UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

"REDISEÑO DE LA RED DE SWITCHES DE LA UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL APLICANDO REDES DEFINIDAS POR SOFTWARE."

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERA EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

MEJIA MEDINA BLANCA LUCIA

DIRECTOR: ING. BOLIVAR JÁCOME

Quito, Abril 2015

DECLARACIÓN

Yo **BLANCA LUCIA MEJIA MEDINA**, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Tecnológica Equinoccial puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

_____

MEJIA MEDINA BLANCA LUCIA

C.I. 1723553507

CERTIFICACIÓN

Certifico que el presente trabajo que lleva por título "**REDISEÑO DE LA RED DE SWITCHES DE LA UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL APLICANDO REDES DEFINIDAS POR SOFTWARE**", que, para aspirar al título de **Ingeniera en Informática y Ciencias de la Computación** fue desarrollado por **Blanca Lucia Mejía Medina**, bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería; y cumple con las condiciones requeridas por el reglamento de Trabajos de Titulación artículos 18 y 25.

_____

Ing. Bolívar Jácome

DIRECTOR DEL TRABAJO

C.I. 1707004618

DEDICATORIA

Con todo mi esfuerzo y cariño dedico este trabajo a mi familia, por su apoyo incondicional a lo largo de estos años de estudio hasta llegar a la culminación del presente proyecto, por su motivación, fuerzas en los momentos difíciles y por haber creído en mí. Gracias infinitas, los quiero mucho.

AGRADECIMIENTO

Principalmente quiero agradecer a Dios, por la protección en este camino, por haberme dado las fuerzas necesarias y las ganas para seguir, luchar y cumplir mis metas.

A mis queridos padres, por ser mi principal inspiración, por su amor, paciencia, confianza y esfuerzo, gracias por guiarme con su ejemplo y ayudarme a ser la persona que soy.

A mis hermanos, por el aliento y palabras oportunas en todos los momentos de mi vida.

Agradezco a una de las personas más importantes a lo largo de mi carrera, al amigo y maestro Ing. Freddy Velastegui, gracias infinitas por todas las enseñanzas, consejos y principalmente por la gran lección de vida que me dejó. Siempre lo recordaré.

A mi director y guía Ing. Bolívar Jácome por su tiempo, paciencia, por compartir todos sus conocimientos, sabiduría y experiencias para llegar a la culminación de este trabajo y a mis lectores Ing. Víctor Hugo Gálvez e Ing. Galo Ramos por su tiempo en la revisión del presente trabajo y por toda la orientación recibida de su parte.

A la Universidad Tecnológica Equinoccial por abrirme las puertas y así permitir prepararme y educarme en sus aulas y a todos quienes fueron mis profesores gracias por los conocimientos brindados.

A mis amigos y compañeros de aula por compartir momentos especiales e inolvidables.

ÍNDICE DE CONTENIDOS

PÁGINA

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE FIGURAS

PÁGINA

ÍNDICE DE ANEXOS

RESUMEN

El presente proyecto de titulación hace un estudio sobre las características de las redes definidas por software SDN, así como también muestra el proceso de simulación de una parte de la red de switches de la Universidad Tecnológica Equinoccial, correspondiente al primer piso de los laboratorios de computación del bloque IDIC del Campus Matriz Quito Av. Mariana de Jesús y Occidental aplicando los fundamentos de redes definidas por software, con el objetivo de experimentar y adquirir nuevos conocimientos de las redes del futuro. La simulación de redes definidas por software se desarrolló en un entorno virtual, implementado en un emulador de red Mininet y la codificación del script de red se desarrolló en el lenguaje de programación Python.

En la primera parte se describe la problemática que actualmente se está presentando en el diseño e implementación de redes de datos corporativas, en especial la red de datos de la UTE, la misma que continuamente está incrementándose no solo a nivel de servicios sino también a nivel de infraestructura tecnológica, por lo que es necesario pensar en otras alternativas para el diseño de las mismas.

En la segunda parte se hace una revisión de los fundamentos sobre Redes Definidas por Software SDN, tales como, arquitectura, controladores, protocolo OpenFlow, dispositivos, otros, así como también temas relacionados con virtualización, redes LAN y herramientas para la simulación de una red SDN personalizada.

En una tercera fase se hace una explicación sobre la metodología del ciclo de vida en cascada, seleccionada para lograr la simulación e implementación de las redes SDN ya que por sus fases de desarrollo fue la que mejor se ajustó a este proyecto.

En la cuarta parte se explica detalladamente todo el proceso llevado a cabo para la implementación y simulación de las redes SDN, que incluye los requerimientos de hardware y software, la topología personalizada de la red, pruebas de simulación de conectividad y análisis de los resultados obtenidos.

Finalmente se adjuntan varios anexos informativos tales como: parte de la planimetría de la red actual de la UTE, el rediseño de red de Switches de la Universidad Tecnológica Equinoccial aplicando redes SDN para el piso 1 del bloque IDIC del Campus Matriz Quito Av. Mariana de Jesús y Occidental, un manual de instalación de la máquina virtual para facilitar la implementación y configuración de la simulación en el emulador Mininet, el código fuente de la red utilizado para este proyecto y un Datasheet de un modelo de Switch compatible con SDN y OpenFlow.

El estudio y aplicación de las redes SDN permitirá mejorar procesos y brindar una administración más eficiente a las mismas, mediante la programación de nuevas reglas, políticas de gestión, volumen de tráfico y control en general, la programación de la lógica se encuentra directamente configurada en un controlador que será el que maneje todo el comportamiento de la red.

ABSTRACT

This titling project makes a study on network characteristics defined by SDN software defined networks, as well as the simulation shows a network of switches Universidad Tecnologica Equinoccial for the first floor of the computer labs IDIC block matrix Quito Campus Av. Mariana de Jesus and Occidental applying the fundamentals of software defined networks, in order to experience and acquire new knowledge of the networks of the future. The simulation software defined networks developed in a virtual environment, implemented in a network emulator Mininet and codification of network script developed in Python programming language.

In the first part the problems currently being presented in the design and implementation of corporate data networks, especially the data network of Universidad Tecnologica Equinoccial, it is not only continuously increasing service level but also at the level described technological infrastructure, so it is necessary to consider other alternatives for the design of them.

In the second part of a review of the basics of Software Defined Networks SDN, such as architecture, controllers, OpenFlow protocol, devices, other, as well as issues related to virtualization, LAN networks and tools for the simulation of a network is Custom SDN.

In a third phase an explanation of the methodology life cycle cascade selected to achieve the simulation and implementation of SDN networks as for their development phases was the one that best adjusted to this project is done.

In the fourth part explained in detail the process undertaken for the implementation and simulation of the SDN network, which includes hardware requirements and software, customized network topology simulation tests connectivity and analysis of results .

Finally attached several informative annexes such as:. Part of the mapping of the existing network of the joint venture, the redesign of network switches Universidad Tecnologica Equinoccial SDN networks applying for apartment 1 block matrix IDIC Campus Av Quito Mariana de Jesus and Occidental, a manual installation of the virtual machine to facilitate the implementation and configuration of simulation in the emulator Mininet, the source code of the network used for this project and a Datasheet Switch model supports SDN and OpenFlow.

The study and application of the SDN network will improve processes and provide a more efficient administration thereof, by programming new rules, policies, management, and control traffic volume overall programming logic is configured directly in a controller that will handle all the network behavior.

INTRODUCCIÓN

1. INTRODUCCIÓN

El presente proyecto permite conocer los conceptos y la arquitectura de las Redes Definidas por Software (SDN) conjuntamente con su protocolo OpenFlow y temas relacionados con redes, topologías y medios, así como también, la implementación de una simulación de red SDN mediante el programa Mininet, instalado en una máquina virtual con ciertas aplicaciones tales como Wireshark y xterm que permiten observar, administrar el tráfico y manipular independiente cada nodo de red.

Durante muchos años las redes de comunicación de datos se han desarrollado en base a diferentes modelos tales como el modelo OSI, el modelo TCP/IP y el modelo jerárquico. En la actualidad y gracias al desarrollo de nuevas tecnologías, aplicaciones y servicios (*cloud computing*), el diseño de las redes de la nueva generación será mediante software, donde cada administrador configurará rápidamente una red de acuerdo a sus necesidades y requerimientos.

SDN es la creación de redes y la virtualización de centros de datos de manera segura y eficiente, y se construye usando el protocolo OpenFlow, para lo cual se define el protocolo que permite a un controlador utilizar un conjunto común de instrucciones para añadir, modificar o borrar entradas en la tabla de direccionamiento de un switch.

SDN y OpenFlow se puede utilizar para proporcionar nuevos servicios y características que van a cambiar la forma en la creación de redes y los impactos innovadores de éstas, permitiendo quitar, agregar o cambiar cualquier regla de los switches cuando sea necesario o asignando prioridad y hasta bloqueando tipos específicos de paquetes. (Siamak, 2013).

Las nuevas tecnologías de redes SDN son más personalizadas, lo que implica tener más conocimientos sobre temas tales como:

- Redes de gran escala
- Movilidad
- QoS
- Migración de máquinas virtuales
- Gran cantidad de datos (*BigData*).
- Aplicaciones Móviles
- Servicios en la Nube
- Planos de Datos
- Servicios de OpenFlow

Uno de los principios de SDN es la separación del control de un dispositivo de red y los planos de datos. Esta separación permite a un operador de red tener ciertas ventajas de control sobre éstas ya sea de forma centralizada o distribuida. También tiene una ventaja potencial sobre la capacidad para configurar y controlar a menos costo el hardware. Las redes SDN optimizan y simplifican las operaciones al unirse más entre aplicaciones, servicios y dispositivos, ya sea en forma real o virtual. Esto se logra mediante el empleo de un punto de control de red centralizada en el controlador de SDN que facilita la comunicación entre aplicaciones que desean interactuar con los elementos de red y que deseen transmitir información a otras aplicaciones. (Thomas D. Nadeau & Ken Gray, 2013).

Actualmente la red de datos de la Universidad Tecnológica Equinoccial (UTE) experimenta un crecimiento en el volumen y tráfico de información debido entre otras cosas, al incremento del número de usuarios (estudiantes, docentes, personal administrativo y de servicios), aumento de convenios con instituciones públicas y privadas, más y mejores proyectos de investigación y de vinculación con la colectividad, nuevos servicios de red, entre otros.

Esto trae como consecuencia el crecimiento de la infraestructura física y tecnológica, el aumento en el tráfico de la red, el incremento de vulnerabilidades, tiempos de respuesta más elevados (mayor retardo), entre otros.

Como contribución a la solución a estos problemas a largo plazo, se pone a consideración este proyecto de tesis que aporta con las bases teóricas y metodológicas para la simulación del comportamiento de las SDN, lo que ayudará en el diseño e implementación de las redes de nueva generación en la UTE.

Por lo antes indicado se justifica que el modelo de red SDN es indudablemente necesario para cualquier organización y campus donde se manipulen grandes cantidades de datos, permitiendo así administrar de manera virtual las configuraciones de los equipos y reduciendo el alto consumo de recursos físicos, técnicos y económicos.

Este estudio es importante para la UTE ya que permite simular una red SDN de una parte de los laboratorios de computación del Instituto de Informática y Computación (IDIC) y que a futuro, con el uso de equipos con tecnología SDN, pueda ser implementada en toda la red del campus.

El objetivo principal de este proyecto de titulación es rediseñar la red de switches del primer piso de los laboratorios del IDIC del Campus Matriz Quito Av. Mariana de Jesús y Occidental de la UTE, sede Quito, a través de la simulación de la misma utilizando la tecnología SDN.

Una vez cumplido con el objetivo principal se procederá a cumplir los objetivos específicos que se describen a continuación:

- Explicar el funcionamiento de las redes SDN, a través de una sólida fundamentación de las leyes y teorías que sustentan la misma.
- Diseñar la topología lógica de Switches de red SDN de los laboratorios de computación del IDIC, a partir de la topología física del mismo.
- Simular el funcionamiento de la red SDN, del primer piso de los laboratorios de computación del IDIC.
- Analizar los tiempos de respuesta y tráfico de la red SDN simulada, a través del uso de comandos de conectividad y el analizador de tráfico Wireshark.

MARCO TEÓRICO

2. MARCO TEÓRICO

En actualidad los servicios en la nube están ganando terreno en la administración de grandes servidores de datos a nivel de empresas, campus e instituciones. En este capítulo se dará a conocer conceptos, arquitectura y funcionamiento de redes definidas por software.

2.1. REDES DE COMPUTADORAS

Desde que el ser humano tiene capacidad de interactuar, ha desarrollado mecanismos y sistemas que permiten establecer una conexión a distancias superiores de las alcanzadas por sus propios medios. Al poco de aparecer los ordenadores, se sintió la necesidad de interconectarlos para que se pudiesen comunicar entre sí como lo hacemos los humanos y poder compartir información y recursos.

Una red de computadoras, es un conjunto de equipos informáticos y software conectados entre sí por medio de dispositivos físicos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios. (Barceló Ordinas, Iñigo Griera, Martí Escalé, Peig Olivé, & Perramon Tornil, 2004).

2.1.1. TIPOS DE REDES

Las redes de computadoras según su extensión y tamaño se dividen en varios tipos, esto depende de la zona geográfica, la infraestructura y la distancia. Dependiendo de las características que se requieren para ésta, se elegirá: la topología, los medios (guiados y no guiados), los dispositivos de red y los servicios. A continuación se presenta una descripción de las características de los diferentes tipos de redes.

- **Redes LAN**

Una red LAN es un conjunto de dispositivos (switch, computadoras, impresoras, radios inalámbricos) y medios (cable UTP, fibra óptica, ondas electromagnéticas) interconectados entre sí, para compartir información (base de datos, aplicaciones, programas) y recursos (servidores, impresoras, teléfonos y otros dispositivos de red) en distancias en el orden de los metros o decenas de metros. Este tipo de redes son amplias a nivel mundial (Tanenbaum, 2003).

Entre las principales características de las redes LAN se pueden indicar las siguientes:

- **Tamaño lógico**: Depende de la clase de IP y cantidad de hosts conectados en la red.
- **Tamaño físico**: Depende del área de cobertura de la red (menor a 100 mts).
- **Velocidad**: Entre 10 Mbps (Ethernet), 100 Mbps (Fast Ethernet) y 1000 Mbps (Gigabit Ethernet).
- **Topología**: Esta red está construida en una topología en estrella y estrella extendida, donde todos los equipos están conectados a un punto central de red (switch).
- **Tecnología:** Cableada (LAN) o inalámbrica (WLAN).

- **Redes MAN**

Una red de área metropolitana (MAN*)* permite área de cobertura en el orden de Km. o decenas de Km., por ejemplo la red del municipio de la ciudad de Quito (Metro Ethernet) abarca una ciudad o la red de televisión por cable (TV cable) disponible en muchas ciudades.

- **Redes WAN**

Una red de área amplia (WAN*)* es un conjunto de dispositivos y medios, que permiten la interconexión de diferentes redes LAN o dispositivos a nivel mundial, cubriendo una gran área geográfica, con frecuencia un país o un continente.

Como ejemplos de redes WAN se pueden indicar: La red de telefónica celular, Internet, TV Satélite, entre otras. En el caso de la red de Internet, los clientes son quienes poseen a los hosts (computadoras personales), mientras que, por lo general, las compañías telefónicas o los proveedores de servicios de Internet poseen y operan la subred de comunicación. (Tanenbaum, 2003).

- **Redes PAN**

Una red de área personal (PAN*)* es un conjunto de dispositivos y medios interconectados entre sí, para compartir información y recursos a distancias en orden de los metros. Bluetooth es un ejemplo de red PAN, ya que es una tecnología que permite crear una red Ethernet con vínculos inalámbricos entre equipos móviles (celulares, laptops, tablet, etc.).

