 26: Proceso de instalación de la VM

- Ahora tenemos nuestra máquina virtual con usuario registrado.





Captura 27: Vista de inicio de sesión de la nueva VM con el usuario registrado

### Instalación de Mininet

Mininet ofrece tres formas de obtener el software:

- **Máquina virtual:** Mininet proporciona una máquina virtual preconfigurada con Mininet instalado. Esta es la forma más fácil de comenzar a usar Mininet.
- **Código fuente:** Mininet también está disponible como código fuente. Puedes descargar el código fuente y compilarlo tú mismo. Esta opción te da más control sobre la instalación de Mininet.
- **Paquete:** Mininet también está disponible como paquete. Puedes descargar el paquete y ejecutarlo para instalar Mininet. Esta opción es la más sencilla, pero no te da tanto control sobre la instalación de Mininet como la opción de código fuente.

En este caso, clonaremos el repositorio de git y lo instalaremos manualmente. Esto nos permitirá personalizar la instalación de Mininet según nuestras necesidades.

- Abrir el terminal en la máquina virtual creada.



Captura 28: Terminal en la VM creada.

- Clonamos el repositorio de git con el comando:

**git clone <http://github.com/mininet/mininet>**



```
pci_lab@pci-VirtualBox:~$ git clone http://github.com/mininet/mininet
```

Captura 29: Instalación de mininet clonando el repositorio de git.

En caso de salir un error respecto al comando git no encontrado, procedemos a instalarlo con el comando **sudo apt install git** y procedemos a correr el comando del punto 2.

```
Command 'git' not found, but can be installed with:  
sudo apt install git  
pci_lab@pci-VirtualBox:~$ sudo apt install git  
[sudo] password for pci_lab:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  git-man liberror-perl  
Suggested packages:  
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk  
  nitweb nit-rpc nit-mediaWiki nit-cvs
```

Captura 30: Instalación de git.

```
pci_lab@pci-VirtualBox:~$ git clone http://github.com/mininet/mininet  
Cloning into 'mininet'...  
warning: redirecting to https://github.com/mininet/mininet/  
remote: Enumerating objects: 10347, done.  
remote: Counting objects: 100% (193/193), done.  
remote: Compressing objects: 100% (115/115), done.  
remote: Total 10347 (delta 104), reused 143 (delta 76), pack-reused 10154  
Receiving objects: 100% (10347/10347), 3.33 MiB | 6.31 MiB/s, done.  
Resolving deltas: 100% (6885/6885), done.  
pci_lab@pci-VirtualBox:~$
```

Captura 31: Repositorio clonado.

- Nos ubicamos en la carpeta de mininet usando el comando **cd mininet**, luego usamos el comando **git tag** para verificar que se encuentre la versión 2.2.2 y finalmente **git checkout -b 2.2.2**



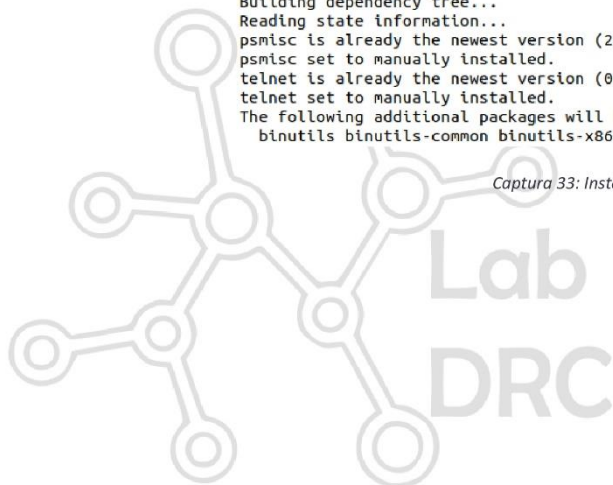
```
pc1_lab@pc1-VirtualBox:~$ cd mininet
pc1_lab@pc1-VirtualBox:~/mininet$ git tag
1.0.0
2.0.0
2.1.0
2.1.0p1
2.1.0p2
2.2.0
2.2.1
2.2.2
2.3.0
2.3.0b1
2.3.0b2
2.3.0d3
2.3.0d4
2.3.0d5
2.3.0d6
2.3.0rc1
2.3.0rc2
2.3.1b2
2.3.1b3
2.3.1b4
cs244-spring-2012-final
pc1_lab@pc1-VirtualBox:~/mininet$ git checkout -b 2.2.2
Switched to a new branch '2.2.2'
pc1_lab@pc1-VirtualBox:~/mininet$
```

Captura 32: Checkout de la versión 2.2.2.

- Después de clonar el repositorio de git, solo queda ejecutar el script **install.sh** para instalar Mininet. La opción **nfv** instalará también el protocolo OpenFlow y Open vSwitch. En caso de dar error el comando solo **util/install.sh -nfv** quiere decir que el firewall de Ubuntu tiene bloqueado el puerto que usa git por lo que deberemos usar https y para ello tenemos el comando **git config --global url."https://github.com/".insteadOf git://github.com/**

```
pc1_lab@pc1-VirtualBox:~/mininet$ git config --global url."https://github.com/"
insteadOf git://github.com/
pc1_lab@pc1-VirtualBox:~/mininet$ util/install.sh -nfv
Detected Linux distribution: Ubuntu 20.04 focal amd64
sys.version_info(major=3, minor=8, micro=10, releaselevel='final', serial=0)
Detected Python (python3) version 3
Installing Mininet dependencies
[sudo] password for pc1_lab:
Reading package lists...
Building dependency tree...
Reading state information...
psmisc is already the newest version (23.3-1).
psmisc set to manually installed.
telnet is already the newest version (0.17-41.2build1).
telnet set to manually installed.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu blt gcc-9 libasan5
```

Captura 33: Instalación de Mininet.



- Mediante el comando **“test pingall”** después de haber creado una topología cualquiera, se puede comprobar el funcionamiento de mininet.

```
root@pc1-VirtualBox:~# sudo mn --switch ovsbr --test pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
*** Stopping 1 controllers
c0
*** Stopping 2 links
..
*** Stopping 1 switches
s1
*** Stopping 2 hosts
h1 h2
*** Done
completed in 0.377 seconds
root@pc1-VirtualBox:~#
```

*Captura 38: Prueba de instalación.*



### Instalación de RYU

El proyecto Ryu es un marco de trabajo para el desarrollo de controladores SDN. Está escrito en Python y es compatible con la mayoría de las versiones de OpenFlow. También soporta NetConf y OF-Config. Ryu es el framework más utilizado en los entornos didácticos debido a su sencillez y facilidad de uso. Es posible implementar aplicaciones SDN basadas en las APIs de Ryu, o crear controladores SDN personalizados con comportamientos nuevos.

Ryu ofrece dos formas de instalación:

- Instalación directa con pip
- Usando el comando:

```
pip install ryu
```

- Instalación manual clonando el repositorio. (En este caso es la que usaremos)
- Usando los comandos:

```
git clone git://github.com/osrg/ryu.git
```

```
cd ryu
```

```
cd ryu
```

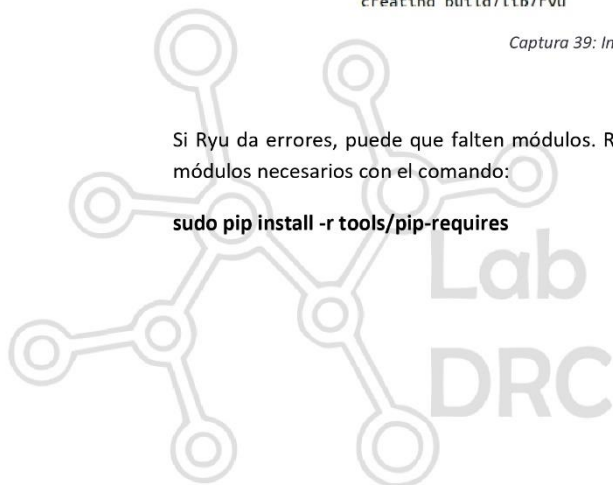
```
python3 ./setup.py install
```

```
pc1_lab@pc1-VirtualBox:~/mininet/ryu$ python3 ./setup.py install
running install
[pbr] Writing ChangeLog
[pbr] Generating ChangeLog
[pbr] ChangeLog complete (0.2s)
[pbr] Generating AUTHORS
[pbr] AUTHORS complete (0.2s)
running build
running build_py
creating build
creating build/lib
creating build/lib/ryu
```

Captura 39: Instalación de ryu.

Si Ryu da errores, puede que falten módulos. Ryu proporciona una herramienta para instalar los módulos necesarios con el comando:

```
sudo pip install -r tools/pip-requirements
```





```

root@pc1-VirtualBox:~/mininet/ryu# sudo pip install -r tools/pip-requirements
Collecting pip==20.3.4
  Downloading pip-20.3.4-py2.py3-none-any.whl (1.5 MB)
    |#####| 1.5 MB 831 kB/s
Collecting eventlet==0.31.1
  Downloading eventlet-0.31.1-py2.py3-none-any.whl (224 kB)
    |#####| 224 kB 6.7 MB/s
Collecting msgpack==0.4.0
  Downloading msgpack-1.0.5-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (322 kB)
    |#####| 322 kB 9.0 MB/s
Collecting netaddr
  Downloading netaddr-0.8.0-py2.py3-none-any.whl (1.9 MB)
    |#####| 1.9 MB 5.5 MB/s
Collecting oslo.config==2.5.0
  Downloading oslo.config-9.2.0-py3-none-any.whl (128 kB)
    |#####| 128 kB 6.7 MB/s
Requirement already satisfied: ovs==2.6.0 in /usr/lib/python3/dist-packages (from -r tools/pip-requirements
(line 9)) (2.13.0)
Collecting packaging==20.9
  Downloading packaging-20.9-py2.py3-none-any.whl (40 kB)
    |#####| 40 kB 1.2 MB/s
Collecting routes
  Downloading Routes-2.5.1-py2.py3-none-any.whl (40 kB)
    |#####| 40 kB 6.5 MB/s
Requirement already satisfied: stx==1.4.0 in /usr/lib/python3/dist-packages (from -r tools/pip-requirements
(line 12)) (1.14.0)
Collecting tinyrpc==1.0.4
  Downloading tinyrpc-1.0.4.tar.gz (26 kB)
Collecting webob==1.2
  Downloading WebOb-1.8.7-py2.py3-none-any.whl (114 kB)
    |#####| 114 kB 9.6 MB/s
Collecting dnspython==2.0.0, >=1.15.0
  Downloading dnspython-1.16.0-py2.py3-none-any.whl (188 kB)
    |#####| 188 kB 10.2 MB/s
Collecting greenlet==0.3

```

Captura 40: Instalación de módulos de ryu.

## Instalación de cURL

cURL es una herramienta para hacer peticiones HTTP. cURL nos ayudará a usar el API REST de las aplicaciones Ryu.

Instalamos usando el comando **sudo apt-get install curl**

```

pc1_lab@pc1-VirtualBox:~/mininet$ sudo apt-get install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl libcurl4
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 161 kB/396 kB of archives.
After this operation, 1.127 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ec.archive.ubuntu.com/ubuntu focal-updates/main amd64 cur
40 0 1ubuntu2 10 [161 kB]

```

Captura 41: Instalación de cURL.

Uso de la opción GET: **curl http://localhost:8080**

Uso de la opción PUT: **curl -X PUT http://localhost:8080/resource**

Uso de la opción POST: **curl -X POST -d '<json>' http://localhost:8080/resource**



### Instalaciones Adicionales

- Si se presenta un problema al momento de utilizar la herramienta de Miniedit o esta mismo no se ejecuta, ejecutar lo siguiente:

Salimos de la carpeta de mininet “**cd ..**”; Luego instalamos un paquete en específico de la página oficial de mininet para que nos permita realizar las acciones de manera correcta dentro de mininet, mediante el comando “**sudo mininet/util/install.sh -a**”

```
Typical install.sh options include:
• -a: install everything that is included in the Mininet VM, including dependencies like Open vSwitch as well the additions like the OpenFlow wireshark dissector and POX. By default these tools will be built in directories created in your home directory.
```

Captura 42: Instalación del paquete específico

- Para el correcto funcionamiento de la implementación de Vlans a la hora de utilizar miniedit se realiza las siguientes instalaciones.

Mediante el comando “**sudo apt-get install bridge-utils**” se instala los paquetes necesarios para la implementación de las vlans en la graficadora Miniedit.

```
root@pc1-VirtualBox:~# sudo apt-get install bridge-utils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Paquetes sugeridos:
  ifupdown
Se instalarán los siguientes paquetes NUEVOS:
  bridge-utils
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 30,5 kB de archivos.
Se utilizarán 112 kB de espacio de disco adicional después de esta operación.
Des:1 http://ec.archive.ubuntu.com/ubuntu focal/main amd64 bridge-utils amd64 1.6-2ubuntu1 [30,5 kB]
Descargados 30,5 kB en 1s (32,2 kB/s)
Seleccionando el paquete bridge-utils previamente no seleccionado.
(Leyendo la base de datos ... 172073 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../bridge-utils_1.6-2ubuntu1_amd64.deb ...
Desempaquetando bridge-utils (1.6-2ubuntu1) ...
Configurando bridge-utils (1.6-2ubuntu1) ...
Procesando disparadores para man-db (2.9.1-1) ...
root@pc1-VirtualBox:~#
```

Captura 43: Instalación del paquete de vlans

Con el comando “**sudo apt-get install vlan**” habilitamos el paquete de instalación específico para las vlans. Y por medio del comando “**sudo modprobe 8021q**” se comprueba que no exista algún error.

```
root@pc1-VirtualBox:~# sudo apt-get install vlan
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
vlan ya está en su versión más reciente (2.0.4ubuntu1.20.04.1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
root@pc1-VirtualBox:~#
```

Captura 44: Instalación de las vlans

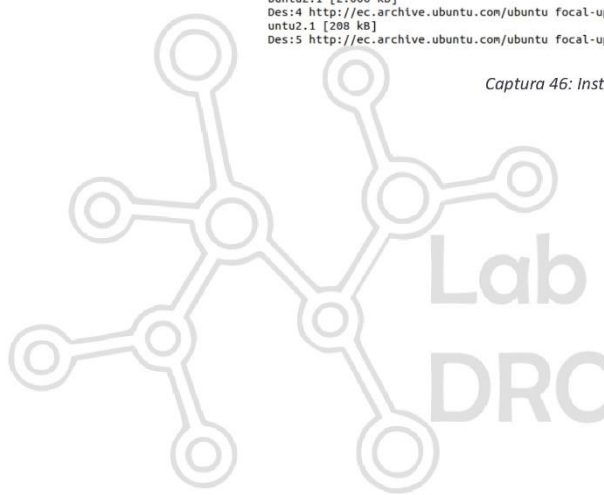
```
root@pc1-VirtualBox:~# sudo modprobe 8021q
root@pc1-VirtualBox:~# █
```

Captura 45: Prueba de funcionamiento

- En caso de no poseer la aplicación de Wireshark, es posible su instalación mediante el comando **"apt wireshark-qt"**.

```
root@pc1-VirtualBox:~# apt install wireshark-qt
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 libbc-ares2 libdouble-conversion3 liblua5.2-0 libpcr2-16-0 libqt5core5a libqt5dbus5 libqt5gui5
 libqt5multimedia5 libqt5multimedia5-plugins libqt5multimedia5-gsttools5 libqt5multimedia5-widgets5
 libqt5network5 libqt5opengl5 libqt5printsupport5 libqt5svg5 libqt5widgets5 libsnm12ldbl libsnappy1v5
 libspandsp2 libssh-gcrypt-4 libwireshark-data libwireshark13 libwireshark10 libwsutil11
 libxcb-xinerama0 libxcb-xinput0 qt5-gtk-platformtheme qttranslations5-l10n wireshark
 wireshark-common
Paquetes sugeridos:
 qt5-image-formats-plugins qtwayland5 snmp-mibs-downloader geopipupdate geopip-database
 geopip-database-extra libjs-leaflet libjs-leaflet.markercluster wireshark-doc
Se instalarán los siguientes paquetes NUEVOS:
 libbc-ares2 libdouble-conversion3 liblua5.2-0 libpcr2-16-0 libqt5core5a libqt5dbus5 libqt5gui5
 libqt5multimedia5 libqt5multimedia5-plugins libqt5multimedia5-gsttools5 libqt5multimedia5-widgets5
 libqt5network5 libqt5opengl5 libqt5printsupport5 libqt5svg5 libqt5widgets5 libsnm12ldbl libsnappy1v5
 libspandsp2 libssh-gcrypt-4 libwireshark-data libwireshark13 libwireshark10 libwsutil11
 libxcb-xinerama0 libxcb-xinput0 qt5-gtk-platformtheme qttranslations5-l10n wireshark
 wireshark-common wireshark-qt
0 actualizados, 31 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 32,9 MB/32,9 MB de archivos.
Se utilizarán 163 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] s
Des:1 http://ec.archive.ubuntu.com/ubuntu focal/universe amd64 libdouble-conversion3 amd64 3.1.5-4ubuntu
1 [37,9 kB]
Des:2 http://ec.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpcr2-16-0 amd64 10.34-7ubuntu0.1
[181 kB]
Des:3 http://ec.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqt5core5a amd64 5.12.8+dfsg-0u
buntu2.1 [2.006 kB]
Des:4 http://ec.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqt5dbus5 amd64 5.12.8+dfsg-0u
buntu2.1 [208 kB]
Des:5 http://ec.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libqt5network5 amd64 5.12.8+dfsg-
```

Captura 46: Instalación de Wireshark



- En caso de no poder ejecutar un archivo Python guardado, ejecutar el comando: **“apt install Python”**

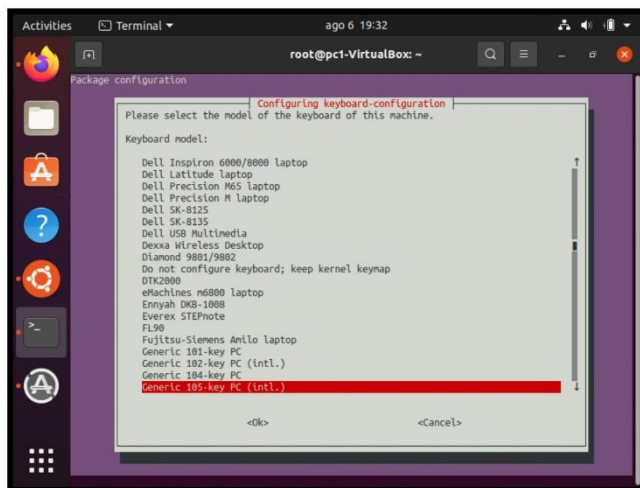
```
root@pc1-VirtualBox:~# apt install python
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Nota, seleccionando «python-is-python2» en lugar de «python»
Se instalarán los siguientes paquetes NUEVOS:
 python-is-python2
0 actualizados, 1 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 2.496 B de archivos.
Se utilizarán 10,2 kB de espacio de disco adicional después de esta operación.
Des:1 http://ec.archive.ubuntu.com/ubuntu focal/universe amd64 python-is-python2 all 2.7.17-4 [2.496 B]
Descargados 2.496 B en 1s (4.235 B/s)
Seleccionando el paquete python-is-python2 previamente no seleccionado.
(Leyendo la base de datos ... 172100 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../python-is-python2_2.7.17-4_all.deb ...
Desempaquetando python-is-python2 (2.7.17-4) ...
Configurando python-is-python2 (2.7.17-4) ...
```

Captura 47: Instalación de Python

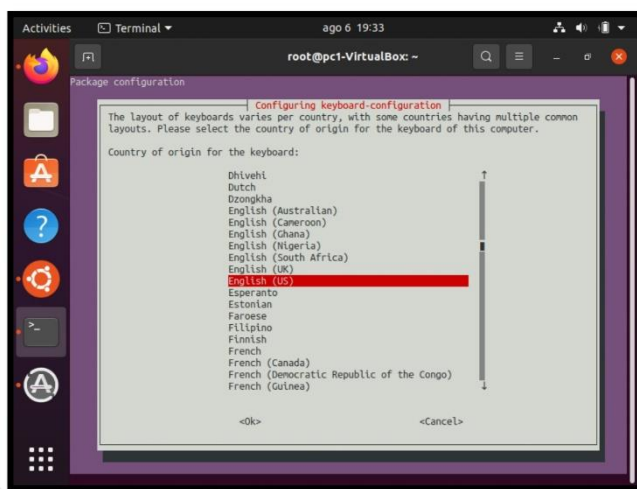


### Configuraciones Adicionales

#### PARA CONFIGURAR EL TECLADO AL IDIOMA PREFERIDO



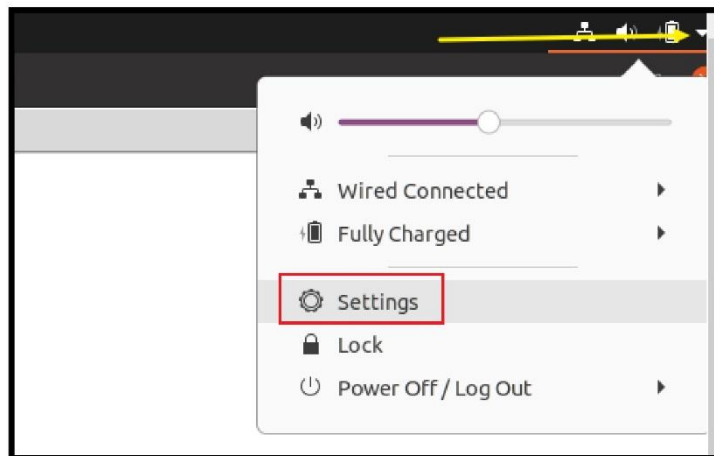
Captura 48: Configuraciones del teclado



Captura 49: Elección del idioma preferido

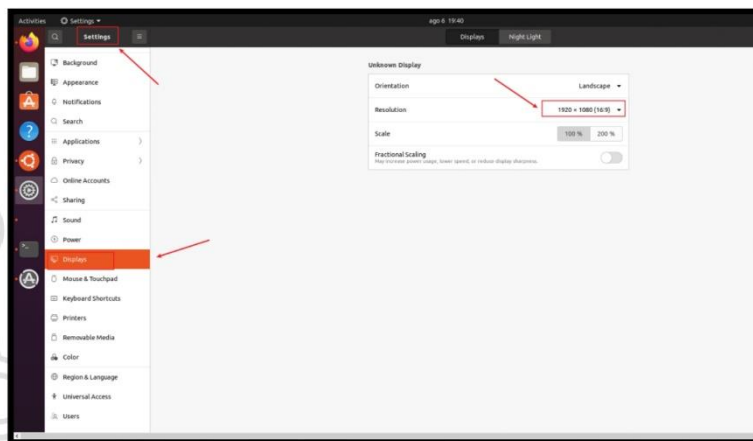
### PARA AMPLIAR EL ÁREA DE TRABAJO DENTRO DE UBUNTU

Dar clic en la flecha de la esquina superior derecha de la pantalla como se muestra en la imagen y seleccionar la opción de configuraciones