- **Redes HAN**

Redes de área domestica (HAN*)* nacen de la necesidad de facilitar la comunicación y la interconexión entre diferentes dispositivos digitales o equipos existentes en el interior o inmediaciones de una casa. Como ejemplo de protocolos para redes HAN se puede indicar: redes X10, KNX, Lonworks, otros. Estos estándares permiten la comunicación entre diferentes dispositivos de una red domótica.

- **Redes Inalámbricas WLAN**

Las redes locales inalámbricas (WLAN) se diferencian de las redes LAN cableadas, en la naturaleza del medio que se emplean para transmitir sus datos. Las redes WLAN también conocidas como Wi-Fi utilizan como medio de transmisión el aire. El acceso a los elementos cableados es físicamente complejo porque se encuentran protegidos por paredes y puertas, sin embargo el limitar el acceso por el medio de transmisión inalámbrico es más complicado, lo que conlleva que se hayan convertido en objetivos interesantes de posibles ataques. Estos problemas de seguridad son de especial relevancia en empresas o entornos empresariales por la confidencialidad de la información utilizada en los mismos. Por esta razón, las redes inalámbricas necesitan mecanismos de seguridad adicional o simplemente diferente, para garantizar un nivel adecuado de seguridad. (Andreu, Pellejero & Lesta, 2006).

2.1.2. TOPOLOGÍA DE REDES

Las redes de computadoras cumplen la tarea de interconectar los diferentes hosts o dispositivos de manera que estos se comuniquen y compartan recursos e información entre sí. La manera de conectar los distintos elementos de una red determina el comportamiento de ésta, para lo cual se inventaron las topologías, que se refiere al estudio de las formas que se emplea en un diseño de redes de comunicación, una topología está formada de tres elementos que son, nodos, enlaces y equipos terminales. Su eficiencia y aprovechamiento dependerá también de los protocolos de comunicación que se utilicen. Para seleccionar la mejor topología el administrador debe saber la cantidad de hosts existentes dentro de la red.

Las topologías de red, conocidas como topologías físicas más utilizadas son:

- Topología en Estrella.

- Topología en Estrella Extendida.
- Topología en Bus.
- Topología en Anillo.
- Topología en Malla.

A continuación se hace una breve descripción de las características de cada una de ellas.

- **Topología en Estrella**

Cuando un ordenador envía una trama en la red, ésta aparece de inmediato en las entradas del resto de ordenadores. (Barceló Ordinas, Íñigo Griera, Martí Escalé, Peig Olivé & Perramon Tornil, 2004). En la figura 1 se puede observar la topología en estrella que consiste en conectar, a través de cables cada ordenador a un punto central, este punto puede ser un hub o un switch. En esta topología el número de enlaces (conexiones) es igual a (n – 1), siendo n el número de nodos.

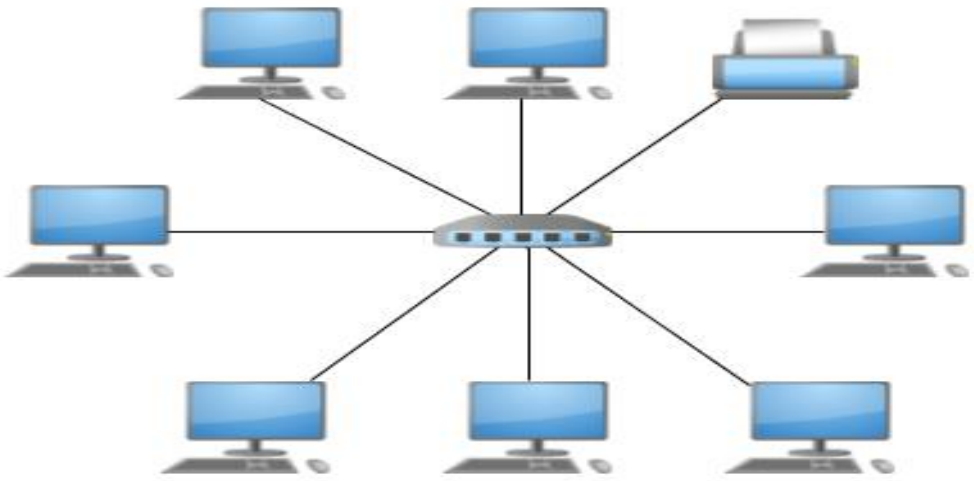


Figura 1. Topología en Estrella

- **Topología en estrella extendida**

La topología en estrella extendida es igual a la topología estrella, con la diferencia de que cada nodo que se conecta con el nodo central también es el centro de otra estrella. Generalmente el nodo central está ocupado por un hub o un switch, y los nodos secundarios también pueden ser hubs o switchs. La ventaja de esto es que el cableado es más corto y limita la cantidad de dispositivos que se deben interconectar con cualquier nodo central. La topología en estrella extendida es sumamente jerárquica, y busca que la información se mantenga local. Esta es la forma de conexión utilizada actualmente en redes de datos y de telefonía convencional. (Cruz Álvarez, Melo Quiñónez, & Rodríguez Sierra, 2008). En la figura 2 se puede observar la topología en estrella extendida que consiste en interconectar pequeñas redes de topología en estrella entre sí mediante un switch. En esta topología el número de enlaces (conexiones) es igual a $(n - 1)$, siendo n el número de nodos.

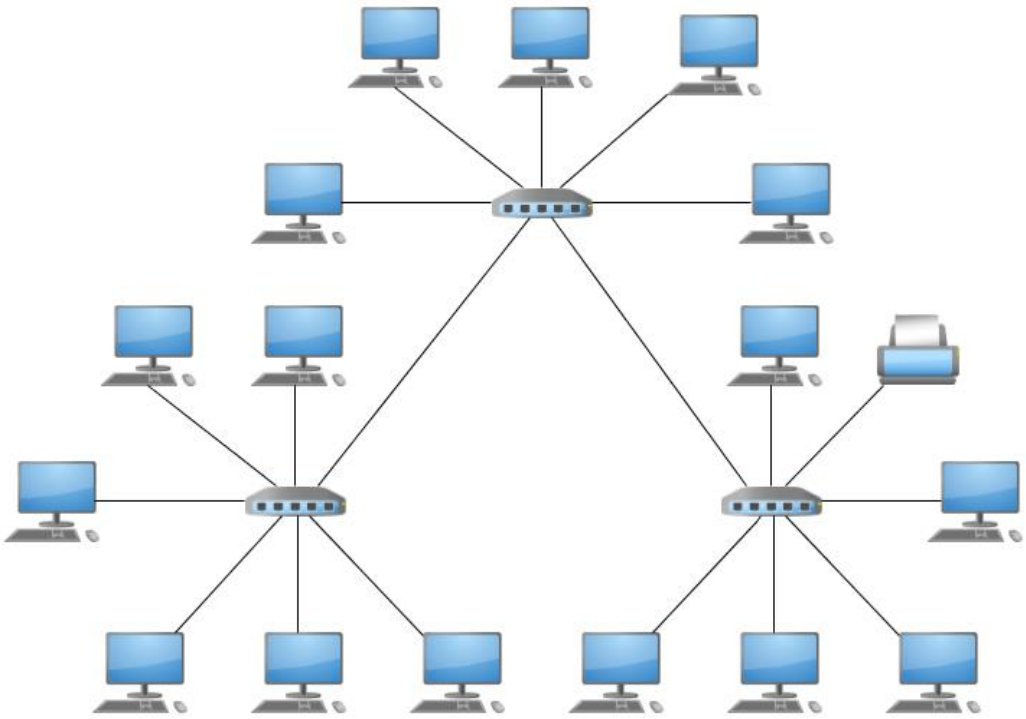


Figura 2. Topología en Estrella Extendida

- **Topología en Bus**

En el momento en que un ordenador envía una trama, todos los ordenadores la comprueban si son el destinatario de la misma, si es así la reciben, caso contrario la descartan. (Barceló Ordinas, Íñigo Griera, Martí Escalé, Peig Olivé & Perramon Tornil, 2004). En la figura 3 se puede observar la topología en bus que consiste en un cable al que se unen todas las estaciones de la red. Todos los ordenadores están pendientes de si hay actividad en el cable. En esta topología el número de enlaces (conexiones) es igual a 1, siendo **n** el número de nodos.

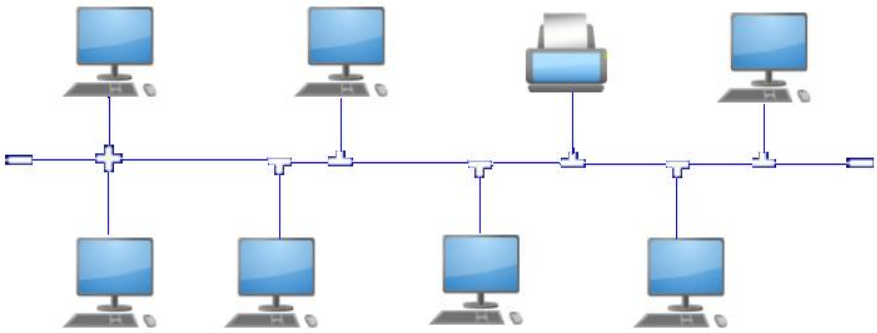


Figura 3. Topología en Bus

- **Topología en Anillo**

Cuando un ordenador quiere enviar una trama a otro, un token (pequeño mensaje) es enviado simultáneamente a todos los ordenadores, si el ordenador que recibe el token es el destinatario, éste recibe la información, la circulación por el anillo es unidireccional. (Barceló Ordinas, Íñigo Griera, Martí Escalé, Peig Olivé & Perramon Tornil, 2004). En la figura 4 se puede observar la topología en anillo que consiste en conectar cada ordenador a dos más, de manera que se forme un anillo.

En esta topología el número de enlaces (conexiones) es igual a **n**, siendo **n** el número de nodos.

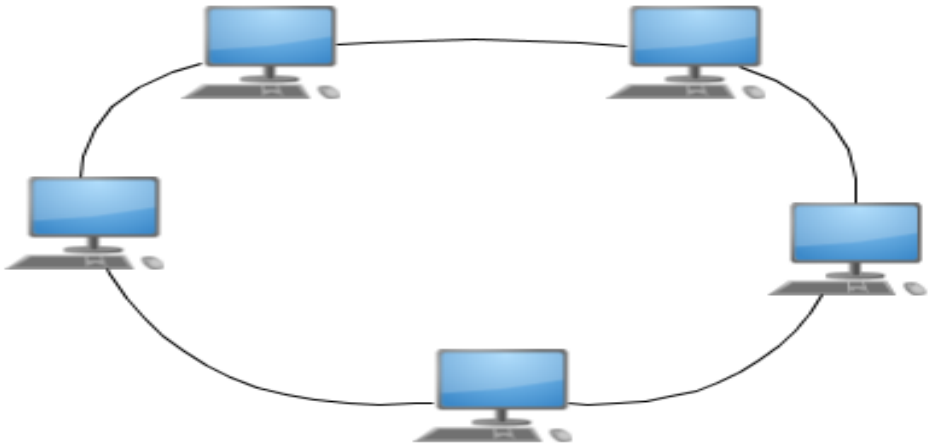


Figura 4. Topología en Anillo

- **Topología en Malla**

En una topología en malla, cada nodo se enlaza directamente con todos los demás. La ventaja de esta conexión es que crea enlaces redundantes, es decir si algún enlace se cae, la información puede circular a través de otro enlace, dando como resultado una red tolerante a fallos. Además, esta topología permite que la información circule por varias rutas a través de la red. (Cruz Álvarez, Melo Quiñónez, & Rodríguez Sierra, 2008).

En esta topología el número de enlaces (conexiones) es igual a **n** (**n** – 1)/2, siendo **n** el número de nodos. En la figura 5 se puede observar una topología en malla en donde cada nodo está conectado directamente a todos los demás.

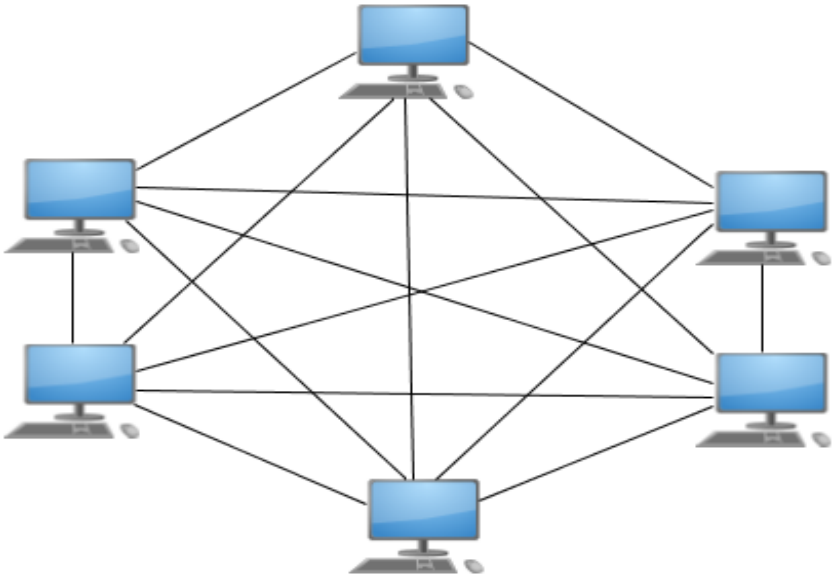


Figura 5. Malla Completa

2.2. PROTOCOLOS

Son un conjunto de normas y reglas que se aplican en la informática para la comunicación entre dos o más dispositivos, de esta manera se puede intercambiar información y datos. En redes de datos éstos se pueden clasificar en dos grupos: protocolos enrutados y protocolos de enrutamiento.

Actualmente los protocolos enrutados más utilizados en redes de datos son:

- IPv4
- IPv6

2.2.1. Protocolo de Internet (IP)

El Protocolo de Internet es un protocolo de capa de red (Capa 3), diseñado en 1981 para usarse en sistemas interconectados de redes de comunicación y conmutación de paquetes. El Protocolo de Internet y el Protocolo de Control de Transmisión (TCP), son la base de los protocolos de Internet.

El IP tiene dos funciones principales:

- Entrega de datagramas a través de la intranet en la modalidad de mejor esfuerzo.
- Fragmentación y reensamblado de datagramas.

Se considera al IP un protocolo de “mejor esfuerzo” ya que no garantiza que un paquete transmitido realmente llegue al destino ni que los datagramas transmitidos sean recibidos en el orden en que fueron enviados. La función principal de IP es llevar paquetes de datos de un nodo fuente a un nodo destino. Este proceso se logra identificando cada paquete enviado con una dirección numérica llamada dirección IP. El protocolo IP no tiene mecanismos de confiabilidad a diferencia de los demás protocolos. En vez de tener dichos medios, este protocolo no hace uso de ellos para que sean implementados por protocolos de capa superior.

El único mecanismo de detección de errores es la suma de verificación para el encabezado IP. Si el procedimiento de la suma de verificación falla, el datagrama será descartado y con ello no será entregado a un protocolo de nivel superior. El esquema de direccionamiento IP es integral al proceso de enrutamiento de datagramas IP a través de la intranet. Cada dirección IP tiene componentes específicos y un definido formato básico.

Existen dos estándares de direccionamiento IP: la versión 4 (IPv4) y la versión 6 (IPv6). Actualmente la mayoría del tráfico IP es realizado con direccionamiento IPv4, y aunque se pretende que IPv6 reemplace a IPv4 en un futuro, ambos protocolos coexistirán durante algún tiempo.

2.2.2. IPv4

Las direcciones IPv4 se expresan por un número binario de 32 bits permitiendo un espacio de 4.294.967.296 direcciones posibles. Las

direcciones IP se pueden expresar como número de notación decimal: se dividen los 32 bits de la dirección de cuatro octetos. El valor decimal de cada octeto está comprendido en el rango de 0 a 255 (el número binario de 8 bits más alto es 11111111 y esos bits de derecha a izquierda, tienen valores decimales de 1, 2, 4, 8, 16, 32, 64 y 128, lo que suma 255). (Esparza, 2013).