Captura 50: Configuraciones de pantalla

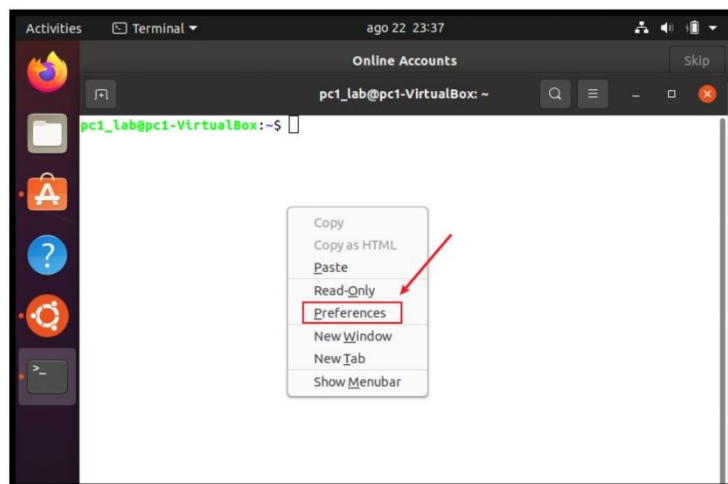
Dentro de la opción de pantalla seleccionar la resolución apropiada.



Captura 51: Especificaciones del tamaño del área de trabajo.

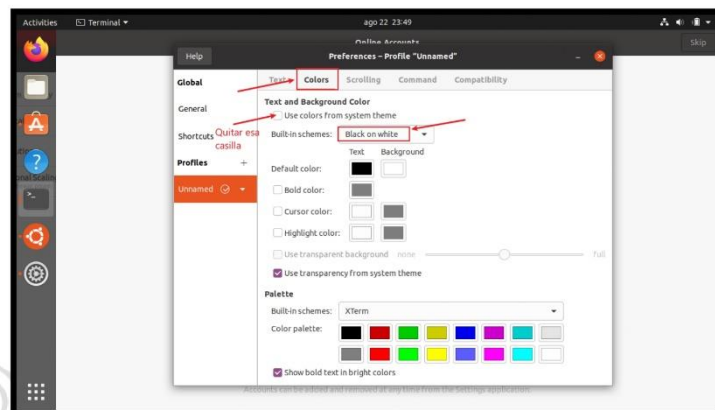
### PARA CAMBIAR DE COLOR LA CONSOLA

Click derecho sobre la consola y seleccionar la opción "Preferences"

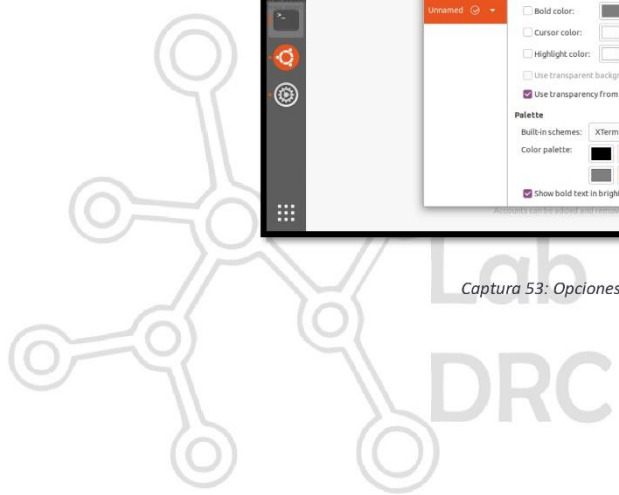


Captura 52: Opción Preferences de la consola

Elegir la opción de colores y seleccionar el color de fondo apropiado.



Captura 53: Opciones de color de la consola





## Anexo 2: Práctica 1

# Práctica #1

## Introducción a las Redes SDN en mininet: topología básica

### Objetivos:

- Implementar una topología básica en la herramienta de Mininet.
- Configurar las respectivas Vlans para separar las distintas áreas en las que trabajan los equipos dentro de la topología.
- Realizar los comandos básicos para el entendimiento de una red SDN básica.

### Marco teórico:

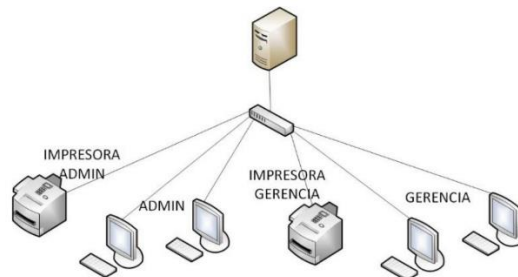


Figura 1: Topología para el escenario "Small Office"

El **escenario de SOHO** (por sus siglas en inglés *Small office/Home Office*) lo podemos observar en la figura 1 y consiste en un entorno de red localizado en una pequeña oficina, donde se interconectan dispositivos periféricos, computadoras, impresoras, etc. Se espera que este proporcione una gestión simplificada y centralizada de la red, permita la segmentación de la red en áreas funcionales distintas y ofrezca servicios de calidad diferenciada.

### Equipos requeridos:

- Laptop o PC
- Conexión a Internet
- Softwares respectivamente instalados
- Mininet en funcionamiento

### Instrucciones

#### Topología de red SDN básica mediante el uso de mininet y minedit

1. Ejecutar la máquina virtual previamente creada en VirtualBox.
2. Dentro de la máquina virtual ejecutar "terminal"



Captura 1: Terminal de la máquina virtual

3. Dentro del "Terminal" se ejecuta los comandos de creación de la topología entrando como administrador (root) mediante el comando **"sudo -i"**. Una vez se ingresa como administrador se introduce la contraseña previamente creada para obtener el acceso.
4. Existen 2 maneras de inicializar una topología. Mediante los comandos del cmd de mininet o mediante la graficadora llamada miniedit.

**a. Mediante comandos de mininet**

Se realizará la topología de la figura 1, para esto se ejecuta el comando **"mn --topo single, 6"** con el fin de representar una topología simple que incluye seis hosts, un controlador SDN y un Switch.

```

root@pci-VirtualBox: ~
pci_lab@pci-VirtualBox:~$ sudo -i
[sudo] password for pci_lab:
root@pci-VirtualBox:~# mn --topo single,6
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
    
```

Entramos en modo root con la clave de la máquina virtual

Usamos ese comando para crear la topología SOHO

Captura 2: Creación de la topología SOHO





b. Mediante la graficadora de miniedit

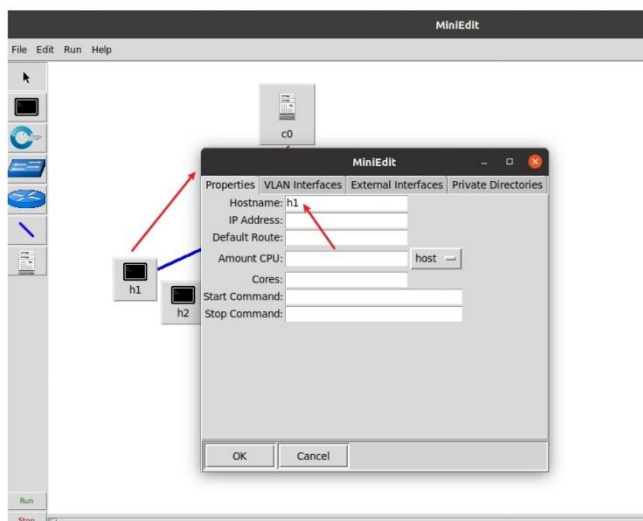
Se realiza la topología de la figura 1 por medio de la graficadora miniedit; para abrirla se ejecuta el comando “*mininet/examples/miniedit.py*”



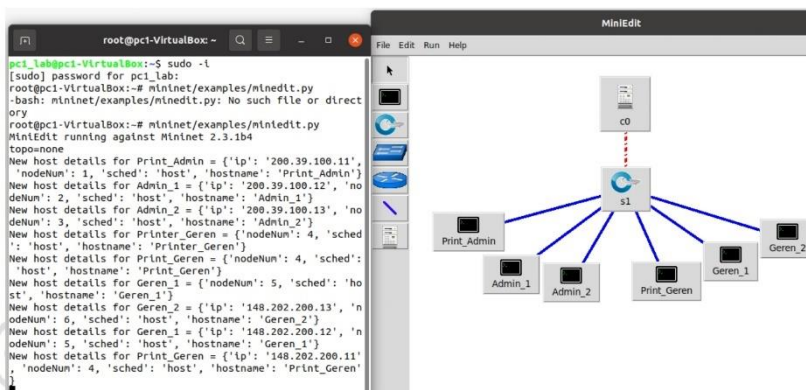
Captura 3: Graficadora miniedit

Dentro de miniedit se implementa la primera topología con los elementos correspondientes. Cada elemento implementado se puede cambiar el respectivo nombre mediante el clic izquierdo sobre el mismo y seleccionando propiedades.





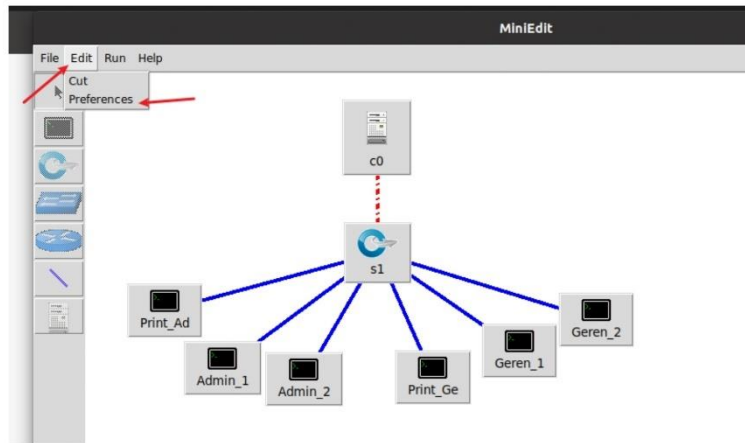
Captura 4: Cambio de nombre de los elementos



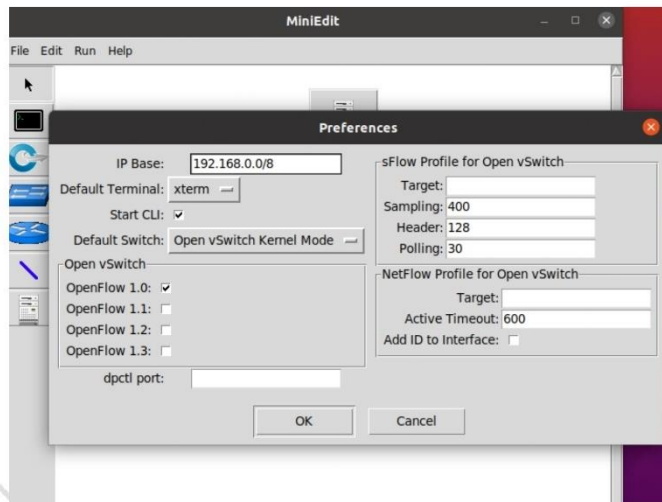
Captura 5: Topología SOHO en miniedit



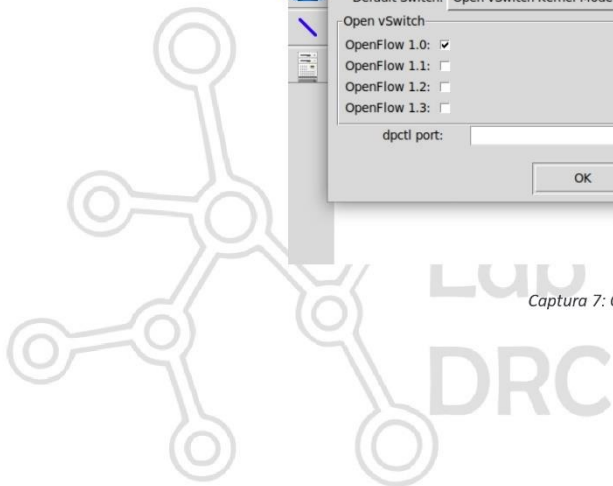
Para habilitar la configuración por comando dentro de la topología realizada en miniedit se abre la opción de "edit" seguido de la opción de preferencias y se habilita el CLI.



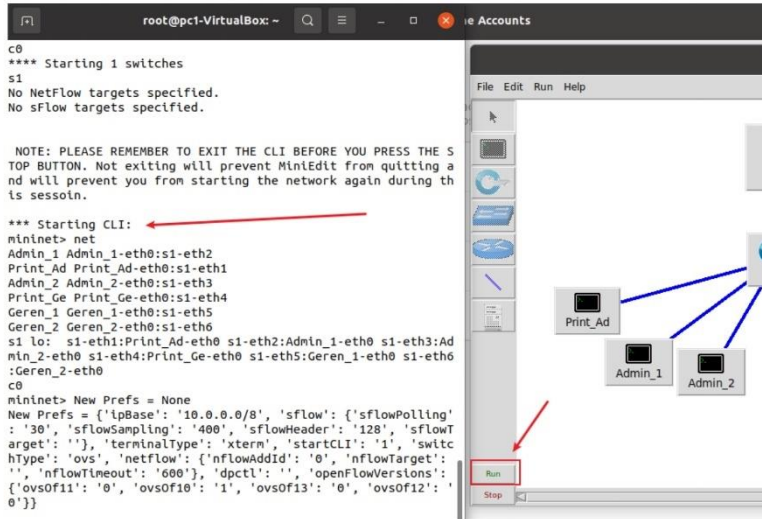
Captura 6: Opción Edit



Captura 7: CLI habilitado

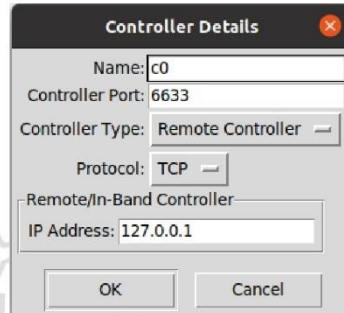


Una vez habilitado el CLI, se puede realizar las pruebas y configuraciones de la topología realizada en miniedit en tiempo real; con solo dar un clic en correr la red.

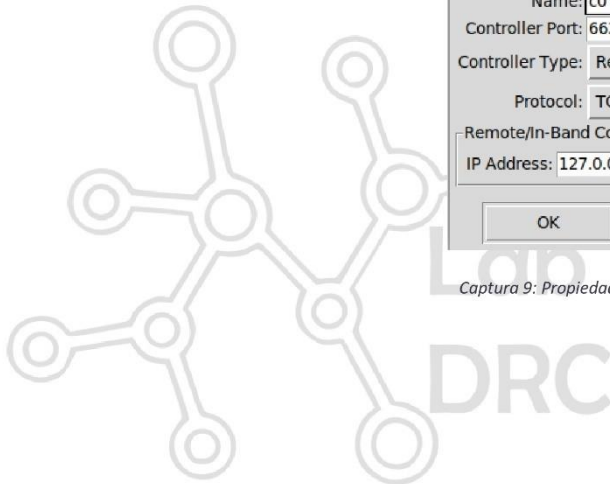


Captura 8: Ejecución de la topología SOHO en miniedit

- Una vez creada la topología SOHO en miniedit se elegirán las siguientes propiedades para el controlador remoto a implementar.



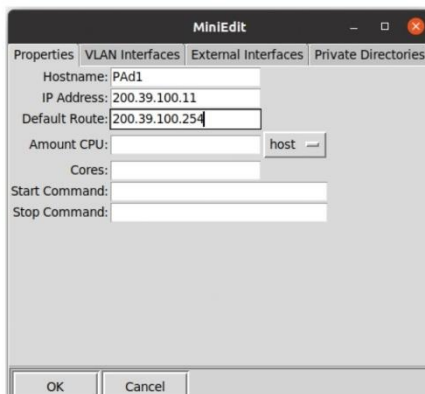
Captura 9: Propiedades para el controlador



6. Para la configuración de los hosts se toma en cuenta la siguiente tabla:

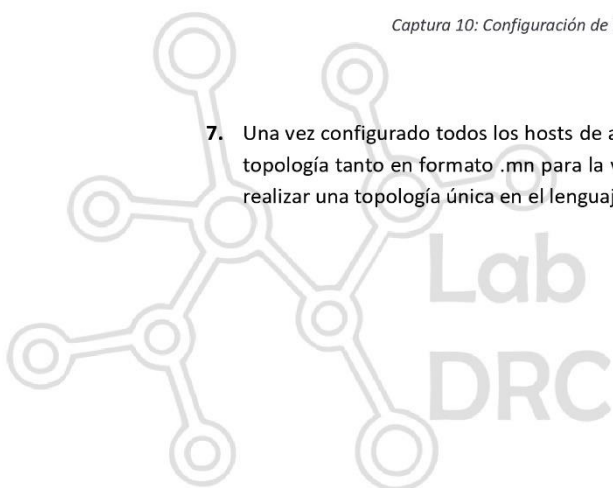
HOST	IP	Default Route	Mask	Vlan
PAd1	200.39.100.1	200.39.100.254	24	100
Ad1	200.39.100.12	200.39.100.254	24	100
Ad2	200.39.100.13	200.39.100.254	24	100
PGe1	148.202.200.11	148.202.200.254	24	200
Ge1	148.202.200.12	148.202.200.254	24	200
Ge2	148.202.200.13	148.202.200.254	24	200

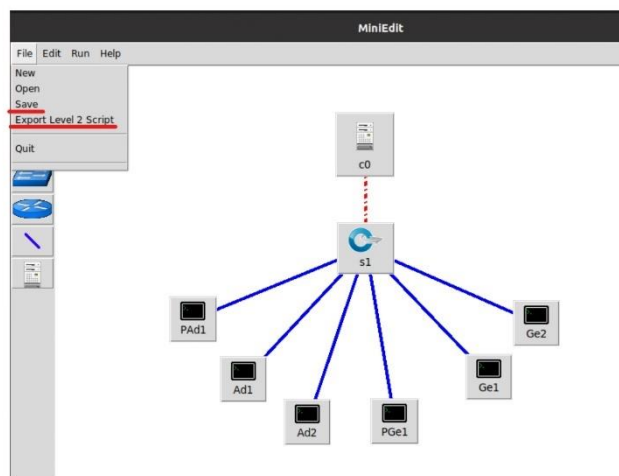
De igual manera se configura en la opción de "propiedades" de cada host.



Captura 10: Configuración de la impresora de administración.

7. Una vez configurado todos los hosts de acuerdo con las IPs correspondientes, se guarda la topología tanto en formato .mn para la visualización en miniedit; y en el formato .py para realizar una topología única en el lenguaje de programación Python.



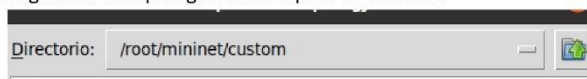


Captura 11: Opciones de guardados.

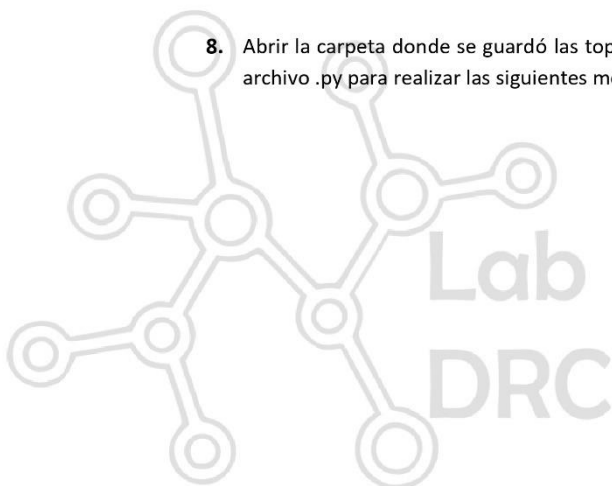
La opción “save” permite guardar la topología en formato .mn con el fin de poder ser ejecutada y visualizada en miniedit.

La opción “export level 2 script” permite guardar la topología en formato .py con el fin de ser editada en el lenguaje de Python y a su vez se pueda ejecutar en la parte de consola.

Se recomienda guardar la topología en la carpeta de mininet.