Como se indica en la figura 6, una dirección IP está dividida en octetos y agrupados éstos en dos campos importantes: el campo host y el campo red, en donde el tamaño de cada uno de ellos caracteriza la clase de dirección IP (clase A, B, C, D) y si es pública o privada.

- **Formato de Dirección IP Versión 4**

En una red TCP/IP a cada computadora se le asigna una dirección lógica de 32 bits, en donde cada bit en el octeto tiene un peso binario. El valor mínimo para un octeto es 0 y el valor máximo es 255. Más adelante se hará una descripción detallada de las diferentes clases de direcciones IP utilizadas en el diseño de redes. En la figura 6 se muestra el formato básico de una dirección IP con sus 32 bits agrupados en 4 octetos.

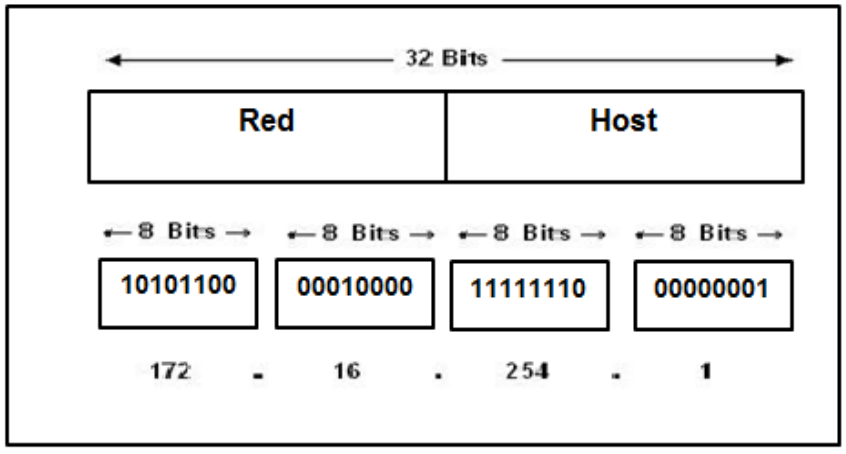


Figura 6. Formato de dirección IP versión 4

2.2.3. IPv6

Con el paso del tiempo IPv4 llegó a ser uno de los protocolos de mayor utilización y se creía que este protocolo iba a solventar las necesidades de la época así como también del futuro, ya que se pensó que la direcciones que ofrecía eran suficientes para abastecer a todos y cada uno de los usuarios que necesiten de una dirección de internet. El crecimiento en el área tecnológica en la última década ocasionó que las direcciones IPv4 se agotaran más rápido de lo pensado debido a que las personas ya cuentan con varios equipos que necesitan de una conexión a internet como por ejemplo: Smartphone, Tablet y todo lo referente a una conexión móvil. (Barcenilla & Eijo, 2004).

Esto permitió que las entidades de regulación desarrollen otro tipo de direccionamiento IP, al que denominaron IPv6, el mismo que permitirá cubrir las necesidades de conectividad de más de 7000 millones de personas en el mundo. A continuación en la figura 7 se presenta la cabecera de IPv6 con sus respectivos campos:

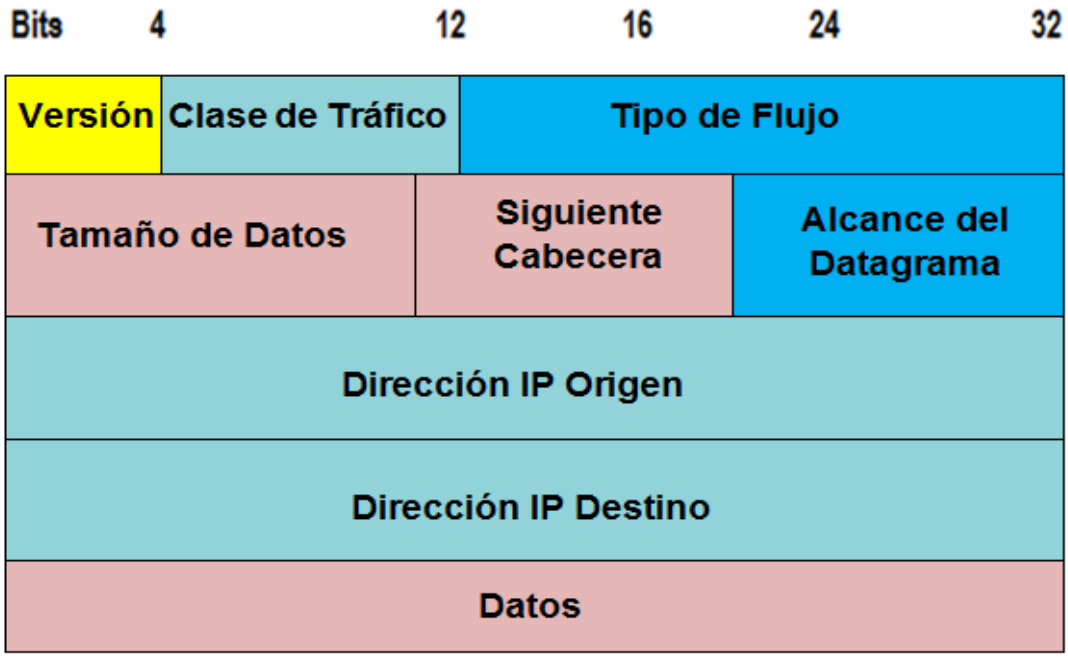


Figura 7. Formato de Cabecera de IPv6

En donde:

- **Versión:** Versión del protocolo, es equivalente a 4 bits.
- **Clase de tráfico:** Indica el servicio solicitado por un paquete y asigna una prioridad, clasifica paquetes con diferentes requisitos y los entrega en tiempo real, es equivalente a 8 bits.
- **Tipo de Flujo:** Lleva a cabo procesamientos de tráfico, el router debe escoger una ruta y procesar información en tiempo real**,** es equivalente a 20 bits.
- **Tamaño de Datos:** Contiene solo el tamaño de la carga útil, sin incluir la cabecera IPv6, es equivalente a 16 bits.
- **Siguiente cabecera:** Indica que cabecera es la siguiente después de la cabecera fija de IPv6, es equivalente a 8 bits.
- **Alcance del datagrama:** Número de saltos máximos que posee un paquete, el máximo lo determina el origen y se decrementa en 1 cada vez que un nodo reenvía un paquete, cuando el límite de saltos llega a 0 el paquete es descartado, es equivalente a 8 bits.
- **IP Origen:** Dirección de origen de un paquete, es equivalente a 128 bits.
- **IP Destino:** Dirección del destino de un paquete, es equivalente a 128 bits.

- **Prefijos de IPv6**

Los prefijos son identificadores de subredes, rutas o rangos de direcciones IPv6, que son expresados de la misma forma que en la notación CIDR utilizada en IPv4.

En una dirección IPv6 un host que por ejemplo tiene una dirección 3FFE:B00:C18:1::1/64, primero se identifica al prefijo 64 que corresponde a 64 bits correspondientes al número de red y los otros 64 bits restantes como parte del host. Cualquier prefijo que sea menor a 64 bits es una ruta o un

rango de direcciones que son resumidos de una porción del espacio de direcciones.

2.3. ESTÁNDARES Y ORGANISMOS

Se puede definir un estándar como un patrón, modelo o punto de referencia a seguir para medir o comparar cosas de un mismo tipo. En la industria de las comunicaciones, los estándares se han creado para definir las características físicas, eléctricas y de procedimiento de los equipos de comunicación. Hay una serie de ventajas y desventajas en el proceso de estandarización. A continuación, se citan las más relevantes. (Stallings, 2004).

2.3.1 Ventajas

- La existencia de un estándar para un software o equipo dado asegura potencialmente un gran mercado. Esto estimula la producción masiva y, en algunos casos, el uso de integración a gran escala (LSI) o integración a muy gran escala (VLSI), reduciéndose así los costes.

- Un estándar permite que los productos de diferentes fabricantes se comuniquen, dotando al comprador de mayor flexibilidad en la selección y uso de los equipos.

2.3.2 Desventajas

- Los estándares tienden a congelar la tecnología. Mientras que un estándar se desarrolla, se revisa y se adopta, se pueden haber desarrollado otras técnicas más eficaces. (Stallings, 2004).

2.3.3. Tipos de Estándares y Organismos

- **Estándar ISO**

Organización de Estándares Internacionales: esta organización tiene a su cargo una amplia gama de estándares, incluyendo aquellos referidos al *networking*. (Cruz Alvarez, Melo Quiñonez, & Rodriguez Sierra, 2008).

- **Estándar ANSI**

Instituto Nacional Americano de Normalización: ayuda a desarrollar y aprueban los estándares de los EE.UU. e internacionales, entre otras cosas comunicaciones y *networking*. (Cruz et al., 2008).

- **Estándar ITU**

Unión Internacional de Telecomunicaciones (ITU): es el organismo especializado de las Naciones Unidas encargado de regular las telecomunicaciones a nivel internacional, entre las distintas administraciones y empresas operadoras. (Cruz et al., 2008).

- **Estándar IEEE**

Instituto de Ingenieros Eléctricos y Electrónicos: Organización profesional cuyas actividades incluyen el desarrollo de estándares de comunicaciones y redes. Los Estándares de LAN (IEEE) son los de mayor importancia de la actualidad (802.x). (Cruz et al., 2008).

2.4. MODELOS DE REFERENCIA

2.4.1. MODELO OSI

El modelo básico de referencia OSI, afronta el problema de las comunicaciones de datos y las redes informáticas dividiéndolo en niveles o capas. Cada participante de la red se comunica con su par a través de lenguajes de niveles o capas denominadas PDU. (Barceló Ordinas, Iñigo Griera, Martí Escalé, Peig Olivé, & Perramon Tornil, 2004).

El Modelo de referencia OSI, también conocido como modelo 7 capas, para una mejor comprensión, estudio y análisis divide una red en siete niveles o capas llamadas:

- Aplicación
- Presentación
- Sesión
- Transporte
- Red
- Enlace
- Física

En la figura 8 se puede observar las capas del Modelo OSI con sus respectivos PDU, en donde:

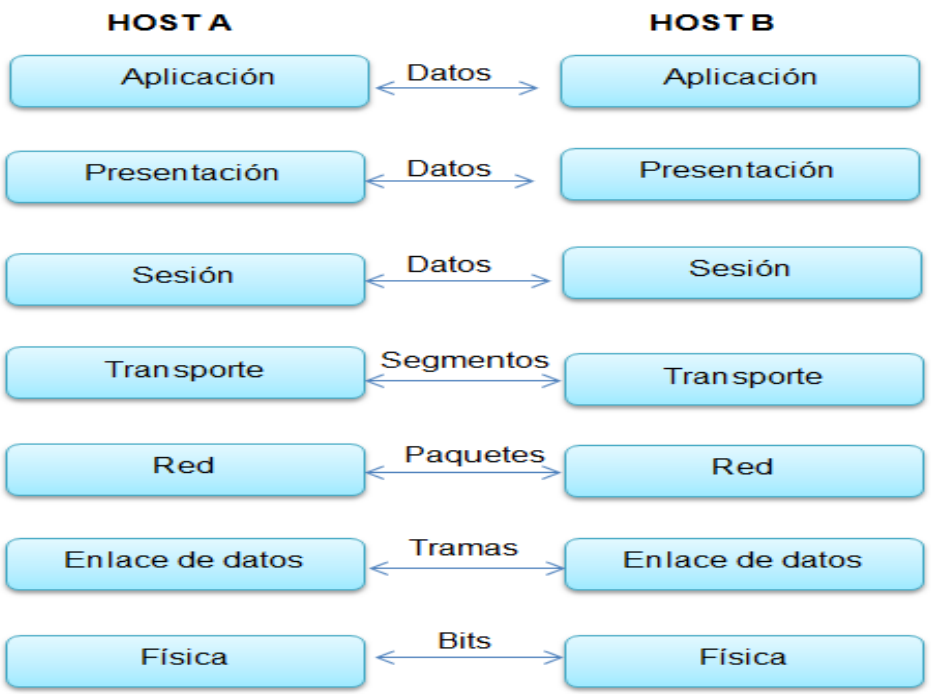


Figura 8. Capas del Modelo OSI

- **La Capa Física**

En esta capa se lleva a cabo la transmisión de bits puros a través de un canal de comunicación. Los aspectos del diseño implican asegurarse de que cuando un lado envía un bit 1, éste se reciba en el otro lado como tal, no como bit 0. (Tanenbaum, 2003).

En la capa física, el transmisor envía un conjunto de bits los mismos que son codificados en diferentes señales sean éstas eléctricas, electromagnéticas o luz, dependiendo el medio de conexión, permitiendo que la señal sea interpretada en el receptor. En esta capa se realizan las conexiones físicas entre la tarjeta de red del computador y su correspondiente par, a través de medios.

Un medio es el elemento que permite la conexión de dos o más dispositivos de red y en forma general se los clasifica en:

- Medios guiados

    - Cable Coaxial.
    - Cable trenzado.
    - Fibra Óptica.

- Medios no guiados

    - Radio frecuencia.
    - Microondas.
    - Infrarrojos: Control remoto.
    - Láser.
    - Wifi: Frecuencias 2.4 y 5.

Las principales funciones que se realizan a nivel de la capa Física son:

- Establecer el tipo de medios físicos por donde va a viajar la información en una red de comunicación de datos.
- Establecer las características de los materiales que se van a utilizar para su comunicación, como conectores mecánicos y conexiones eléctricas.
- Manipular diferentes tipos de señales eléctricas y/o electromagnéticas.
- Permite la trasmisión del flujo de bits a través del medio físico.
- Asegurar y garantizar la perfecta conexión y comunicación entre los medios.

En la figura 9 se muestra un ejemplo del PDU (bits) de la capa física

Figura 9. Ejemplo del PDU de la capa física

- **La Capa de Enlace de Datos**

La tarea principal de esta capa es transformar un medio de transmisión puro en una línea de comunicación que, al llegar a la capa de red, aparezca libre de errores de transmisión. Logra esta tarea haciendo que el emisor fragmente los datos de entrada en tramas de datos (típicamente, de algunos cientos o miles de bytes) y transmitiendo las tramas de manera secuencial. Si el servicio es confiable, el receptor confirma la recepción correcta de cada trama devolviendo una trama de confirmación de recepción. (Tanenbaum, 2003).

En esta capa se define todo el direccionamiento dentro de una red tal como por ejemplo, el físico para el acceso a la red, de la topología de red, del acceso hacia la red, de las notificaciones de errores si el caso los presentara, de la distribución ordenada y correcta de las tramas y finalmente del control del flujo de datos y de información. Mediante la tarjeta de red se establece la conexión, posee una dirección MAC (control de acceso al medio) y una LLC (control de enlace lógico). El PDU en esta capa es la trama y en esta capa los switches realizan sus funciones.

En la figura 10 se puede observar el PDU de la capa de enlace que se denomina como trama.

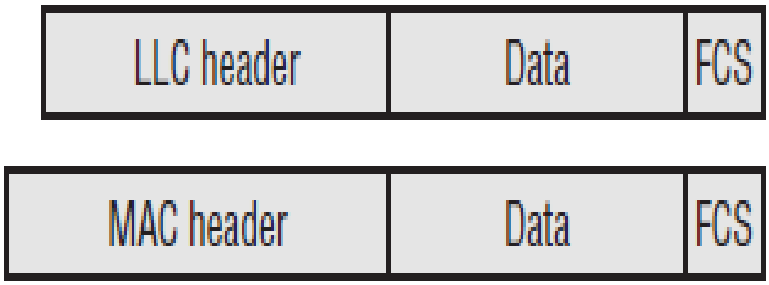


Figura 10. Trama es el PDU de la capa de Enlace

- **La Capa de Red**

Esta capa controla las operaciones de la subred. Un aspecto clave del diseño es determinar cómo se enrutan los paquetes desde su origen a su destino. Las rutas pueden estar basadas en tablas estáticas (enrutamiento estático) codificadas en la red y que rara vez cambian.

En esta capa se define el direccionamiento lógico, que permite a los hosts identificar las tramas destinadas a ellos. Este direccionamiento es único, identifica el hardware de red que se está usando y el fabricante, y no se puede cambiar.

En el enrutamiento estático la ruta que seguirán los paquetes hacia un destino particular es determinada por el administrador de la red. Las rutas también pueden determinarse cuando los routers intercambian información de enrutamiento (enrutamiento dinámico).

En el enrutamiento dinámico los routers deciden la ruta que seguirán los paquetes hacia un destino sin la intervención del administrador de red, las rutas pueden cambiar para reflejar la topología o el estado de la red. (Tanenbaum, 2003).

En la figura 11 se puede observar el PDU de la capa de red que se representa por paquetes.

Figura 11. Paquete es el PDU de la capa de Red

- **La Capa de Transporte**

La función básica de esta capa es aceptar los datos provenientes de las capas superiores, dividirlos en unidades más pequeñas llamados fragmentos

si es necesario, pasar éstas a la capa de red y asegurarse de que todos los datos lleguen correctamente al otro extremo. La capa de transporte también determina qué tipo de servicio proporcionar a la capa de sesión y, finalmente, a los usuarios de la red. El tipo de conexión de transporte más popular es un canal punto a punto libre de errores que entrega mensajes o bytes en el orden en que se enviaron. Sin embargo, otros tipos de servicio de transporte posibles son la transportación de mensajes aislados, que no garantiza el orden de entrega, y la difusión de mensajes a múltiples destinos. El tipo de servicio se determina cuando se establece la conexión. (Tanenbaum, 2003).

Esta capa es la encargada de realizar el transporte de los datos provenientes en un paquete desde una máquina o host origen hacia un destino, realizando su función de una manera independiente de la capa de red. Hay dos tipos de servicio en la capa de transporte, orientado y no orientado a la conexión.

- **Comunicación orientada a la conexión:** Es una operación de transporte fiable, cuando un dispositivo desea transmitir algún paquete establece una comunicación con otro dispositivo creando una sesión con previo acuerdo. Los dos dispositivos se comunican mediante el envío y la recepción de mensajes creando un envío de mensajes a través de la red, cuando la transferencia de archivos y las dos partes están listas se procede y se lleva a cabo. Por ejemplo, en una llamada telefónica las dos partes están listas, primero se inicia la conexión, se utiliza y por último se finaliza.

- **Comunicación no orientada a la conexión**: En esta conexión a diferencia de la anterior no se establece ninguna confirmación y puede realizar una transferencia de archivos sin que la otra parte este lista. Debido a que no se recibe una confirmación se pueden perder ciertos paquetes o archivos, es por esto que el remitente realiza un

descanso para la transmisión de más datos. Por ejemplo, en un servicio de fax, las dos partes no siempre están disponibles.

En la figura 12 se puede observar el PDU de la capa de transporte que se representa por segmentos.

Figura 12. Segmento es el PDU de la Capa de transporte

- **La Capa de Sesión**

Esta capa permite que los usuarios de máquinas diferentes establezcan sesiones entre ellos. Las sesiones ofrecen varios servicios, como el control de diálogo.

- Administración de token (que impide que las dos partes traten de realizar la misma operación crítica al mismo tiempo) y sincronización (la adición de puntos de referencia a transmisiones largas para permitirles continuar desde donde se encontraban después de una caída). (Tanenbaum, 2003).
- Control al establecer una sesión entre el emisor y receptor de un archivo, es decir quien envía y quien recibe.
- Establecer orden entre dos comunicaciones, debe ser concurrente y evitar que las dos se realicen al mismo tiempo.
- Establecer de manera correcta la comunicación entre dos máquinas, efectuar las operaciones establecidas a los largo de toda la transmisión y garantizar la reanudación en caso de alguna interrupción o problema.
- Sobre la capa de sesión actúan los *firewall* donde este decide si bloquear o no los accesos a los puertos de un ordenador, mediante previas configuraciones.

- **La Capa de Presentación**

A diferencia de las capas inferiores, a las que les corresponde principalmente mover bits, a la capa de presentación le corresponde la sintaxis y la semántica de la información transmitida. A fin de que las computadoras con diferentes representaciones de datos se puedan comunicar, las estructuras de datos que se intercambiarán se pueden definir de una manera abstracta, junto con una codificación estándar para su uso “en el cable”. La capa de presentación maneja estas estructuras de datos abstractas y permite definir e intercambiar estructuras de datos de un nivel más alto (por ejemplo, registros bancarios). (Tanenbaum, 2003).

Esta capa hace las veces de un traductor para que los equipos hablen el mismo idioma ya que permite cifrar datos y comprimirlos, cuando dos equipos presentan diferentes representaciones de datos la capa de presentación realiza una conversión para que los dos equipos tengan una interpretación similar de estos datos y puedan realizar sus acciones sin problemas de comunicación.

- **La Capa De Aplicación**

Esta capa contiene varios protocolos que los usuarios requieren con frecuencia. Un protocolo de aplicación de amplio uso es HTTP (Protocolo de Transferencia de Hipertexto), que es la base de *World Wide Web*. Cuando un navegador desea una página Web, utiliza este protocolo para enviar al servidor el nombre de dicha página. A continuación, el servidor devuelve la página. Otros protocolos de aplicación se utilizan para la transferencia de archivos, correo electrónico y noticias en la red. (Tanenbaum, 2003).

En la capa de aplicación se manejan distintos protocolos y aplicaciones, cada vez se van implementando nuevas aplicaciones y por esta razón el

número de protocolos crece muy rápido. A continuación se muestran estos protocolos:

- HTTP (*HyperText Transfer Protocol*)
- FTP (*File Transfer Protocol*)
- SMTP (*Simple Mail Transfer Protocol*)
- POP (*Post Office Protocol*)
- SSH (*Secure Shell*)
- Telnet

Para la facilitar la administración de una red en el nivel de aplicación se manejan los siguientes protocolos:

- SNMP (*Simple Network Management Protocol*)
- DNS (*Domain Name Server*)

2.4.2. MODELO TCP/IP

Se denomina globalmente la familia de protocolos TCP/IP. Esta familia consiste en una extensa colección de protocolos que se han especificado como estándares de Internet por parte de IAB (*Internet Architecture Board*).

El modelo Internet gira en torno a la pila de protocolos TCP/IP, en donde IP es un protocolo que proporciona mecanismos de interconexión entre redes de área local y TCP proporciona mecanismos de control de flujo y errores entre los extremos de la comunicación. En figura 13 se puede observar las capas del Modelo TCP/IP, y a continuación se hace una descripción de ellas:

- **Acceso a la Red**

Acceso a la red es la primera capa del modelo TCP/IP y es la que permite transmitir datos y especificar la forma en que cada paquete debe enrutarse o

direccionarse, sea cual sea el tipo de red utilizada para el flujo de tráfico, está conectada mediante telefonía, datos y cualquier tipo de conexión a la red.

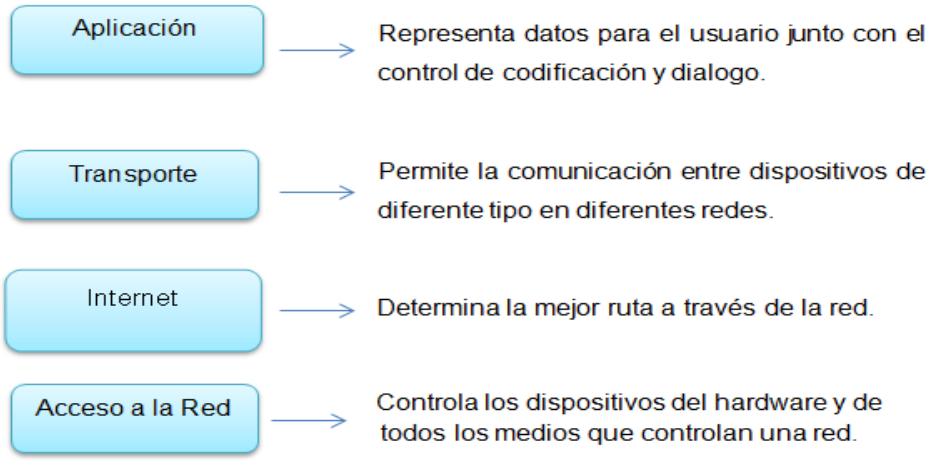


Figura 13. Capas del Modelo TCP/IP

- **Capa de Internet**

Todos estos requerimientos condujeron a la elección de una red de conmutación de paquetes basada en una capa de interred no orientada a la conexión. Esta capa, llamada capa de interred, es la pieza clave que mantiene unida a la arquitectura. Su trabajo es permitir que los hosts inyecten paquetes dentro de cualquier red y que éstos viajen a su destino de manera independiente (podría ser en una red diferente). Tal vez lleguen en un orden diferente al que fueron enviados, en cuyo caso las capas más altas deberán ordenarlos, si se desea una entrega ordenada. (Tanenbaum, 2003).

- **Capa de Transporte**

La capa que está arriba de la capa de interred en el modelo TCP/IP se llama capa de transporte. Está diseñada para permitir que las entidades iguales en los hosts de origen y destino puedan llevar a cabo una conversación, tal

como lo hace la capa de transporte OSI. Aquí se han definido dos protocolos de transporte de extremo a extremo. El primero, TCP (Protocolo de Control de Transmisión), es un protocolo confiable, orientado a la conexión, que permite que un flujo de bytes que se origina en una máquina se entregue sin errores en cualquier otra máquina en la interred.

Divide el flujo de bytes entrantes en mensajes discretos y pasa cada uno de ellos a la capa de interred. En el destino, el proceso TCP receptor reensambla en el flujo de salida los mensajes recibidos. TCP también maneja el control de flujo para asegurarse de que un emisor rápido no sature a un receptor lento con más mensajes de los que puede manejar. (Tanenbaum, 2003).

- **Capa de Aplicación**

El modelo TCP/IP no tiene capas de sesión ni de presentación, no se han necesitado, por lo que no se incluyen. La experiencia con el modelo OSI ha probado que este punto de vista es correcto: son de poco uso para la mayoría de las aplicaciones.

En esta capa se manejan dos tipos de protocolos: TCP y UDP. Dentro del grupo TCP se tiene los siguientes:

- HTTP
- FTP
- SMTP
- DNS

Dentro del grupo UDP se pueden indicar:

- TFTP
- DNS

La capa de aplicación define las aplicaciones de red y los servicios de Internet estándar que puede utilizar un usuario. Estos servicios utilizan la capa de transporte para enviar y recibir datos. Existen varios protocolos de capa de aplicación. En la lista siguiente se incluyen ejemplos de protocolos de capa de aplicación:

- Servicios TCP/IP estándar como los comandos ftp, tftp y telnet.
- Comandos UNIX "r", como rlogin o rsh.
- Servicios de nombres, como NIS o el sistema de nombre de dominio (DNS).
- Servicios de directorio (LDAP).
- Servicios de archivos, como el servicio NFS.
- Protocolo simple de administración de red (SNMP), que permite administrar la red.
- Protocolo RDISC (*Router Discovery Server*) y protocolos RIP (*Routing Information Protocol*). (ORACLE, 2010).

A este nivel se manejan algunos términos importantes en networking:

- **Intranet**. En este entorno, a la intranet se le llama nivel de red local o, simplemente, nivel de red. Está formado por una red LAN, o WAN (de conexión punto a punto) homogénea. Son accesos a redes internas de una oficina, empresa que acceden a información de uso y manejo exclusivo de un dominio.

- **Extranet** (nivel de *Internetworking*). Este nivel confiere unidad a todos los miembros de la red y, por consiguiente, es el que permite que todos se puedan interconectar, con independencia de si se conectan a la misma por medio de línea telefónica, ISDN o una LAN Ethernet. El direccionamiento y la asignación de direcciones constituyen sus principales funciones. Todos los equipos conectados a Internet implementan este nivel.

En la capa de aplicación se manejan números de puertos TCP y UDP deben utilizar para comunicarse con las capas superiores, porque son lo que mantiene un registro de las distintas conversaciones que atraviesan la red al mismo tiempo. Los números se asignan dinámicamente por el host de origen y será equivalente a un número a partir de 1024. Los números por debajo de 1024 se consideran los números de puerto bien conocidos y están definidos en el RFC 3232 mientras que los números 1024 y superiores son utilizados por las capas superiores de establecer sesiones con otros host.

En la tabla 1 se muestra un ejemplo de los puertos más conocidos y utilizados directa o indirectamente en un proceso comunicación de los datos.

Tabla 1. Puertos y servicios TCP/UDP

Puerto	**Servicio o aplicación**
21	FTP (*File Transfer Protocol*)
23	Telnet (Acceso a terminal remoto)
25	SMTP (*Simple Mail Transfer Protocol*)
53	DNS (Sistema de nombre de dominio)
70	Servicio Gopher
79	Servicio Finger
80	HTTP (*Hyper Text Transfer Protocol*)
110	POP3 (*Post Office Protocol*)
119	NNTP (*Network News Transfer Protocol*)
143	IMAP (*Internet Message Access Protocol*)
161	SNMP (*Simple Network Management Protocol*)
546	DHCP (Dynamic Host Configuration Protocol)

2.4.3. MODELO TRES CAPAS DE CISCO

La jerarquía tiene muchos beneficios en el diseño de las redes y nos ayuda a hacerlas más predecibles. En sí, definimos funciones dentro de cada capa, ya que las redes grandes pueden ser extremadamente complejas e incluir múltiples protocolos y tecnologías; así, el modelo nos ayuda a tener un modelo fácilmente entendible de una red y por tanto a decidir una manera apropiada de aplicar una configuración. A continuación en la figura 14 se presenta las capas y sus funciones típicas.

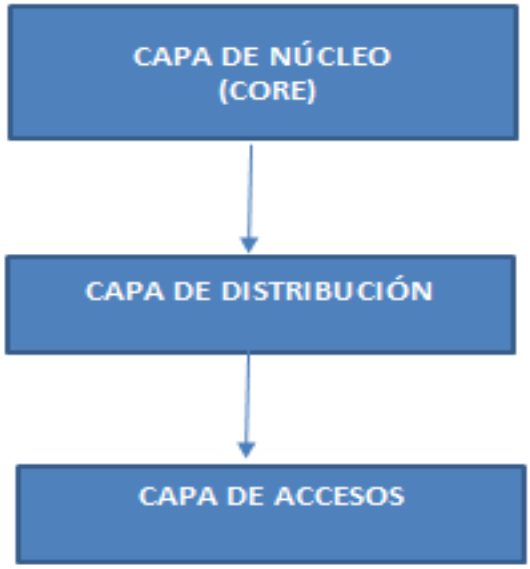


Figura 14. Modelo de tres capas de CISCO

- **La capa de Núcleo *(core layer).***

Es la capa principal, está es la encargada y responsable de transportar grandes cantidades de tráfico de forma confiable y veloz por lo que la latencia y la velocidad son factores importantes en esta capa. El tráfico que transporta es común a la mayoría de los usuarios, pero el tráfico se procesa

en la capa de distribución que a su vez envía las solicitudes nuevamente a la capa de núcleo si es necesario.

- **La capa de Distribución *(distribution layer).***

En esta capa se comunican la capa de núcleo y la capa de accesos, las funciones principales de la capa de distribución es proveer el enrutamiento, el filtrado, el acceso a la WAN y si es necesario determinar que paquetes deben llegar a la capa de núcleo, responder inmediatamente las solicitudes por ejemplo, como traer un archivo desde un servidor y seleccionar la mejor ruta para su recepción. En esta capa se aplican normas y políticas de red.

- **La capa de Acceso (*access layer*)**

En esta capa se controla a los usuarios y el acceso de grupos de trabajo (*workgroup access*) o los recursos de *internetwork*, se denomina también como capa de escritorio (*desktop layer*). Los recursos más utilizados por los usuarios deben ser ubicados localmente, y el tráfico de servicios remotos es manejado aquí, y entre sus funciones principales están definir el control de acceso y políticas de red, creación de dominios de colisión separados (segmentación), y la conectividad de grupos de trabajo en la capa de distribución (*workgroup connectivity*).