8. Abrir la carpeta donde se guardó las topologías dentro de la máquina virtual y ejecutar el archivo .py para realizar las siguientes modificaciones por código de programación.



```

info( '*** Adding controller\n' )
c0=net.addController(name='c0',
                    controller=RemoteController,
                    ip='127.0.0.1',
                    protocol='tcp',
                    port=6633)

info( '*** Add switches\n' )
s1 = net.addSwitch('s1', cls=OVSKernelSwitch)

info( '*** Add hosts\n' )
Ad1 = net.addHost('Ad1', cls=Host, defaultRoute='via 200.39.100.254')
PGe1 = net.addHost('PGe1', cls=Host, defaultRoute='via 148.202.200.254')
Ad2 = net.addHost('Ad2', cls=Host, defaultRoute='via 200.39.100.254')
PAd1 = net.addHost('PAd1', cls=Host, defaultRoute='via 200.39.100.254')
Ge2 = net.addHost('Ge2', cls=Host, defaultRoute='via 148.202.200.254')
Ge1 = net.addHost('Ge1', cls=Host, defaultRoute='via 148.202.200.254')

```

Captura 12: Información del controlador, switch y hosts.

```

info( '*** Post configure switches and hosts\n' )
Ad1.cmd('ip addr del 10.0.0.2/8 dev Ad1-eth0')
Ad1.cmd('vconfig add Ad1-eth0 100')
Ad1.cmd('ifconfig Ad1-eth0.100 200.39.100.12/24')
Ad1.cmd('ip route add default via 200.39.100.254')

PGe1.cmd('ip addr del 10.0.0.4/8 dev PGe1-eth0')
PGe1.cmd('vconfig add PGe1-eth0 200')
PGe1.cmd('ifconfig PGe1-eth0.200 148.202.200.11/24')
PGe1.cmd('ip route add default via 148.202.200.254')

Ad2.cmd('ip addr del 10.0.0.3/8 dev Ad2-eth0')
Ad2.cmd('vconfig add Ad2-eth0 100')
Ad2.cmd('ifconfig Ad2-eth0.100 200.39.100.13/24')
Ad2.cmd('ip route add default via 200.39.100.254')

PAd1.cmd('ip addr del 10.0.0.1/8 dev PAd1-eth0')
PAd1.cmd('vconfig add PAd1-eth0 100')
PAd1.cmd('ifconfig PAd1-eth0.100 200.39.100.11/24')
PAd1.cmd('ip route add default via 200.39.100.254')

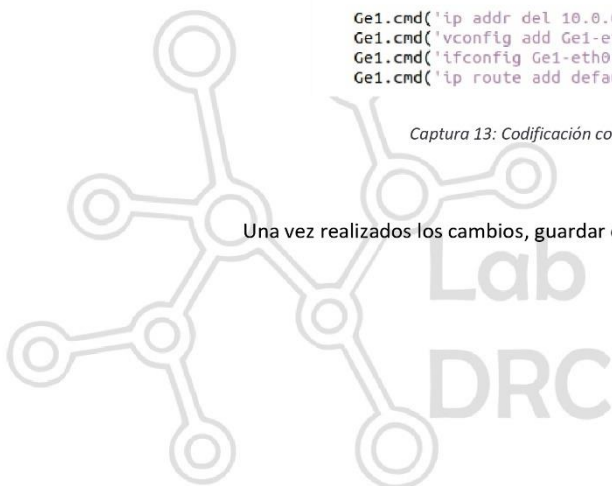
Ge2.cmd('ip addr del 10.0.0.6/8 dev Ge1-eth0')
Ge2.cmd('vconfig add Ge2-eth0 200')
Ge2.cmd('ifconfig Ge2-eth0.200 148.202.200.13/24')
Ge2.cmd('ip route add default via 148.202.200.254')

Ge1.cmd('ip addr del 10.0.0.5/8 dev Ge1-eth0')
Ge1.cmd('vconfig add Ge1-eth0 200')
Ge1.cmd('ifconfig Ge1-eth0.200 148.202.200.12/24')
Ge1.cmd('ip route add default via 148.202.200.254')

```

Captura 13: Codificación complementaria de IPs y VLANs

Una vez realizados los cambios, guardar el script.





9. Abrir en consola la topología guardada con el nombre correspondiente entrando a la carpeta donde se encuentra y utilizar el comando `“sudo python (nombre).py”`. Para verificar que las configuraciones sean correctas se utiliza el comando `“net”` donde se observa cómo las conexiones de los elementos dentro de la red SDN y los puertos que se utilizan. A su vez mediante el comando `“ifconfig”` se observa la dirección IP y máscara que posee el host; esto se debe cambiar para la introducción de las VLANs con el fin de separar el área de administración del área de gerencia dentro de la topología SOHO.

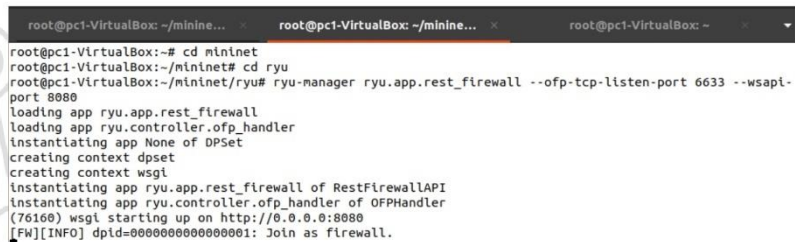
```

root@pc1-VirtualBox:~# cd mininet
root@pc1-VirtualBox:~/mininet# cd custom
root@pc1-VirtualBox:~/mininet/custom# sudo python SOHO.py
*** Adding controller
*** Add swlitches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
Ad1 PGe1 Ad2 PAD1 Ge2 Ge1
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> net
Ad1 Ad1-eth0:s1-eth2
PGe1 PGe1-eth0:s1-eth4
Ad2 Ad2-eth0:s1-eth3
PAD1 PAD1-eth0:s1-eth1
Ge2 Ge2-eth0:s1-eth0
Ge1 Ge1-eth0:s1-eth5
s1 lo: s1-eth1:PAD1-eth0 s1-eth2:Ad1-eth0 s1-eth3:Ad2-eth0 s1-eth4:PGe1-eth0 s1-eth5:Ge1-eth0 s1-eth6:G
e2-eth0
c0
mininet> PAD1 ifconfig
PAD1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

```

Captura 14: Topología SOHO en consola

10. Abrir dos terminales más, uno para la implementación del controlador remoto de `“ryu”`; y el otro para la configuración de este. Para esto, abrir la carpeta de mininet `“cd mininet”` y también la carpeta de ryu `“cd ryu”`, seguido habilitar el firewall del controlador remoto con el comando `“ryu-manager ryu.app.rest_firewall --ofp-tcp-listen-port 6633 --wsapi-port 8080”` para que este habilite los puertos que el controlador utiliza.



```

root@pc1-VirtualBox:~/mininet# cd ryu
root@pc1-VirtualBox:~/mininet/ryu# ryu-manager ryu.app.rest_firewall --ofp-tcp-listen-port 6633 --wsapi-
port 8080
loading app ryu.app.rest_firewall
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.rest_firewall of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(76160) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=0000000000000001: Join as firewall.

```

Captura 15: Firewall del controlador ryu





11. En el tercer terminal, habilitar la dirección junto al puerto con el que trabaja el controlador remoto mediante el uso del comando cURL previamente instalado en la máquina virtual "curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000001"; seguido habilitar el firewall en el switch para la designación de la red junto a la VLAN correspondiente, con los comandos:

"curl -X POST -d '{"nw\_src": "200.39.100.0/24", "nw\_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/100" (VLAN 100)

"curl -X POST -d '{"nw\_src": "148.202.200.0/24", "nw\_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/200" (VLAN 200)

```

root@pc1-VirtualBox: ~/minine... root@pc1-VirtualBox: ~/minine... root@pc1-VirtualBox: ~
root@pc1-VirtualBox:~# curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000001
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": "firewall running."}}]
root@pc1-VirtualBox:~# curl -X POST -d '{"nw_src": "200.39.100.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/100
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1", "vlan_id": 100}]}]
root@pc1-VirtualBox:~# curl -X POST -d '{"nw_src": "148.202.200.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/200
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1", "vlan_id": 200}]}]
root@pc1-VirtualBox:~# █
    
```

Captura 16: Configuración del controlador remoto

```

(76160) accepted ('127.0.0.1', 38644)
127.0.0.1 - - [07/Sep/2023 19:59:39] "PUT /firewall/module/enable/0000000000000001 HTTP/1.1" 200 217 0.002676
(76160) accepted ('127.0.0.1', 55964)
127.0.0.1 - - [07/Sep/2023 19:59:53] "POST /firewall/rules/0000000000000001/100 HTTP/1.1" 200 241 0.000670
(76160) accepted ('127.0.0.1', 40870)
127.0.0.1 - - [07/Sep/2023 20:00:03] "POST /firewall/rules/0000000000000001/200 HTTP/1.1" 200 241 0.000623
█
    
```

Captura 17: Vlans habilitadas



12. Mediante el comando **“ping”** se puede comprobar la conexión entre los dispositivos de la red y a su vez cuales no se comunican entre sí. Por otro lado, si se desea realizar ping dentro de un host específico se abre el cmd de dicho host por medio del comando **“xterm”**.

Nota: Para realizar ping se debe escribir la dirección ip del destino.

```
mininet> PAd1 ping 200.39.100.12
PING 200.39.100.12 (200.39.100.12) 56(84) bytes of data.
64 bytes from 200.39.100.12: icmp_seq=1 ttl=64 time=0.683 ms
64 bytes from 200.39.100.12: icmp_seq=2 ttl=64 time=0.096 ms
64 bytes from 200.39.100.12: icmp_seq=3 ttl=64 time=0.092 ms
64 bytes from 200.39.100.12: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 200.39.100.12: icmp_seq=5 ttl=64 time=0.098 ms
64 bytes from 200.39.100.12: icmp_seq=6 ttl=64 time=0.174 ms
64 bytes from 200.39.100.12: icmp_seq=7 ttl=64 time=0.088 ms
^C
--- 200.39.100.12 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6132ms
rtt min/avg/max/mdev = 0.088/0.189/0.683/0.203 ms
mininet> PAd1 ping 148.202.200.11
PING 148.202.200.11 (148.202.200.11) 56(84) bytes of data.
From 200.39.100.11 icmp_seq=1 Destination Host Unreachable
From 200.39.100.11 icmp_seq=2 Destination Host Unreachable
From 200.39.100.11 icmp_seq=3 Destination Host Unreachable
^C
--- 148.202.200.11 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5100ms
pipe 4
mininet> █
```

Captura 18: Pruebas de ping

13. Mediante los comandos **“nodes”** se obtiene la información de los nodos conectados entre sí y con el comando **“dump”** la respectiva información de estos.
14. Para visualizar los paquetes que se envían en tiempo real, se utiliza el comando **“xterm”** en uno de los hosts y luego se ejecuta el comando **“wireshark”**.