2.4.4. MODELO SDN *(Software Defined Networking)*

El modelo SDN está basado en un servidor de control llamado OpenFlow, el cual verifica el flujo de datos en la red dependiendo de su configuración, este servidor va conectado al o a los equipos de infraestructura (switch), los cuales reciben órdenes del controlador y las aplica para la administración de la red.

A continuación en la figura 15 se presenta las capas del modelo SDN y la descripción de cada una de sus funciones.

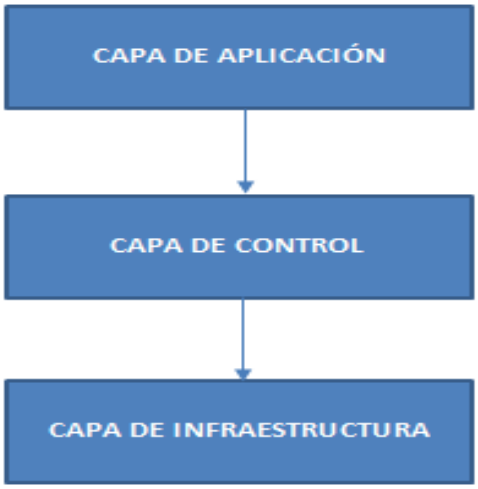


Figura 15. Modelo SDN

- **Capa de Aplicación**

En esta capa se presentan las aplicaciones de negocio a los usuarios finales permitiendo simplificar y automatizar las tareas de configuración y gestión de nuevas solicitudes y servicios en la red.

- **Capa de Control**

En la capa de control o plano de control es donde se configura y controla los nodos de red para dirigir correctamente los flujos de datos, se selecciona el mejor camino para el tráfico de paquetes. Realiza funciones internas del controlador SDN y de los diferentes elementos de red, brinda la funcionalidad de un control centralizado que supervisa el comportamiento de reenvío de información en la red a través de una interfaz abierta.

- **Capa de Infraestructura**

Constituyen los nodos de una red que realizan la conmutación y el direccionamiento de paquetes. En esta capa se proporciona un acceso abierto y programable al diseño de la topología personalizada de red SDN.

2.4.5. MODELO CICLO DE VIDA

Un modelo del ciclo de vida en general es una serie de etapas que atraviesa un sistema desde su concepción, creación, desarrollo, ejecución, pruebas y mantenimiento del mismo. El modelo de ciclo de vida indica que todas las fases y actividades de un proyecto se desarrollan de forma lineal de acuerdo con las siguientes características:

- Describe las fases principales de desarrollo de software.
- Define las fases esperadas en el desarrollo de un sistema desde su inicio hasta su fin.
- Ayuda a administrar el progreso del desarrollo.
- Provee un espacio de trabajo para la definición de un proceso detallado de desarrollo de software.

- **MODELO CICLO DE VIDA EN CASCADA**

La figura 16 muestra cada una de las fases del modelo de ciclo de vida en cascada, en donde se puede observar que el inicio de una nueva fase ocurre cuando la anterior ha finalizado, generándose además una retroalimentación entre cada par de fases. (Laboratorio Nacional de Calidad del Software, 2009).

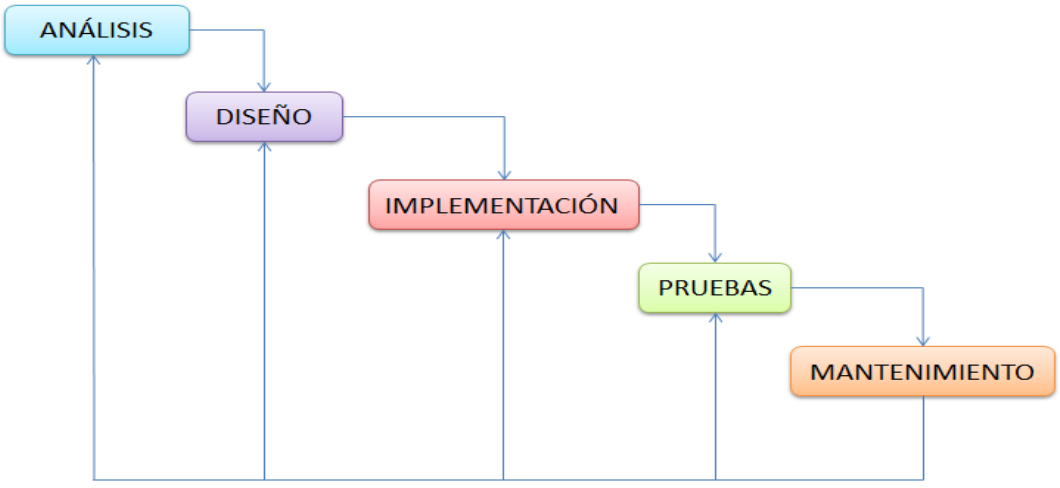


Figura 16. Modelo del ciclo de vida en cascada

Este modelo sigue un proceso secuencial y se define como modelo de ciclo de vida en cascada ya que sus fases se desarrollan ordenadamente hacia abajo una a partir de la otra, por lo tanto, cualquier error de diseño detectado en la etapa anterior conduce necesariamente al rediseño, aumentando los costos y tiempos de desarrollo.

Las fases del modelo de ciclo de vida en cascada son las siguientes:

- **Fase de Análisis:** En esta fase se analiza el problema a resolver o el objetivo a cumplir y se identifican las necesidades y requerimientos de los usuarios finales.

- **Fase de Diseño:** Una vez que se ha recopilado toda la información y definido con exactitud los requerimientos se procede a realizar el diseño lógico y físico de la red, así como también definir el modelo para el desarrollo del software.

- **Fase de Implementación:** A partir del análisis y diseño establecidos previamente, en esta etapa ya se procede a generar el código fuente del sistema mediante el uso de herramientas y técnicas de programación.

- **Fase de Pruebas:** Generar una serie de ensayos que garanticen el correcto funcionamiento de dicho programa bajo el mayor número de situaciones posibles y determinar si el sistema cumple con todos los requerimientos establecidos inicialmente.

- **Fase de Mantenimiento:** Esta es la etapa más sensible ya que se interactúa con el usuario y se comprueba si se cumple con lo establecido inicialmente, es importante mantener una estructura de actualización, verificación y validación que permita a dicho programa

ser útil y mantenerse actualizado según las necesidades o requerimientos planteados durante su vida útil.

Ventajas del modelo ciclo de vida en cascada

- El modelo de ciclo de vida en cascada es aplicable a proyectos estables donde se conocen claramente las actividades a realizar en cada una de las fases antes de su implementación.
- Fases independientes donde cada una debe completarse y terminarse antes de empezar con otra.
- Modelo sencillo a seguir ya que posee un orden de desarrollo en sus fases.
- Ahorra costos ya que se define una lista de actividades específicas a elaborarse en cada fase.

Desventajas del modelo ciclo de vida en cascada

- Algunos tipos de proyectos no necesariamente se deben desarrollar de forma secuencial, esto puede producir una mala estructuración de las fases, generando gastos mayores y tiempos fuera de los establecidos.
- El cliente no siempre tiene establecidas todas las actividades a realizarse desde el principio, esto puede producir cambios en el desarrollo del proyecto.
- Este modelo es aplicable para proyectos pequeños puesto que en éstos se tiene claramente definidas las actividades.
- El producto final se obtiene al finalizar el ciclo de vida.
- Si surgen cambios en un determinado momento es complicado regresar a etapas anteriores para realizar sus modificaciones.
- El producto final obtenido puede que no refleje todos los requisitos del usuario.

2.5. DIRECCIONAMIENTO IPV4

Las direcciones IPv4 se dividen en 5 clases: A, B, C, D y E, siendo las tres primeras las que se utilizan para configurar a los host. En la tabla 2 se muestra la descripción de cada clase y sus principales características.

Tabla 2. Tipos de clases de direcciones IP

CLASE	DESCRIPCIÓN	RANGO DEL PRIMER OCTETO EN BINARIO	RANGO DEL PRIMER OCTETO EN DECIMÁL	NÚMERO DE DIRECCIONES DE HOST
A	Contiene 8 bits para direcciones de red y 24 bits para host. Son utilizadas para redes grandes.	0	1 – 126	16,777,214
B	Contiene 16 bits para direcciones de red y 16 bits para host. Son utilizadas para redes medianas y grandes.	10	128 – 191	65,534
C	Contiene 24 bits para direcciones de red y 8 bits para host. Son utilizadas para redes pequeñas.	110	192 – 223	254
D	Usado para direccionamiento multicast	1110	224 – 239	No aplicable.
E	Utilizado para investigación	11110	240 – 255	No aplicable.

2.5.1. Las Redes de Clase A

En la figura 17 se muestra la arquitectura de una dirección IP clase A, siendo el rango de direcciones clase A privadas: 10.0.0.0 – 10.255.255.254, cualquier dirección fuera de este rango se denomina una IP clase A pública.

RED	HOST	HOST	HOST
8 bits	8 bits	8 bits	8 bits

Figura 17. Red Clase A

2.5.2. Las Redes de Clase B

En la figura 18 se muestra la arquitectura de una dirección IP clase B, siendo el rango de direcciones clase B privadas: 172.16.0.0 – 172.31.255.254, cualquier dirección fuera de este rango se denomina una IP clase B pública.

RED	RED	HOST	HOST
8 bits	8 bits	8 bits	8 bits

Figura 18. Red Clase B

2.5.3. Las Redes de Clase C

En la figura 19 se muestra la arquitectura de una dirección IP clase C, siendo el rango de direcciones clase C privadas: 192.168.0.0 – 192.168.255.254, cualquier dirección fuera de este rango se denomina una IP clase C pública.

RED	RED	HOST	HOST
8 bits	8 bits	8 bits	8 bits

Figura 19. Red Clase C

2.6. VIRTUALIZACIÓN DE SEVIDORES

Antiguamente, para configurar diversos servicios de red se necesitaba el uso de varias computadoras en los cuales se instalaba un sistema operativo servidor y se levantaba el o los servicios requeridos. Actualmente se hace lo mismo pero con solo un equipo servidor en el que puedan residir de forma lógica varios servicios a través del uso de máquinas virtuales.

"La abstracción del sistema operativo y de las aplicaciones del hardware físico posibilita un entorno de servidores más simplificado, ágil y económico. Mediante la virtualización de servidores, en un único servidor se pueden ejecutar varios sistemas operativos como máquinas virtuales, todos con acceso a los recursos informáticos del servidor subyacente.

La mayoría de los servidores funcionan a menos de un 15 % de su capacidad, lo que fomenta la complejidad y la proliferación de servidores. La virtualización de servidores se ocupa de estas ineficiencias. VMware vSphere proporciona una completa plataforma de virtualización de servidores que logra:

- *Un aumento del 80 % de la utilización de los recursos de servidor;*
- *Un ahorro de hasta el 50 % en gastos operativos y de capital;*
- *Un índice de consolidación de servidores de 10:1 o superior"*. (VMWARE, 2015).

Una máquina virtual no afecta o alterna de ninguna manera a los procesos o configuraciones de la máquina real ya que se maneja en entornos virtuales es decir como si fuese una máquina totalmente aislada e independiente. Es una buena herramienta para entender procesos y nuevas características de manera simulada y posteriormente si se requiera el caso implementar toda la simulación de manera real.

2.6.1. Virtualización de Redes

La virtualización de redes permite que ciertos tipos de componentes puedan compartir su capacidad y sus recursos para desempeñar simultáneamente múltiples funciones. Dicho en otras palabras, una red virtual es una red de interconexión compuesta por un conjunto de dispositivos virtuales. Está virtualización permite realizar un enfoque de lo que será una futura red de computadoras, cada red virtual está aislada y tiene sus propios protocolos y su gestión individual. Una red virtual también es un conjunto de dispositivos lógicos conectados de acuerdo a ciertas topologías definidas por el administrador de la red.

2.8. REDES DEFINIDAS POR SOFTWARE

Redes definidas por software (SDN) nacen debido al gran crecimiento de las redes de datos y por lo tanto el incremento de recursos físicos, técnicos, humanos y económicos para la mantener la funcionalidad de éstas, por esta razón los administradores consideran cambiar e innovar la infraestructura de sus redes añadiendo y creando nuevas tecnologías para los enlaces de comunicación. Los administradores tendrán el absoluto control y monitoreo de las redes de comunicación, desde los inicios de su infraestructura hasta los cambios y crecimiento de la misma. Para entender acerca de la tecnología de SDN a continuación se hará una definición de ella, de su arquitectura, del protocolo OpenFlow y posteriormente de la herramienta de simulación de un prototipo denominada Mininet. (Vivek, 2013).

Una red SDN optimiza y simplifica las operaciones de red al unirse más estrechamente entre aplicaciones, los servicios y los dispositivos de red, ya sea de forma real o virtual. Una razón para el cambio a SDN es que permite la programación de interfaces en los equipos lógicos de red que se necesiten configurar. SDN facilita que organizaciones reemplace una interfaz manual en un equipo de conexión en red por una interfaz lógica, generando la

automatización de las tareas como su configuración, dirección de políticas y también permitiendo que la red responda a requisitos de aplicación. El control sobre una red SDN se ejecuta mediante reglas programáticas que definen patrones y comportamientos del flujo de tráfico. Las reglas se programan desde un dispositivo denominado servidor controlador para SDN (NOX, POX, Beacon y Floodling). La gestión de este tipo de red se encuentra centralizada sobre este dispositivo que permite manejar toda la inteligencia de la red. Más adelante se detallarán los diferentes tipos de controladores.

Investigaciones realizadas por la multinacional Google después de haber implementado SDN en su red interna se destacan los siguientes aspectos:

- La red SDN está lista para su uso en el mundo real.
- Simplifica la gestión de la red.
- Capacidad de administración mejorada.
- Menos costo de implementación.
- Tiempo de interacción mucho más rápido.
- Menos pérdida y degradación de paquetes.
- Redes más determinísticas.
- Alto grado de estabilidad.
- Mejoramiento del enrutamiento de paquetes.

A pesar de ello, muchos diseñadores de redes ven a esta tecnología todavía muy tierna, por lo que no se atreven a implementarla y a explotarla como se debe, pues aducen que OpenFlow no está completo para el diseño de una red real por las siguientes razones:

- Maneja un conjunto reducido y básico de instrucciones.
- Un sistema centralizado definido por software debe ser lento.
- Altos tiempos de convergencia en casos de fallo.
- Otros.

Si esto fuera así, Google no arriesgaría tiempo y dinero en investigar y aplicar la tecnología SDN en su infraestructura y brindar el servicio que ofrece a la comunidad a nivel mundial.

En la figura 20 se observa una sencilla estructura de SDN, que se compone de un controlador, unos dispositivos finales y un protocolo para comunicarse con los dispositivos.

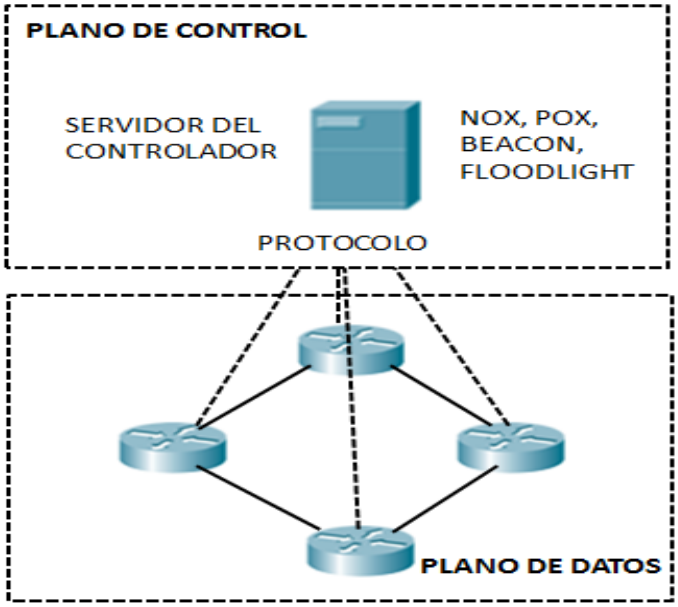


Figura 20. Estructura SDN

2.8.1 Arquitectura de SDN

La arquitectura de una red SDN se compone de un servidor controlador el cual emite e inspecciona todas las órdenes dentro del flujo de datos en una red, distintos dispositivos que están interconectados, los cuales reciben, obedecen y aplican las ordenes emitidas por el controlador y un protocolo para comunicarse.

En la figura 21 se puede observar detalladamente la arquitectura SDN, el controlador es el encargado en manejar el plano de control comunicándose

con los dispositivos finales a través del protocolo OpenFlow el cual se detallará posteriormente.

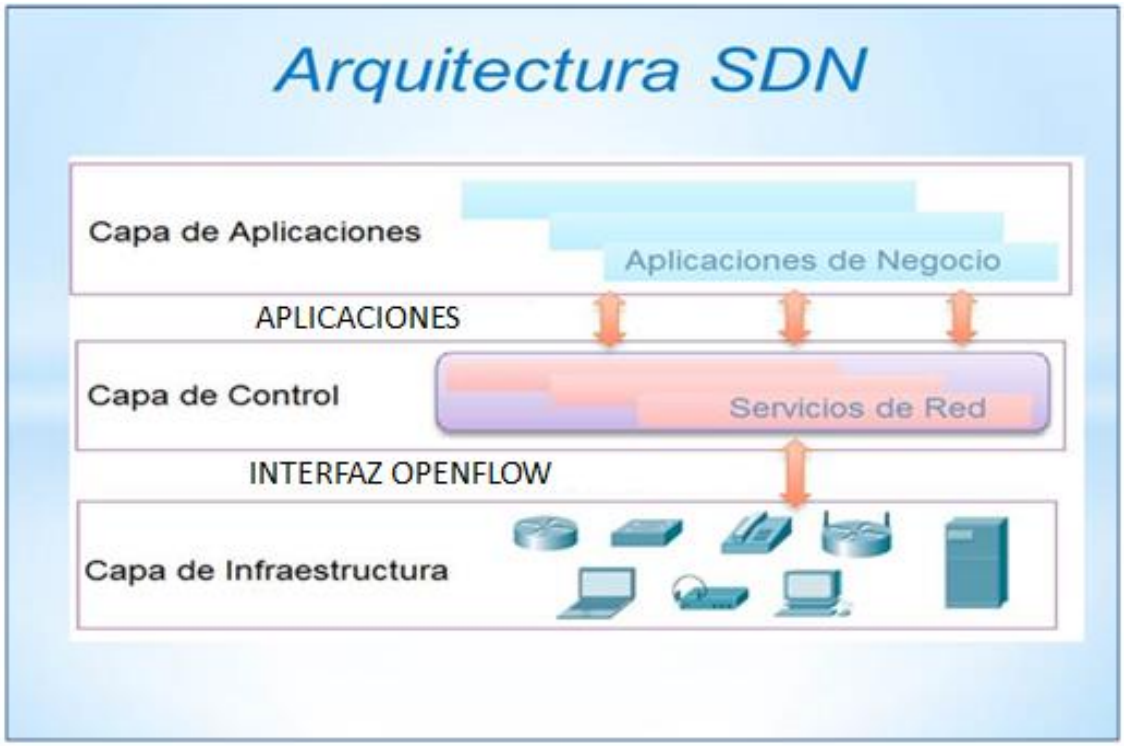


Figura 21. Arquitectura SDN

- **Ventajas de la Arquitectura SDN**

La arquitectura de SDN tiene múltiples beneficios para mayor orden y control de las redes, a continuación se detallarán sus ventajas:

- Se administra la red a través de un solo dispositivo denominado servidor controlador SDN, en el cual se programan todas las reglas de circulación de tráfico.
- Mayor seguridad, SDN genera una confiabilidad en el manejo central de dispositivos.
- Mayor satisfacción al cliente ya que las redes son adaptables a los cambios y nuevos requerimientos que necesite para una determinada actividad o aplicación.

2.8.2 Controladores SDN

Es un núcleo central de SDN donde se maneja y opera toda la lógica de la red, es aquí donde el administrador definirá todo tipo de reglas y políticas de la misma, por ejemplo, en los diferentes equipos o dispositivo de red se necesitan un *firmware*, en las SDN pasa exactamente lo mismo, estas redes necesitan un componente o un cerebro llamado controlador que completa la funcionalidad de SDN, existen varios tipos de componentes (NOX, POX, Beacon, Floodlight, entre otros), la diferencia entre todos ellos es el lenguaje de programación con el que se establecerán las reglas para los flujos de sus datos, en este cerebro o controlador el administrador de la red se encarga en definir normas para administrar el flujo de tráfico de datos que se genere en la red, la principal diferencia con las redes tradicionales es que no hay que esperar que el fabricante implemente estas reglas o actualizaciones.

- ***NOX***

NOX es uno de los principales componentes que deben tener las redes definidas por software, este es el cerebro o controlador que realiza las funciones de escoger y tomar la decisión sobre cuáles son las mejores rutas disponibles para que mediante éstas, los flujos de paquetes pueden viajar desde un origen hacia un destino, almacena datos, estadísticas, políticas, reglas y toda la información que contiene una red. Es uno de los primeros controladores con el que se realizaron pruebas en redes SDN, obteniendo buenos resultados especialmente en el rendimiento de la red.

NOX fue desarrollado por Nicira Networks y donado a la comunidad científica de desarrolladores en 2008, este componente es compatible con sistemas operativos Linux como Ubunto y Debbian. A continuación de detallarán las principales características de NOX:

- Incluye aplicaciones y soporte específicamente para C++.

- Entrada síncrona y asíncrona, que permite controlar entradas y salidas de grandes flujos de datos dentro de la red SDN en cualquier instante.

- Está orientado a recientes distribuciones de Linux como Ubuntu 11.10 y 12.04, pero la implementación es fácil y es posible realizar sobre Debian, Red Hat y CentOs.

En la actualidad NOX tiene disponibles dos versiones:

- **NOX-*Classic***: Esta versión tiene soporte para Python y C++ y otras aplicaciones. Actualmente se encuentra en desuso y ya no se realiza desarrollo en esta línea. No se recomienda para nuevos usuarios el uso de esta versión, sin embargo su rango de módulos es más amplio que la nueva versión.

- **NOX**: Es la nueva versión solo contiene soporte exclusivo para C++, en esta versión ya no contiene soporte para Python, contiene más aplicaciones que NOX-*Classic*, tiene una gran base de código más limpio y es más rápido. Su rango de aplicaciones es más reducido que la versión clásica.

Los objetivos principales del controlador NOX son:

- Proveer una plataforma que permita a los desarrolladores realizar innovaciones en el manejo de red.

- Tener un manejo centralizado de dispositivos de conectividad, como switches, control de admisiones a nivel de usuarios y un núcleo de políticas para la seguridad de la red.

- ***POX***

Este es un controlador similar a NOX con la diferencia de que utiliza Python y C++ como lenguajes de programación, el objetivo de crear este controlador fue para permitir un desarrollo rápido y dinámico en reglas de flujo de tráfico. A continuación de detallarán las principales características de POX*:*

- Posee una Interfaz OpenFlow "Pythonic" llamada así por encontrarse en un ambiente de desarrollo de Python.

- Se puede combinar con “PyPy” que es un entorno de ejecución para una fácil implementación de programas escritos en Python por lo tanto es compatible con plataformas como Linux, Windows, MAC OS*.*

- Contiene componentes reutilizables para la selección de rutas, caminos óptimos, descubrimiento de topologías, otros.

- Soporta la GUI (Interfaz gráfica de usuario) y herramientas de NOX y la misma visualización.

- Se desempeña con un mejor rendimiento en comparación con las aplicaciones de NOX.

- ***BEACON***

Beacon es un software basado en Java donde se programan reglas de flujo en controladores SDN de manera rápida, multiplataforma y modular. A continuación de detallarán las principales características de *Beacon* (Erickson, 2013).

- **Estable**: Ha estado en desarrollo desde principios de 2010, y ha sido utilizado en varios proyectos de investigación, cursos de redes y pruebas de implementaciones y en *Data Centers*.

- **Multiplataforma**: Beacon está escrito en Java y gracias a su compatibilidad con cualquier dispositivo puede correr en muchas plataformas, en especial servidores Linux, ya que se ejecuta sobre la máquina virtual propia de Java JVM (*Java Virtual Machine*).

- **Código Abierto**: Beacon opera bajo la licencia GPL v2 y la licencia de la Universidad de Stanford FOSS v1.0.

- **Dinámico**: Los paquetes de código en Beacon se pueden iniciar, parar refrescar e instalar en tiempo de ejecución, sin interrumpir otros paquetes no dependientes (es decir, sin necesidad de desconectar los switches).

- **Desarrollo rápido**: Beacon es fácil de poner en marcha ya que Java y Eclipse simplifican el desarrollo y la depuración de sus aplicaciones.

- **Rápido**: Beacon es rápido gracias a los múltiples hilos de ejecución.

- ***FLOODLIGHT***

Al igual que Beacon, Floodlight también está escrito en java y corre sobre su máquina virtual propia de Java JVM (*Java Virtual Machine*). A continuación de detallarán las principales características de Floodlight*:*

- Trabaja con OpenFlow en switches tanto físicos como virtuales.

- Fácil de configurar con dependencias mínimas.
- Puede manejar redes OpenFlow en islas, es decir que agrupa dispositivos que entienden este protocolo, conectado a una red tradicional que no entiende OpenFlow.
- Requiere Java Developement Kit (JDK) y ANT de Apache, son herramientas utilizadas en Java para realizar tareas repetitivas como ejemplo, bucles de repetición.
- Posee una licencia Apache que permite usar Floodlight para cualquier propósito en específico.

A continuación, la tabla 3 presenta un cuadro comparativo entre los tipos de controladores compatibles con OpenFlow.

Tabla 3. Controladores compatibles con OpenFlow

Controlador	Lenguaje de Programación	Desarrollador
NOX	C++	Nicira
POX	Python	Nicira
Beacon	Java	Stanford
Floodlight	Java	BigSwitch

2.8.3 Proyectos aplicados con SDN

Las SDN se aplicarán en muchos campos, de hecho existen algunos proyectos en versión demo que se están comprobados, por ejemplo:

- Virtualización de redes (*FlowVisor*).
- Creación de prototipos de hardware (*OpenPipes*).
- Balanceo de carga (*PlugNServe*).
- Ahorro de energía (*ElasticTree*).
- Movilidad (*MobileVMs*).
- Ingeniería de tráfico (*Aggregation*) y Video inalámbrico (*OpenRoads*).

2.8.4 Clasificación de redes SDN

Las redes SDN se pueden clasificar en dos tipos: Centralizada y Distribuida. A la hora de implementar la red se debe decidir, desde un punto de vista funcional, cuál de éstas será la más óptima.

- **Centralizada**

En la figura 22 se puede observar la estructura de una red centralizada en OpenFlow, la misma que posee un único controlador y todos los Switches lógicos de la red se conectan a él.

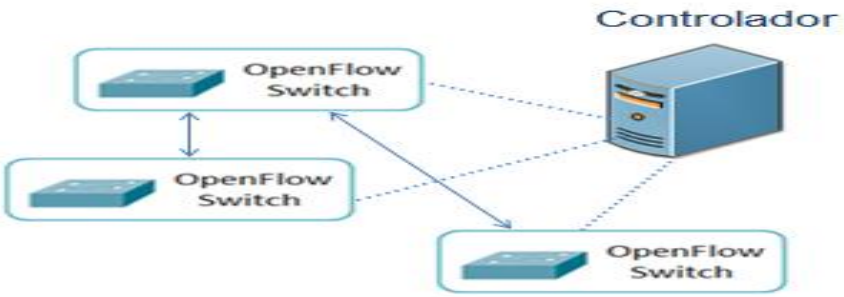


Figura 22. Red Centralizada

- **Distribuida**

En la figura 23 se puede observar la estructura de una red distribuida en OpenFlow, donde existen varios controladores entre los que se reparten la gestión de los nodos de red.

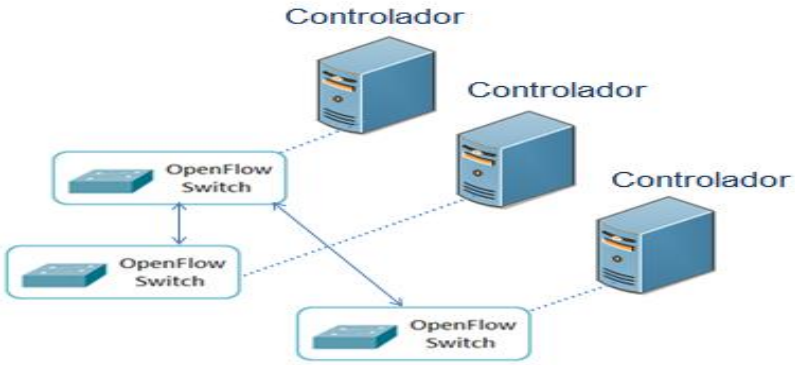


Figura 23. Red Distribuida

2.8.5. Arquitectura de red tradicional Vs. Arquitectura SDN.

Las redes físicas han evolucionado de forma rápida en los últimos años, pero hoy en día se deben optar por otras tecnologías que permitan resolver rápidamente y a menor costo, los nuevos requerimientos de los usuarios, por lo que poco a poco las redes definidas por software se están posicionando en el mundo de las comunicaciones de información.

A continuación, la tabla 4 presenta un cuadro comparativo entre la arquitectura de red tradicional y la arquitectura SDN.

Tabla 4. Arquitectura de red tradicional Vs. Arquitectura SDN

RED TRADICIONAL	REDES DEFINIDAS POR SOFTWARE
Los protocolos se manejan de manera aislada y separada, y se resuelven problemas en concreto.	Permiten manejar redes más dinámicas, dependiendo las necesidades y requerimientos de las aplicaciones actuales.
Se administra la configuración en cada equipo de manera independiente y separada.	Se administran las reglas y comandos mediante un solo controlador, añadiendo o quitando comandos de acuerdo a las necesidades del administrador.
La calidad de servicio es implementada manualmente en cada dispositivo, por el administrador de la red de acuerdo a las necesidades del usuario.	La calidad de servicio se implementa e introduce de una manera más rápida y automática sin necesidad de configurar cada dispositivo por separado.
La seguridad se configura de manera manual en cada dispositivo, por lo tanto siempre quedan brechas para un posible ataque o vulneración de la red.	La seguridad en SDN es centralizada es decir, previene el ataque y evita políticas inconsistentes brechas de seguridad.
Hay que esperar que los fabricantes implementen nuevas actualizaciones en los equipos para poder acceder a ellos.	El administrador realiza las actualizaciones y si lo considera conveniente innova las características de la red mediante la programación de reglas que definen patrones y flujos de datos en SDN.
Requiere gran infraestructura física (edificios), tecnológica (*Data Center*), humana (profesionales de TI) y económica para funcionar.	Son económicas, no se necesita de grandes centros de datos y se requiere de una cantidad mínima de profesionales TI.

2.9. PROTOCOLO OPENFLOW

OpenFlow es un protocolo que permite el reenvío y la separación del plano de control en una red basado en el flujo, proporciona una nueva flexibilidad en la innovación de la red. OpenFlow ahora está encontrando su camino en las aplicaciones comerciales. Sin embargo, esto plantea nuevos retos, como las cuestiones de escalabilidad y el rendimiento. Con OpenFlow configurado en switches, los administradores pueden utilizarlo para aprovechar las tablas de direccionamiento y controlar la estructura y el flujo de tráfico. OpenFlow es la primera interfaz de comunicaciones estándar definida entre las capas de control e infraestructura de una arquitectura de SDN. Además permite acceso directo a la configuración de los modos de reenvío de paquetes en switches, tanto físico como virtual.