Captura 19: Comando xterm

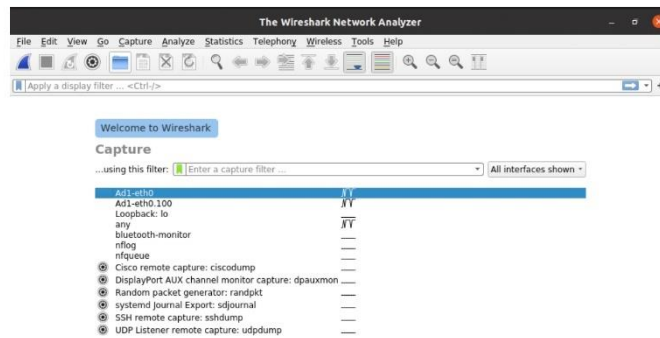
```

"Node: PAD1"
34 bytes from 200.39.100.12: icmp_seq=69 ttl=64 time=0,055 ms
34 bytes from 200.39.100.12: icmp_seq=70 ttl=64 time=0,059 ms
34 bytes from 200.39.100.12: icmp_seq=71 ttl=64 time=0,053 ms
34 bytes from 200.39.100.12: icmp_seq=72 ttl=64 time=0,055 ms
34 bytes from 200.39.100.12: icmp_seq=73 ttl=64 time=0,069 ms
34 bytes from 200.39.100.12: icmp_seq=74 ttl=64 time=0,053 ms
34 bytes from 200.39.100.12: icmp_seq=75 ttl=64 time=0,054 ms

"Node: Ad1"
root@pci-VirtualBox:~/mininet/custom# wireshark
StandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'

```

Captura 20: Comunicación entre los hosts



Captura 21: Interfaz Wireshark



Capturing from Ad1-eth0.100

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
19	8.194188383	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=218/55888, ttl=...
20	8.194295992	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=218/55888, ttl=...
21	9.216126819	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=219/56864, ttl=...
22	9.216148551	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=219/56864, ttl=...
23	10.240177464	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=220/56320, ttl=...
24	10.240192690	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=220/56320, ttl=...
25	11.264050359	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=221/56576, ttl=...
26	11.264078693	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=221/56576, ttl=...
27	12.287961198	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=222/56832, ttl=...
28	12.287976945	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=222/56832, ttl=...
29	13.312628223	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=223/57888, ttl=...
30	13.312634899	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=223/57888, ttl=...
31	14.336088866	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=224/57344, ttl=...
32	14.336022815	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=224/57344, ttl=...
33	14.368605150	52:95:d8:de:f9:61	22:d1:da:c6:09:8e	ARP	42	who has 200.39.100.11? Tell 200.39.100.12
34	14.368728369	22:d1:da:c6:09:8e	52:95:d8:de:f9:61	ARP	42	200.39.100.11 is at 22:d1:da:c6:09:8e
35	15.36004562	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=225/57600, ttl=...
36	15.360049587	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=225/57600, ttl=...
37	16.383991972	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=226/57856, ttl=...
38	16.383978321	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=226/57856, ttl=...
39	17.408026208	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=227/58112, ttl=...
40	17.408035221	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=227/58112, ttl=...
41	18.432110762	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=228/58368, ttl=...
42	18.432141278	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=228/58368, ttl=...
43	19.456092994	200.39.100.11	200.39.100.12	ICMP	98	Echo (ping) request id=0x29f4, seq=229/58624, ttl=...
44	19.456016885	200.39.100.12	200.39.100.11	ICMP	98	Echo (ping) reply id=0x29f4, seq=229/58624, ttl=...

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface Ad1-eth0.100, id 0  
 Ethernet II, Src: 22:d1:da:c6:09:8e [22:d1:da:c6:09:8e], Dst: 52:95:d8:de:f9:61 [52:95:d8:de:f9:61]  
 Internet Protocol Version 4, Src: 200.39.100.11, Dst: 200.39.100.12  
 Internet Control Message Protocol

```

0000 52 95 d5 d6 f9 61 22 d1 da c6 09 8e 08 00 45 00 R...a...1...E
0010 00 54 45 c9 40 00 40 01 9c 82 c8 27 64 0b c8 27 TE @ @ ...d '
0020 64 9c 08 00 c5 2f 29 f4 00 42 86 74 fa 64 00 00 d - / ) ...t d-
0030 00 00 bf 5d 01 00 00 00 00 00 10 11 12 13 14 15 ...] ... *%&
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 ...] ... *%&
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &( ) * , . / 0 1 2 3 4 5
0060 36 37 67
  
```

Captura 22: Análisis de paquetes



## Anexo 3: Práctica 2

# Práctica #2

## Manipulación de elementos de red SDN: escenario centro educativo pequeño

### Objetivos:

- Implementar una topología compleja en la herramienta de Mininet.
- Configurar las respectivas Vlans para separar las distintas áreas en las que trabajan los equipos dentro de la topología.
- Realizar los análisis de red SDN para comprobar el correcto funcionamiento de la red SDN.

### Marco teórico:

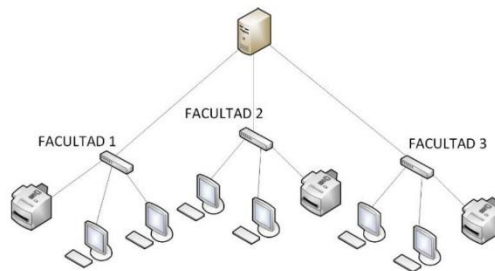


Figura 1: Topología para el escenario "Centro educativo pequeño"

En la figura 1 podemos ver la topología para el escenario de red de un centro educativo pequeño implica la implementación de una red a mayor escala que el anterior escenario, en donde se interconectan aulas, bibliotecas, laboratorios, salas comunes y otros espacios educativos. Aquí, el prototipo de red SDN debe ofrecer una administración centralizada de la red, permitir la implementación de políticas de acceso a Internet y proporcionar un control eficiente del tráfico de red para garantizar una experiencia de aprendizaje óptima.

### Equipos requeridos:

- Laptop o PC
- Conexión a Internet
- Softwares respectivamente instalados
- Mininet en funcionamiento

### Instrucciones

#### Ejercicio 1: Topología de red SDN básica mediante el uso del cmd de mininet

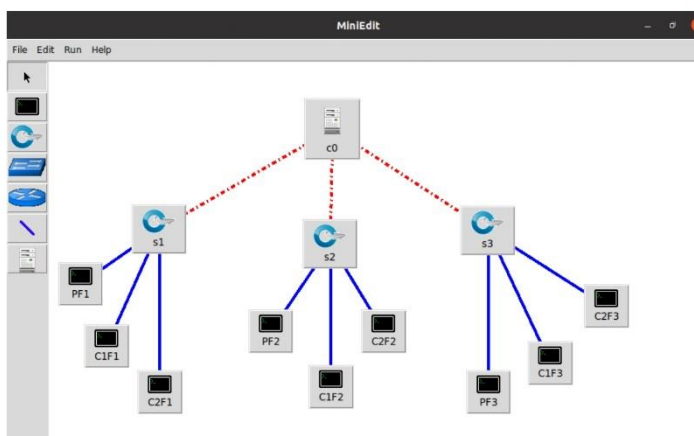
1. Ejecutar la máquina virtual previamente creada en VirtualBox.

- Dentro de la máquina virtual ejecutar "terminal"



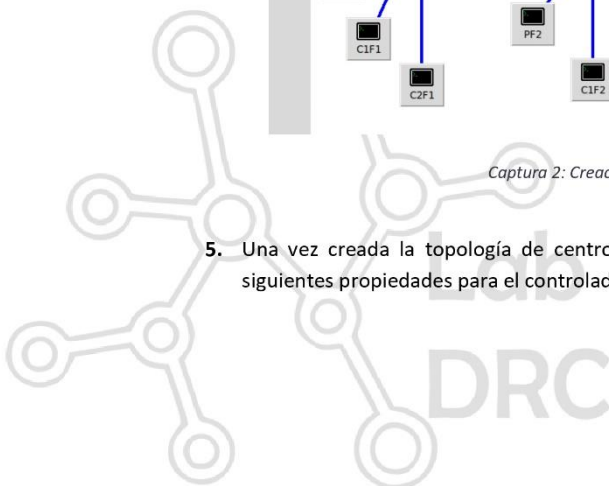
Captura 1: Terminal de la máquina virtual

- Dentro del "Terminal" se ejecuta los comandos de creación de la topología entrando como administrador (root) mediante el comando "sudo -i". Una vez se ingresa como administrador se introduce la contraseña previamente creada para obtener el acceso.
- Abrir miniedit y crear la topología de la figura 1 con las respectivas especificaciones, mediante el comando "mininet/examples/miniedit.py"

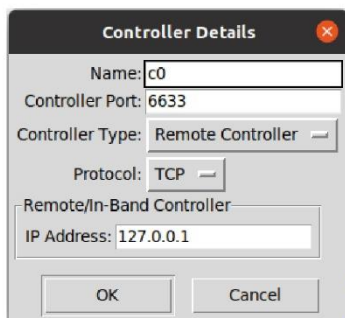


Captura 2: Creación de la topología

- Una vez creada la topología de centro educativo pequeño en miniedit se elegirán las siguientes propiedades para el controlador remoto a implementar.





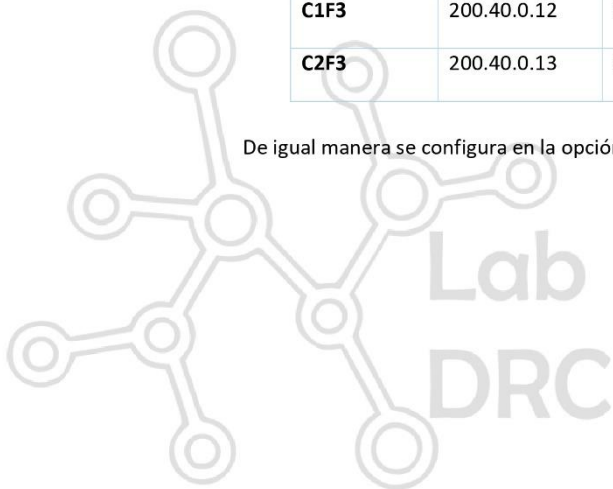


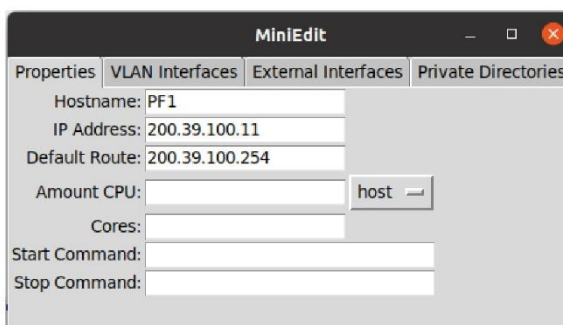
Captura 3: Propiedades para el controlador

6. Para la configuración de los hosts se toma en cuenta la siguiente tabla:

HOST	IP	Default Route	Mask	Vlan
PF1	200.39.100.11	200.39.100.254	24	100
C1F1	200.39.100.12	200.39.100.254	24	100
C2F1	200.39.100.13	200.39.100.254	24	100
PF2	200.39.200.11	200.39.200.254	24	200
C1F2	200.39.200.12	200.39.200.254	24	200
C2F2	200.39.200.13	200.39.200.254	24	200
PF3	200.40.0.11	200.40.0.254	24	300
C1F3	200.40.0.12	200.40.0.254	24	300
C2F3	200.40.0.13	200.40.0.254	24	300

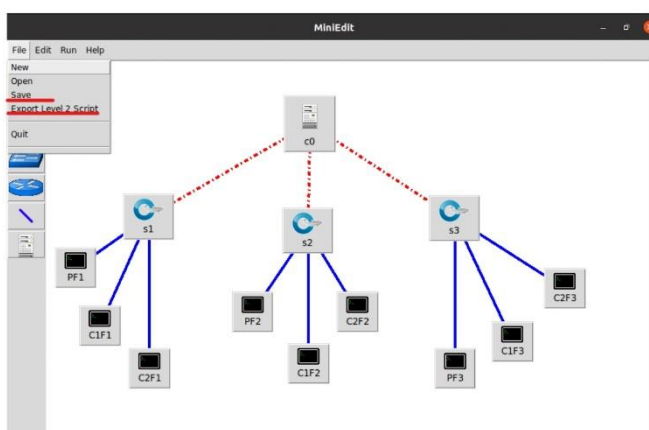
De igual manera se configura en la opción de "propiedades" de cada host.