Es la ausencia de una interfaz abierta al plano de reenvío que ha llevado a la caracterización de este dispositivo de red como monolítico, cerrado, y *mainframe*. Se necesita un protocolo como OpenFlow para mover el control de la red fuera de los switch, centralizando lógicamente al software de control. OpenFlow utiliza dos protocolos TCP y SSL en la capa 4 y 5 del modelo OSI para la comunicación entre el plano de control y el controlador, con estos protocolos se pueden realizar proyectos en redes de área amplia (WAN). (Heller, 2011).

2.9.1 Mensajes de OpenFlow

La estructura de OpenFlow está diseñada para emitir órdenes, establecer la comunicación y generar comandos correspondientes según los mensajes emitidos. Una tabla de flujos de OpenFlow consta de los siguientes campos en sus paquetes:

- **Header:** Es el encabezado de los paquetes, aquí se define y caracteriza el flujo, tipo de información, orden o acción que el controlador va a realizar.

- **Type:** Muestra los diferentes tipos de mensaje, tales como:

    - Mensajes inmutables.
    - Mensajes de configuración.
    - Mensajes asíncronos.
    - Mensajes con comandos al controlador.
    - Mensajes estadísticos.
    - Mensajes de barrera.
    - Mensajes de configuración de colas. (Blandón, 2013).

2.9.2 Tabla de flujos (*flow-table*)

Una tabla de flujos tiene asignada una serie de entradas con una acción u orden específica para cada una, de esta manera se indica al switch como gestionar el flujo de entrada (*flow-entry*). Una tabla de flujos se compone por diferentes entradas, cuyas características se muestran en la tabla 5.

Tabla 5. Entrada de tabla de flujos OpenFlow

ENTRADAS	DESCRIPCIÓN
Match Fields	Detectan los paquetes que se componen entre el flujo de entrada y las cabeceras de cada paquete.
Priority	Orden de preferencia del flujo de entrada.
Counters	Su función es actualizar los paquetes que coincidan.
Instructions	Estas órdenes modifican un conjunto de acciones.
Timeout	Tiempo en el que expira un flujo de entrada en un switch.
Cookie	Valor usado en el controlador SDN, este no procesa flujos.

2.10. MININET

Es una red local diseñada principalmente para el soporte de dispositivos de alta velocidad. Esta red de multimedia es muy transparente, ahora en funcionamiento en una forma de prototipo inicial, usa la técnica de conmutación de paquetes y tiene muchas características que dan un mecanismo de transporte de datos atractivo para conectar dispositivos de red.

2.10.1 Definición de Mininet

Mininet es un emulador de red que genera una colección de dispositivos finales, tales como switches, routers y enlaces en un solo core de Linux. Se utiliza la virtualización ligera para hacer que un solo sistema parezca una red completa. Los programas que se ejecutan pueden enviar paquetes a través de lo que se asemeja a una interfaz de Ethernet real, con una velocidad de enlace y con retardo. Los paquetes se procesan por lo que parece un verdadero switch de Ethernet, un router o *middlebox*, con una determinada cantidad de colas. (Velásquez, 2014).

Es una plataforma diseñada para la emulación de redes. Mininet permite la creación de hosts, switches y links en una máquina virtual o física. Se puede crear cualquier tipo de topología personalizada, basta con definir su código y generarlo ya que posee una API basada en Python incluida. Gracias a Mininet se pueden realizar prácticas, pruebas, simulaciones, depuraciones de forma experimental para el mayor aprendizaje, puede ejecutarse fácilmente y de manera amigable en cualquier computador incluso en una laptop ya que está disponible de manera gratuita.

Su instalación es fácil y se puede descargar directamente desde la página oficial de Mininet y configurase en una máquina virtual con cualquier versión de Linux, o instalarse directamente desde las últimas actualizaciones de

Ubunto o Fedora ya que traen nuevas versiones abiertas de switches virtuales.

2.10.2 Atributos de Mininet

- **Flexible:** Para nuevas topologías y funciones que se deben definir en el software por medio de lenguajes de programación, herramientas y determinados sistemas operativos que el usuario conozca.

- **Desplegable:** Debe tener una configuración y programación bien definidas de manera que no se necesite realizar ningún cambio en su configuración o programación.

- **Interactivo:** Las operaciones de red y gestión de la misma deben realizarse como si se estuviera trabajando en tiempo real.

- **Escalable:** Cada red debe permitir escalar cualquier número de switches sobre una única maquina física.

- **Realista:** Las operaciones, acciones, gestiones deben estar lo más aproximados a la realidad y en tiempo real.

- **Compatible:** Los prototipos deben ser independientes que se puedan realizar experimentos o pruebas de manera independiente o paralela.

2.10.3 Ventajas de Mininet

- Rapidez de arranque, segundos en vez de minutos.
- Mayor escalabilidad, cientos de switches y host.
- Instalación fácil.
- Interfaz amigable.

2.10.4 Topologías creadas por defecto en Mininet

Mininet proporciona compatibilidad con topologías personalizadas creadas en una API de Python, además entre sus funcionalidades viene integrada con prototipos por defecto de redes SDN, la creación y el acceso a éstas topologías es sencillo y de esta manera el administrador puede empezar a relacionarse con Mininet. Los tipos de topologías por defecto que se pueden ejecutar en Mininet son:

- *Single*: En esta topología existe un solo switch con **N** hosts. Donde **N** es el número de hosts o nodos deseados que se encuentren conectados al switch. El comando que se ejecuta para su creación es el siguiente:

  $sudo mn --topo single,N

- *Linear:* En esta topología existe un determinado número de switches interconectados de forma lineal y cada switch va conectado a un host. Donde **N** es el número de switches y hosts. El comando que se ejecuta para su creación es el siguiente:

  $sudo mn –topo linear,N

- *Tree* o árbol: En esta topología como su nombre lo indica es en forma de árbol, consta de dos variables **N** es el nivel de profundidad o *depth* estimada y **M** es la anchura o *fanout*. Donde **N** es el nivel switches interconectados y **M** es el número de hosts conectados a cada switch del último nivel. El comando que se ejecuta para su creación es el siguiente:

  $sudo mn --topo tree,depth=n,fanout=m

- *Custom* o personalizada*:* En esta topología cada administrador creará su topología de acuerdo a sus necesidades, para esto previamente se debe generar un script en una API de Python.

  Las topologías customizadas o personalizadas se encuentran almacenadas en el directorio ./mininet/custom desde donde se ejecutará en Mininet. El comando que se ejecuta para su creación es el siguiente:

$sudo mn –custom ALU_UTE.py –topo sdn

Los tipos de topología estudiados anteriormente solo son ejemplos para empezar a relacionarse con Mininet, pero para el presente Proyecto de Titulación de creará una topología del último tipo, para esto es necesario crear un script llamado ALU_UTE.py en Python y posteriormente importarlo a Mininet para su creación.

Se pueden desarrollar varios tipos de topologías, por la flexibilidad que posee Mininet se pueden implementar los siguientes:

- Topología en árbol: En esta topología los switches van conectados entre sí en varios niveles organizados jerárquicamente.

- Topología en estrella: En esta topología los switches van conectados como su nombre lo indica en forma de estrella.

- Topología lineal: En esta topología los switches van conectados entre sí en un solo nivel jerárquicamente.

2.11. API

En este proyecto de titulación para realizar la codificación del script de la red SDN personalizada se utilizará una API en Python.

API es un entorno de programación de aplicaciones, dicho en otras palabras es una interfaz en donde se pueden comunicar distintos componentes de software, en este caso será la que permita generar el código que permitirá implementar la red SDN personalizada.

2.12. PYTHON

Python ha sido diseñado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”.

Es un lenguaje de programación poderoso y fácil de aprender, cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

Python es un lenguaje gratuito y se puede descargar desde su página web oficial http://www.python.org. Tiene versiones para prácticamente cualquier plataforma en uso: sistemas PC bajo Linux, Microsoft Windows, Macintosh de Apple, etc. (Marzal, Gracia, 2009).

A continuación se explicarán sus principales características:

- Python es un lenguaje muy expresivo, es decir, los programas Python son muy compactos: un programa Python suele ser bastante más corto que su equivalente en lenguajes como C. (Python llega a ser

considerado por muchos un lenguaje de programación de muy alto nivel).

- La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación.
- Python es muy legible.
- El entorno de ejecución de Python detecta muchos de los errores de Programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos.
- Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.

Librerías de Python

Este lenguaje dispone de una gran cantidad de librerías, para la realización de diversos propósitos, alguna de ellas son:

- Acceso a Ficheros, manejo de cadenas.
- Servicios Web.
- Retoque de imágenes.
- Multimedia.
- Interfaces gráficas.
- XML
- Creación de PDFs.

METODOLOGÍA

3. METODOLOGÍA

Para el desarrollo de la simulación del rediseño de la red de switches de la UTE se ha tomado como referencia las fases del modelo de ciclo de vida en cascada, que inicia con el análisis y la recolección de información, modelamiento de la red SDN, simulación de la implementación y pruebas de conectividad.

3.1. MÉTODOS DE INVESTIGACIÓN

En este trabajo se utilizarán varios métodos de investigación científica entre teóricos y empíricos, cuya aplicación se resume en la tabla 6. Además se empezará realizando un análisis de las herramientas, técnicas, servicios, y lo que ofrecen las SDN hasta la actualidad, generando y adquiriendo los conocimientos necesarios para llegar a cumplir con los objetivos propuestos inicialmente que es la simulación del rediseño de la red de switches de la UTE aplicando redes SDN.

Tabla 6. Resumen de los métodos de investigación aplicados

METODOS DE INVESTIGACIÓN	**FASE DE LA INVESTIGACIÓN**			
	INTRODUCCIÓN	**MARCO TEORICO**	**METODOLOGÍA**	**ANALISIS Y RESULTADOS**
TEÓRICOS	• Inductivo • Deductivo	• Analítico • Sintético	• Inductivo • Deductivo	• Analítico
EMPIRICOS	X	X	X	• Simulación • Experimentación

3.2 MÉTODO DE DESARROLLO

Para el desarrollo de este proyecto de redes SDN se utilizará el modelo del ciclo de vida en cascada, el mismo que es adecuado para este tipo de proyectos, ya que se trata del desarrollo de un programa informático para la simulación del comportamiento básico de una red, por lo tanto es necesario empezar realizando el análisis, diseño, implementación, pruebas y mantenimiento del mismo.

3.2.1. Fase de Análisis

En esta fase se realizará un análisis de una parte de la arquitectura actual de la red la UTE del Campus Matriz Quito Av. Mariana de Jesús y Occidental, con el fin de conocer como está estructurada la red de switches y poder crear la simulación del rediseño aplicando la tecnología SDN; además se identificarán los requerimientos y componentes de hardware y software necesarios para llevar a cabo este proyecto, así como también las herramientas y configuraciones necesarias para el funcionamiento de la simulación de la red SDN en el software Mininet.

3.2.2. Fase de Diseño

- **Diseño Lógico**

Para el diseño de la red SDN se considerará el rango de host válido de la dirección IP clase A privada 10.0.0.0/8 ya que Mininet utiliza ésta para simular el comportamiento de una red SDN, se elaborará una tabla de direccionamiento IP para los correspondientes host configurados.

En el anexo 2, se indica la topología lógica de la red a simular, de acuerdo a la distribución física de la misma en el piso 1, corresponde a una topología en estrella extendida con apilación de switches, la misma que está

constituida por un servidor controlador, nueve switches con dos host interconectados a cada uno de ellos. Se establecerá un diámetro de la red SDN correspondiente a la red física actual de la UTE, se explicará que sucede con redundancia, se realizará la estimación de los costos y presupuestos que se invertirá en la realización e implementación para la simulación de redes SDN y finalmente se diseñará una planimetría en donde se definirán los host, switches y el servidor controlador a utilizarse en la simulación de red SDN.

3.2.3. Fase de Implementación

Para la implementación del prototipo se pondrá en marcha todos los procesos necesarios para realizar la simulación del rediseño de la red de switches, desde la instalación de una máquina virtual, el desarrollo de un prototipo de redes SDN en Python por ser el lenguaje de programación más compatible ya que posee librerías directas de Mininet haciendo más fácil su implementación, posteriormente se realizará la correcta validación a través del emulador de simulación Mininet, finalmente se podrá crear y ejecutar la topología seleccionada, añadiéndose los nodos y enlaces seleccionados por el administrador.

3.2.4. Fase de Pruebas

En esta fase se documentará las evaluaciones y pruebas de implementación en un entorno simulado de la red de switches de la UTE utilizando el software Mininet, se comprobará la funcionalidad mediante pruebas de conectividad y se verificará la comunicación en la topología y en la red SDN.

3.2.4. Fase de Mantenimiento

Esta fase no es aplicable para este trabajo ya que se trata de una simulación de red SDN y no de la implementación física de la misma.

ANÁLISIS DE RESULTADOS

4. ANÁLISIS DE RESULTADOS

En este capítulo se hace una descripción del proceso de implementación y desarrollo del prototipo de la red SDN, empezando desde la fase de virtualización, instalación y configuración del controlador, desarrollo del script o código fuente, creación y ejecución lógica de la red hasta las pruebas de conectividad.

A continuación se explicarán detalladamente los recursos y requerimientos necesarios para su implementación.

4.1. Fase de Análisis

En esta fase se realizó el análisis de una parte de la red física de los laboratorios de la UTE del bloque IDIC del Campus Matriz Quito Av. Mariana de Jesús y Occidental que se quiere simular mediante el concepto de redes SDN. En la figura 24 se muestra el diagrama bifilar de la red de switches de los laboratorios del piso 1, en donde se puede observar que cada laboratorio dispone de un switch independiente y cada uno de estos se encuentra conectado en cascada a través de los puertos de up link, formando un apilamiento de switches. La asignación de direcciones IP para las computadoras de cada laboratorio se lo realiza desde la administración a través de un servidor de DHCP.

También se puede observar que para cada switch se encuentra establecido el siguiente esquema de etiquetación: Al_UIO_Occ_Idc_P1_01, el mismo que hace referencia a la red de alumnos, la sede, el campus, el bloque IDIC, el piso 1 y el número de switch.