Captura 4: Configuración de la impresora de la facultad 1.

- Una vez configurado todos los hosts de acuerdo con las IPs correspondientes, se guarda la topología tanto en formato .mn para la visualización en miniedit; y en el formato .py para realizar una topología única en el lenguaje de programación Python.

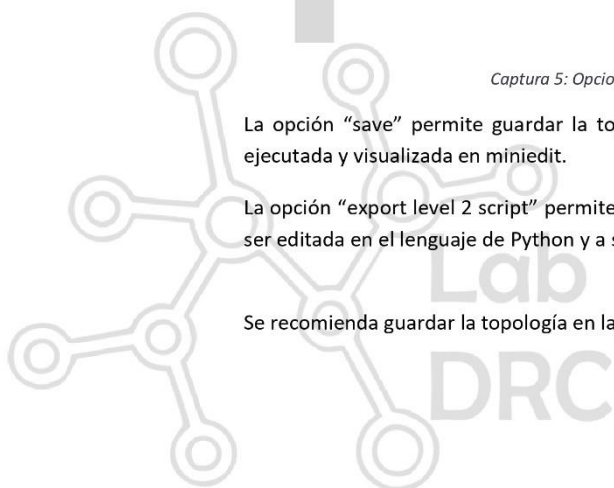


Captura 5: Opciones de guardados.

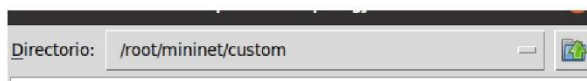
La opción "save" permite guardar la topología en formato .mn con el fin de poder ser ejecutada y visualizada en miniedit.

La opción "export level 2 script" permite guardar la topología en formato .py con el fin de ser editada en el lenguaje de Python y a su vez se pueda ejecutar en la parte de consola.

Se recomienda guardar la topología en la carpeta de mininet.







8. Abrir la carpeta donde se guardó las topologías dentro de la máquina virtual y ejecutar el archivo .py para realizar las siguientes modificaciones por código de programación.

```

info( '*** Adding controller\n' )
c0=net.addController(name='c0',
                    controller=RemoteController,
                    ip='127.0.0.1',
                    protocol='tcp',
                    port=6633)

info( '*** Add switches\n' )
s2 = net.addSwitch('s2', cls=OVSKernelSwitch)
s3 = net.addSwitch('s3', cls=OVSKernelSwitch)
s1 = net.addSwitch('s1', cls=OVSKernelSwitch)

info( '*** Add hosts\n' )
PF3 = net.addHost('PF3', cls=Host, defaultRoute='via 200.40.0.254')
C2F2 = net.addHost('C2F2', cls=Host, defaultRoute='via 200.39.200.254')
C1F2 = net.addHost('C1F2', cls=Host, defaultRoute='via 200.39.200.254')
PF2 = net.addHost('PF2', cls=Host, defaultRoute='via 200.39.200.254')
C2F1 = net.addHost('C2F1', cls=Host, defaultRoute='via 200.39.100.254')
C1F1 = net.addHost('C1F1', cls=Host, defaultRoute='via 200.39.100.254')
PF1 = net.addHost('PF1', cls=Host, defaultRoute='via 200.39.100.254')
C2F3 = net.addHost('C2F3', cls=Host, defaultRoute='via 200.40.0.254')
C1F3 = net.addHost('C1F3', cls=Host, defaultRoute='via 200.40.0.254')
    
```

Captura 6: Información del controlador, switch y hosts.



```

Info( '*** Post configure switches and hosts\n')
PF3.cmd('ip addr del 10.0.0.7/8 dev PF3-eth0')
PF3.cmd('vconfig add PF3-eth0 300')
PF3.cmd('ifconfig PF3-eth0.300 200.40.0.11/24')
PF3.cmd('ip route add default via 200.40.0.254')

C2F2.cmd('ip addr del 10.0.0.6/8 dev C2F2-eth0')
C2F2.cmd('vconfig add C2F2-eth0 200')
C2F2.cmd('ifconfig C2F2-eth0.200 200.39.200.13/24')
C2F2.cmd('ip route add default via 200.39.200.254')

C1F2.cmd('ip addr del 10.0.0.5/8 dev C1F2-eth0')
C1F2.cmd('vconfig add C1F2-eth0 200')
C1F2.cmd('ifconfig C1F2-eth0.200 200.39.200.12/24')
C1F2.cmd('ip route add default via 200.39.200.254')

PF2.cmd('ip addr del 10.0.0.4/8 dev PF2-eth0')
PF2.cmd('vconfig add PF2-eth0 200')
PF2.cmd('ifconfig PF2-eth0.200 200.39.200.11/24')
PF2.cmd('ip route add default via 200.39.200.254')

C2F1.cmd('ip addr del 10.0.0.3/8 dev C2F1-eth0')
C2F1.cmd('vconfig add C2F1-eth0 100')
C2F1.cmd('ifconfig C2F1-eth0.100 200.39.100.13/24')
C2F1.cmd('ip route add default via 200.39.100.254')

C1F1.cmd('ip addr del 10.0.0.2/8 dev C1F1-eth0')
C1F1.cmd('vconfig add C1F1-eth0 100')
C1F1.cmd('ifconfig C1F1-eth0.100 200.39.100.12/24')
C1F1.cmd('ip route add default via 200.39.100.254')

PF1.cmd('ip addr del 10.0.0.1/8 dev PF1-eth0')
PF1.cmd('vconfig add PF1-eth0 100')
PF1.cmd('ifconfig PF1-eth0.100 200.39.100.11/24')
PF1.cmd('ip route add default via 200.39.100.254')

C2F3.cmd('ip addr del 10.0.0.9/8 dev C2F3-eth0')
C2F3.cmd('vconfig add C2F3-eth0 300')
C2F3.cmd('ifconfig C2F3-eth0.300 200.40.0.13/24')
C2F3.cmd('ip route add default via 200.40.0.254')

C1F3.cmd('ip addr del 10.0.0.8/8 dev C1F3-eth0')
C1F3.cmd('vconfig add C1F3-eth0 300')
C1F3.cmd('ifconfig C1F3-eth0.300 200.40.0.12/24')
C1F3.cmd('ip route add default via 200.40.0.254')

```

Captura 7: Codificación complementaria de IPs y VLANs

Una vez realizados los cambios, guardar el script.

9. Abrir en consola la topología guardada con el nombre correspondiente entrando a la carpeta donde se encuentra y utilizar el comando ***“sudo python (nombre).py”***. Para verificar que las configuraciones sean correctas se utiliza el comando ***“net”*** donde se observa cómo las conexiones de los elementos dentro de la red SDN y los puertos que se utilizan. A su vez mediante el comando ***“ifconfig”*** se observa la dirección IP y máscara que posee el host; esto se debe cambiar para la introducción de las VLANs con el fin de separar el área de administración del área de gerencia dentro de la topología.

```

root@pc1-VirtualBox:~# cd mininet
root@pc1-VirtualBox:~/mininet# cd custom
root@pc1-VirtualBox:~/mininet/custom# sudo python CEP.py
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
PF3 C2F2 C1F2 PF2 C2F1 C1F1 PF1 C2F3 C1F3
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> net
PF3 PF3-eth0:s3-eth1
C2F2 C2F2-eth0:s2-eth3
C1F2 C1F2-eth0:s2-eth2
PF2 PF2-eth0:s2-eth1
C2F1 C2F1-eth0:s1-eth3
C1F1 C1F1-eth0:s1-eth2
PF1 PF1-eth0:s1-eth1
C2F3 C2F3-eth0:s3-eth3
C1F3 C1F3-eth0:s3-eth2
s2 lo: s2-eth1:PF2-eth0 s2-eth2:C1F2-eth0 s2-eth3:C2F2-eth0
s3 lo: s3-eth1:PF3-eth0 s3-eth2:C1F3-eth0 s3-eth3:C2F3-eth0
s1 lo: s1-eth1:PF1-eth0 s1-eth2:C1F1-eth0 s1-eth3:C2F1-eth0
c0
mininet> PF1 ifconfig
PF1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500

```

Captura 8: Información de la topología de centro educativo en consola

10. Abrir dos terminales más, uno para la implementación del controlador remoto de "ryu"; y el otro para la configuración de este. Para esto, abrir la carpeta de mininet "cd mininet" y también la carpeta de ryu "cd ryu", seguido habilitar el firewall del controlador remoto con el comando "ryu-manager ryu.app.rest\_firewall --ofp-tcp-listen-port 6633 --wsapi-port 8080" para que este habilite los puertos que el controlador utiliza.

```

root@pc1-VirtualBox:~/mininet/ryu# ryu-manager ryu.app.rest_firewall --ofp-tcp-listen-port 6633 --wsapi-port 8080
loading app ryu.app.rest_firewall
loading app ryu.controller.ofp_handler
instantiating app None of DPSet
creating context dpset
creating context wsgi
instantiating app ryu.app.rest_firewall of RestFirewallAPI
instantiating app ryu.controller.ofp_handler of OFPHandler
(76862) wsgi starting up on http://0.0.0.0:8080
[FW][INFO] dpid=0000000000000001: Join as firewall.
[FW][INFO] dpid=0000000000000003: Join as firewall.
[FW][INFO] dpid=0000000000000002: Join as firewall.

```

Captura 9: Firewall del controlador ryu

11. En el tercer terminal, habilitar la dirección junto al puerto con el que trabaja el controlador remoto mediante el uso del comando cURL previamente instalado en la máquina virtual "curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000001" "curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000002" "curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000003"

; seguido habilitar el firewall en el switch para la designación de la red junto a la VLAN correspondiente, con los comandos:

```
"curl -X POST -d '{"nw_src": "200.39.100.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/100" (VLAN 100)
```

```
"curl -X POST -d '{"nw_src": "200.39.200.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000002/200" (VLAN 200)
```

```
"curl -X POST -d '{"nw_src": "200.40.0.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000003/300" (VLAN 300)
```

```
root@pci-VirtualBox:~# curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000001
[{"switch_id": "0000000000000001", "command_result": {"result": "success", "details": "firewall running."}]
root@pci-VirtualBox:~# curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000002
[{"switch_id": "0000000000000002", "command_result": {"result": "success", "details": "firewall running."}]
root@pci-VirtualBox:~# curl -X PUT http://127.0.0.1:8080/firewall/module/enable/0000000000000003
[{"switch_id": "0000000000000003", "command_result": {"result": "success", "details": "firewall running."}]
root@pci-VirtualBox:~# curl -X POST -d '{"nw_src": "200.39.100.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000001/100
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1", "vlan_id": 100}]]
root@pci-VirtualBox:~# curl -X POST -d '{"nw_src": "200.39.200.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000002/200
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1", "vlan_id": 200}]]
root@pci-VirtualBox:~# curl -X POST -d '{"nw_src": "200.40.0.0/24", "nw_proto": "ICMP"}' http://localhost:8080/firewall/rules/0000000000000003/300
[{"switch_id": "0000000000000003", "command_result": [{"result": "success", "details": "Rule added. : rule_id=1", "vlan_id": 300}]]
root@pci-VirtualBox:~#
```

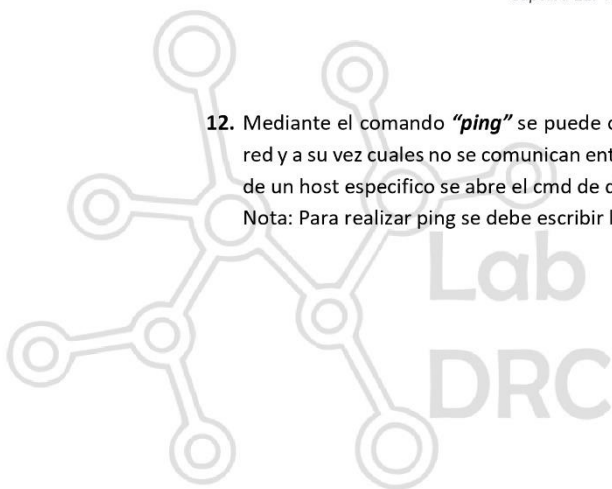
Captura 10: Configuración del controlador remoto

```
(76862) accepted ('127.0.0.1', 36956)
127.0.0.1 - - [07/Sep/2023 21:05:43] "PUT /firewall/module/enable/0000000000000001 HTTP/1.1" 200 217 0.002012
(76862) accepted ('127.0.0.1', 47124)
127.0.0.1 - - [07/Sep/2023 21:05:56] "PUT /firewall/module/enable/0000000000000002 HTTP/1.1" 200 217 0.003006
(76862) accepted ('127.0.0.1', 55518)
127.0.0.1 - - [07/Sep/2023 21:06:07] "PUT /firewall/module/enable/0000000000000003 HTTP/1.1" 200 217 0.00511
```

Captura 11: Vlans habilitadas

12. Mediante el comando **"ping"** se puede comprobar la conexión entre los dispositivos de la red y a su vez cuales no se comunican entre sí. Por otro lado, si se desea realizar ping dentro de un host específico se abre el cmd de dicho host por medio del comando **"xterm"**.

Nota: Para realizar ping se debe escribir la dirección ip del destino.





```

mininet> PF1 ping 200.39.100.13
PING 200.39.100.13 (200.39.100.13) 56(84) bytes of data.
64 bytes from 200.39.100.13: icmp_seq=1 ttl=64 time=0.292 ms
64 bytes from 200.39.100.13: icmp_seq=2 ttl=64 time=0.059 ms
64 bytes from 200.39.100.13: icmp_seq=3 ttl=64 time=0.079 ms
64 bytes from 200.39.100.13: icmp_seq=4 ttl=64 time=0.055 ms
64 bytes from 200.39.100.13: icmp_seq=5 ttl=64 time=0.057 ms
64 bytes from 200.39.100.13: icmp_seq=6 ttl=64 time=0.057 ms
64 bytes from 200.39.100.13: icmp_seq=7 ttl=64 time=0.070 ms
64 bytes from 200.39.100.13: icmp_seq=8 ttl=64 time=0.063 ms
64 bytes from 200.39.100.13: icmp_seq=9 ttl=64 time=0.057 ms
^C
--- 200.39.100.13 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8172ms
rtt min/avg/max/ndev = 0.055/0.087/0.292/0.072 ms
mininet> PF1 ping 200.39.200.11
PING 200.39.200.11 (200.39.200.11) 56(84) bytes of data.
From 200.39.100.11 icmp_seq=1 Destination Host Unreachable
From 200.39.100.11 icmp_seq=2 Destination Host Unreachable
From 200.39.100.11 icmp_seq=3 Destination Host Unreachable
^C
--- 200.39.200.11 ping statistics ---
5 packets transmitted, 0 received, +3 errors, 100% packet loss, time 4072ms
pipe 4
mininet> PF1 ping 200.40.0.11
PING 200.40.0.11 (200.40.0.11) 56(84) bytes of data.
From 200.39.100.11 icmp_seq=1 Destination Host Unreachable
From 200.39.100.11 icmp_seq=2 Destination Host Unreachable
From 200.39.100.11 icmp_seq=3 Destination Host Unreachable
^C
--- 200.40.0.11 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5105ms
pipe 4
mininet> █

```

Captura 12: Pruebas de ping

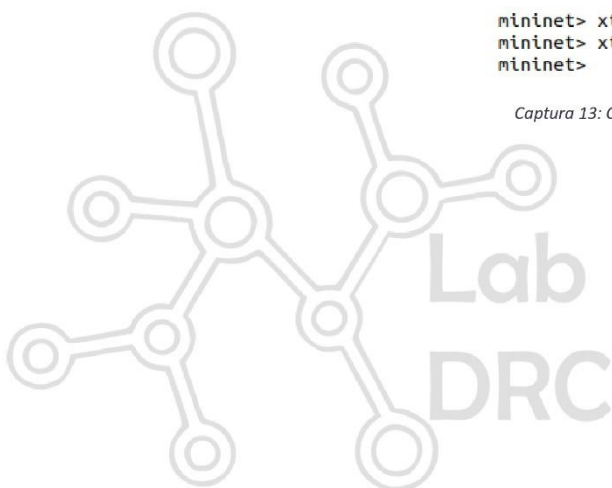
13. Mediante los comandos “nodes” se obtiene la información de los nodos conectados entre sí y con el comando “dump” la respectiva información de estos.
14. Para visualizar los paquetes que se envían en tiempo real, se utiliza el comando “xterm” en uno de los hosts y luego se ejecuta el comando “wireshark”.

```

mininet> xterm PF3
mininet> xterm C1F3
mininet>

```

Captura 13: Comando xterm



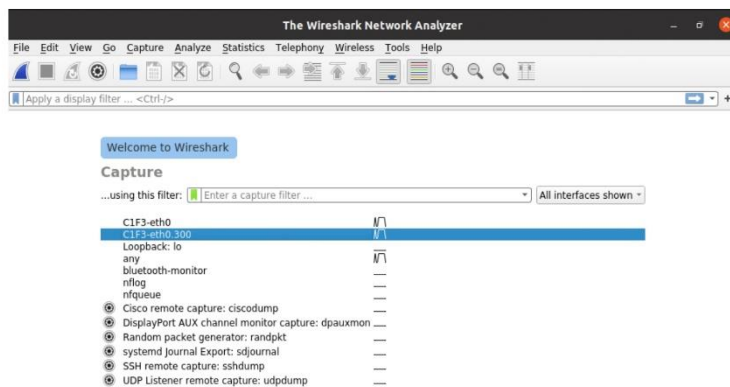
```

"Node: PF3"
PING 200.40.0.12 (200.40.0.12) 56(84) bytes of data,
64 bytes from 200.40.0.12: icmp_seq=1 ttl=64 time=0.363 ms
64 bytes from 200.40.0.12: icmp_seq=2 ttl=64 time=0.071 ms
64 bytes from 200.40.0.12: icmp_seq=3 ttl=64 time=0.060 ms
64 bytes from 200.40.0.12: icmp_seq=4 ttl=64 time=0.045 ms
64 bytes from 200.40.0.12: icmp_seq=5 ttl=64 time=0.063 ms
64 bytes from 200.40.0.12: icmp_seq=6 ttl=64 time=0.053 ms
64 bytes from 200.40.0.12: icmp_seq=7 ttl=64 time=0.063 ms
64 bytes from 200.40.0.12: icmp_seq=8 ttl=64 time=0.056 ms
64 bytes from 200.40.0.12: icmp_seq=9 ttl=64 time=0.045 ms
64 bytes from 200.40.0.12: icmp_seq=10 ttl=64 time=0.051 ms

"Node: C1F3"
root@pc1-VirtualBox:~/mininet/custom# wireshark
StandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'

```

Captura 14: Comunicación entre los hosts



Captura 15: Interfaz Wireshark



Capturing from C1F3-eth0.300

No.	Time	Source	Destination	Protocol	Length	Info
7	3.072086545	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=243/62208, ttl=
8	3.072021641	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=243/62208, ttl=
9	4.096768358	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=244/62464, ttl=
10	4.096784469	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=244/62464, ttl=
11	5.128584951	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=245/62720, ttl=
12	5.128521180	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=245/62720, ttl=
13	6.144378921	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=246/62976, ttl=
14	6.144393614	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=246/62976, ttl=
15	7.168820246	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=247/63232, ttl=
16	7.168835646	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=247/63232, ttl=
17	7.264859592	92:80:ee:7e:b9:83	aa:cf:7f:f3:85:dc	ARP	42	Who has 200.40.0.11? Tell 200.40.0.12
18	7.264253776	aa:cf:7f:f3:85:dc	92:80:ee:7e:b9:83	ARP	42	200.40.0.11 is at aa:cf:7f:f3:85:dc
19	8.192485565	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=248/63488, ttl=
20	8.192594848	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=248/63488, ttl=
21	9.216470939	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=249/63744, ttl=
22	9.216486436	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=249/63744, ttl=
23	10.239980463	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=250/64000, ttl=
24	10.239994953	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=250/64000, ttl=
25	11.264834248	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=251/64256, ttl=
26	11.264850127	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=251/64256, ttl=
27	12.288137483	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=252/64512, ttl=
28	12.288156968	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=252/64512, ttl=
29	13.312880150	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=253/64768, ttl=
30	13.312896210	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=253/64768, ttl=
31	14.335985351	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=254/65024, ttl=
32	14.335998544	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=254/65024, ttl=
33	15.369834739	200.40.0.11	200.40.0.12	ICMP	98	Echo (ping) request id=0x2ca4, seq=255/65280, ttl=
34	15.369859664	200.40.0.12	200.40.0.11	ICMP	98	Echo (ping) reply id=0x2ca4, seq=255/65280, ttl=

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface C1F3-eth0.300, id 0  
 Ethernet II, Src: aa:cf:7f:f3:85:dc (aa:cf:7f:f3:85:dc), Dst: 92:80:ee:7e:b9:83 (92:80:ee:7e:b9:83)  
 Internet Protocol Version 4, Src: 200.40.0.11, Dst: 200.40.0.12  
 Internet Control Message Protocol

```

0000  92 80 ee 7e b9 83 aa cf 7f f3 85 dc 08 00 45 00  ....E
0010  00 54 9c 8d 40 00 40 01 0d b4 c8 28 00 00 c8 28  -T-00...{
0020  00 0c 00 00 3a 00 2c a4 00 f0 ad 87 fa 64 00 00  ....d
0030  00 00 1b ac 0e 09 09 00 00 00 10 11 12 13 14 15  ....
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25  ....!%&*%
  
```

Captura 16: Análisis de paquetes