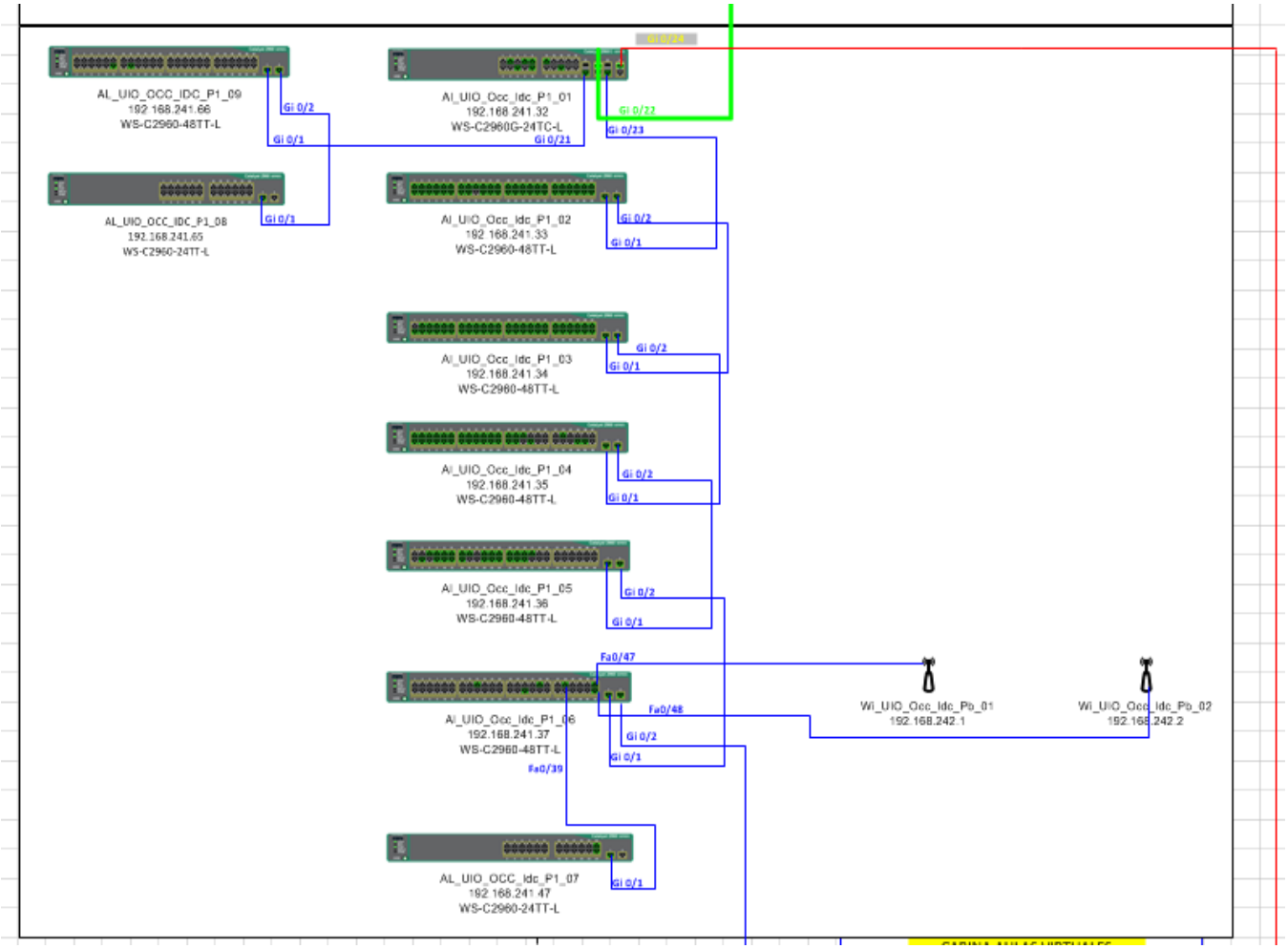


Figura 24. Diagrama de distribución física de una parte de la red de laboratorios del IDIC_UTE

- **Requerimientos de hardware**

Para el desarrollo de este proyecto de tesis se utilizó un computador portátil Toshiba Satellite P775-S5375 que posee las siguientes características:

- Procesador Intel Core i7 de 2.20GHz.
- Memoria RAM instalada de 4 GB.
- Capacidad de disco duro de 449 GB
- Sistema Operativo Windows 7 Home Premium Service Pack 1 de 64 bits.

- **Requerimientos de software**

- El administrador de máquinas virtuales Oracle VM VirtualBox versión 4.3.12 u otros.
- Emulador Mininet.
- Cliente SSH (Putty).
- Cliente SFTP (WinSCP).
- Lenguaje de programación Phyton versión 2.7.3, C++ o .Net
- Wireshark
- Para ejecutar sobre la máquina virtual con sistema operativo Windows o Linux se necesita un X11 u otro X-Manager, para Windows es necesario instalar Xming y para el caso de MacOSX instalar XQuartz.

4.2. Fase de Diseño

Para el diseño de la red SDN se elaboró una tabla de direccionamiento IP en base a una dirección privada clase A, la cual es asignada directamente por Mininet una vez que está ejecutada e implementada la topología de red diseñada en Python.

La Tabla 7 muestra el esquema de direccionamiento IP de la red de switches y host para la simulación.

Tabla 7. Esquema de direccionamiento IP de la red de Switches y host

DISPOSITIVO	NOMBRE	IP	MÁSCARA
h1	Host_UIO_Occ_Idc_P1_01_1	10.0.0.1	255.0.0.0
h2	Host_UIO_Occ_Idc_P1_01_2	10.0.0.2	255.0.0.0
h4	Host_UIO_Occ_Idc_P1_02_1	10.0.0.4	255.0.0.0
h5	Host_UIO_Occ_Idc_P1_02_2	10.0.0.5	255.0.0.0
h7	Host_UIO_Occ_Idc_P1_03_1	10.0.0.7	255.0.0.0
h8	Host_UIO_Occ_Idc_P1_03_2	10.0.0.8	255.0.0.0
h10	Host_UIO_Occ_Idc_P1_04_1	10.0.0.10	255.0.0.0
h11	Host_UIO_Occ_Idc_P1_04_2	10.0.0.11	255.0.0.0
h13	Host_UIO_Occ_Idc_P1_05_1	10.0.0.13	255.0.0.0
h14	Host_UIO_Occ_Idc_P1_05_2	10.0.0.14	255.0.0.0

h16	Host_UIO_Occ_Idc_P1_06_1	10.0.0.16	255.0.0.0
h17	Host_UIO_Occ_Idc_P1_06_2	10.0.0.17	255.0.0.0
h19	Host_UIO_Occ_Idc_P1_07_1	10.0.0.19	255.0.0.0
h20	Host_UIO_Occ_Idc_P1_07_2	10.0.0.20	255.0.0.0
h22	Host_UIO_Occ_Idc_P1_08_1	10.0.0.22	255.0.0.0
h23	Host_UIO_Occ_Idc_P1_08_2	10.0.0.23	255.0.0.0
h25	Host_UIO_Occ_Idc_P1_09_1	10.0.0.25	255.0.0.0
h26	Host_UIO_Occ_Idc_P1_09_2	10.0.0.26	255.0.0.0
s3	Al_UIO_Occ_Idc_P1_01	NO IP	NO MASK
s6	Al_UIO_Occ_Idc_P1_02	NO IP	NO MASK
s9	Al_UIO_Occ_Idc_P1_03	NO IP	NO MASK
s12	Al_UIO_Occ_Idc_P1_04	NO IP	NO MASK
s15	Al_UIO_Occ_Idc_P1_05	NO IP	NO MASK
s18	Al_UIO_Occ_Idc_P1_06	NO IP	NO MASK
s21	Al_UIO_Occ_Idc_P1_07	NO IP	NO MASK
s24	Al_UIO_Occ_Idc_P1_08	NO IP	NO MASK
s27	Al_UIO_Occ_Idc_P1_09	NO IP	NO MASK

- **Diámetro de la red**

  En esta topología de red el diámetro es igual a **9**, puesto que son 9 los switches que conforman la red del primer piso del bloque IDIC, por lo tanto, un paquete que es enviado desde un usuario conectado al primer switch con destino a otro usuario que está conectado al switch más lejano, debe sortear 9 saltos para llegar a su destino.

- **Redundancia**

  En redes SDN no es posible realizar enlaces redundantes debido a que en estas topologías el administrador posee la completa administración sobre los enlaces de red y las diferentes conexiones, permitiendo determinar con claridad el camino y el recorrido que enfrentarán los paquetes.

  Por esta razón, si el administrador desea configurar e implementar una ruta de enlaces tendrá que programarlo directamente en el controlador SDN donde en la actualidad se realiza todas las tareas

que los antiguos protocolos independientes hacían para garantizar el correcto encaminamiento de los paquetes.

- **Presupuestos y costos de la implementación del prototipo**

  Las tablas 8, 9 y 10 muestran un resumen de los costos reales directos, indirectos y totales que se ha gastado en el desarrollo de este proyecto de tesis.

Tabla 8. Costos Directos

Dispositivo	Cantidad	Precio Unitario	Precio Total
Toshiba Satellite P775-S5375	1	$ 1200.00	$ 1200.00
Oracle VM Virtual Box	Free	Free	Free
Emulador Mininet	Free	Free	Free
		Total	$ 1200.00

Tabla 9. Costos Indirectos

Detalle	Concepto	Precio Unitario	Precio Total
Curso CCNA	CCNA Modulo1	$ 269.00	$ 270,00
Capacitación	• *SDN* • *OPENFLOW* • *MININET* • *ORACLE VM VirtualBox*	$ 250,00	$ 250,00
		Total	$ 520.00

Tabla 10. Costo total aproximado del proyecto

Detalle	Valor
Costos Directos	$ 1200.00
Costos Indirectos	$ 520,00
Total	$ 1720.00

- **Planimetría de la red SDN**

En el anexo 2 se puede apreciar la topología de switches de la red SDN del primer piso del Bloque IDIC de la UTE que se utilizó para la simulación, la misma que está en correspondencia con la topología de la red física actual.

4.3. Fase de Implementación

- **Instalación máquina virtual VirtualBox**

La máquina virtual que se seleccionó es Oracle VM VirtualBox versión 4.3.12 y se puede descargar directamente desde la página de VirtualBox, en el anexo 3 se describe paso a paso el proceso de instalación y configuración.

- **Instalación Mininet**

Se puede implementar descargando una máquina virtual previamente configurada con un sistema operativo Linux y Ubunto de 64 bits, y sobre una máquina física con sistema operativo Ubunto versión 11.0 ó superior. La ventaja de trabajar previamente con una máquina virtual es que ya vienen preinstaladas ciertas aplicaciones para Wireshark, controlador SDN y comandos que vienen integrados para una mayor facilidad de uso.

- **Instalación y configuración de la topología personalizada.**

Como se indicó anteriormente, en el anexo 2 se puede observar la topología de la red SDN, los elementos que conforman la misma y el esquema de direccionamiento IP. En la figura 25 se puede apreciar la interfaz del script de red SDN en Python.

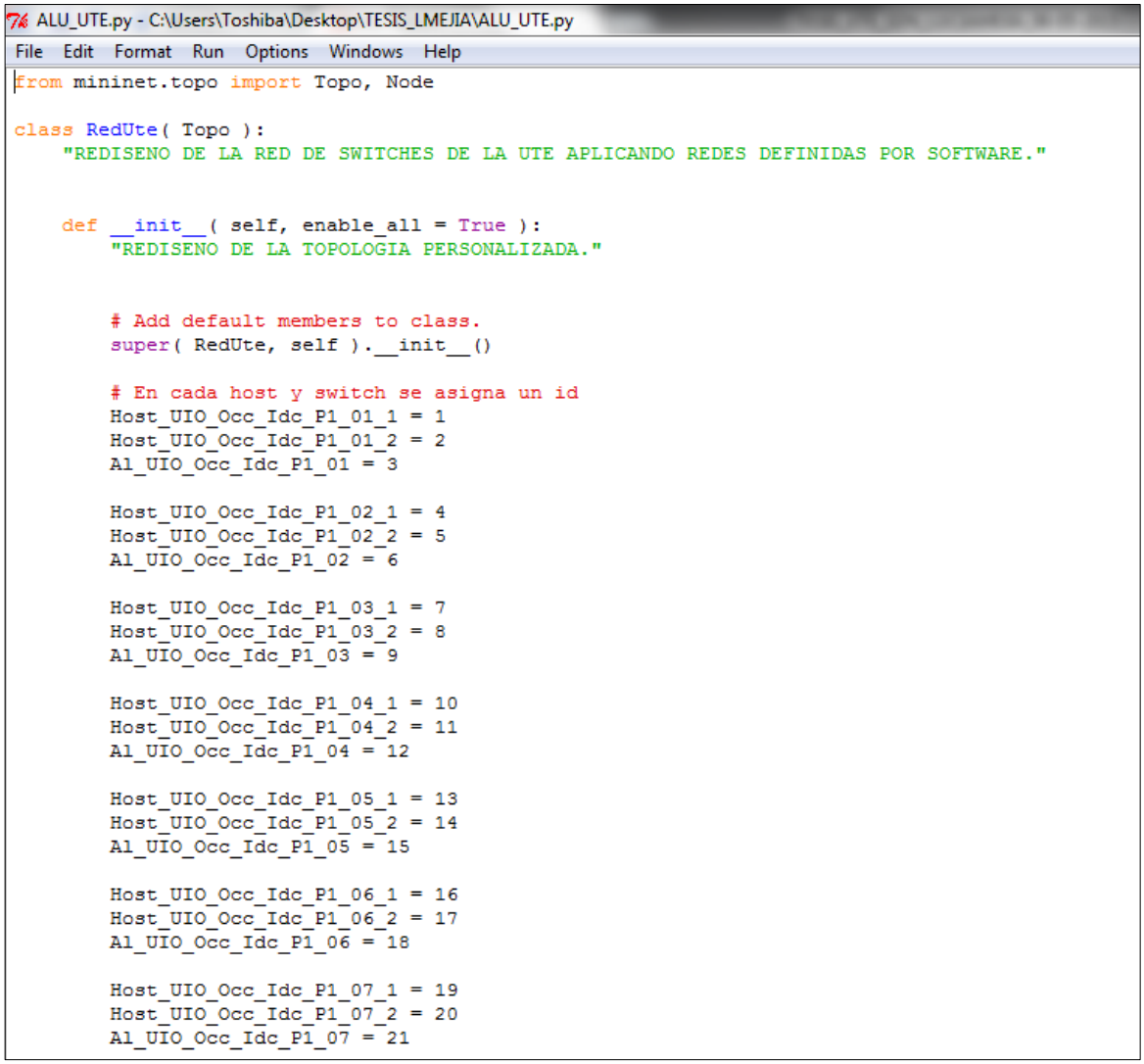


Figura 25. Interfaz en Python

El código fuente que se utilizó para crear la topología de red fue desarrollado en lenguaje de programación Python versión 2.7.3. A continuación se adjunta una parte del el script desarrollado para generar la red SDN, indicando lo que hace cada línea de comando.

- Se importan a las superclases Topo y Node para la topología y nodos a utilizarse en la simulación.

  from Mininet.topo import Topo, Node

- Definición de la clase RedUte la cual hereda métodos y atributos de la superclase Topo.

class RedUte(Topo):

- Indica comentario.

  "REDISENO DE LA RED DE SWITCHES DE LA UTE APLICANDO REDES DEFINIDAS POR SOFTWARE."

- __**init**__ Indica que este código se ejecuta después de crear un objeto de clase de instanciación.

- **self** Indica algún elemento invocado para luego instanciarlo y acceder a atributos y métodos.

- **enable_all = True** Indica que se habilitan todos los elementos de la red.

  def __init__(self, enable_all = True):

- Indica comentario.

  "REDISENO DE LA TOPOLOGIA PERSONALIZADA."

- Invoca al método de inicialización y se agregan todos los miembros por defecto en la clase.

  super(RedUte, self).__init__()

- Asignación de identificadores para hosts y switches.

  Host_UIO_Occ_Idc_P1_01_1 = 1
  Host_UIO_Occ_Idc_P1_01_2 = 2
  Al_UIO_Occ_Idc_P1_01 = 3

- Asignación de nodos diferenciándolos entre host y switches mediante el comando.
  is_switch siendo, (True = switch y False = host).

  self.add_node(Al_UIO_Occ_Idc_P1_01, Node(is_switch=True))
  self.add_node(Host_UIO_Occ_Idc_P1_01_1, Node(is_switch=False))

- Asignación de enlaces de conexión de red entre switches y hosts.

  self.add_edge(Host_UIO_Occ_Idc_P1_01_1, Al_UIO_Occ_Idc_P1_01)
  self.add_edge(Host_UIO_Occ_Idc_P1_01_2, Al_UIO_Occ_Idc_P1_01)
  self.add_edge(Al_UIO_Occ_Idc_P1_01, Al_UIO_Occ_Idc_P1_02)

- Se habilita y enciende a todos los dispositivos de red.

  self.enable_all()

- Es un diccionario que almacena un nombre personalizado como una llave ('sdn') y el nombre de la clase como un valor (lambda: RedUte()) para generar una

topología, lambda es un valor anónimo para llamar a la llave sdn para invocar a la clase RedUte().
topos = { 'sdn': (lambda: RedUte()) }

El programa completo se muestra en el anexo 4.

4.4.4. Ejecución en la máquina virtual de Mininet.

A continuación se procede con la creación y configuración en la máquina virtual de Mininet siguiendo los pasos de instalación. Una vez configurada, se ejecuta la misma ingresando el login y password definidos por defecto (openflow), En la figura 26 se muestra la pantalla de ingreso a Mininet mediante su autenticación.

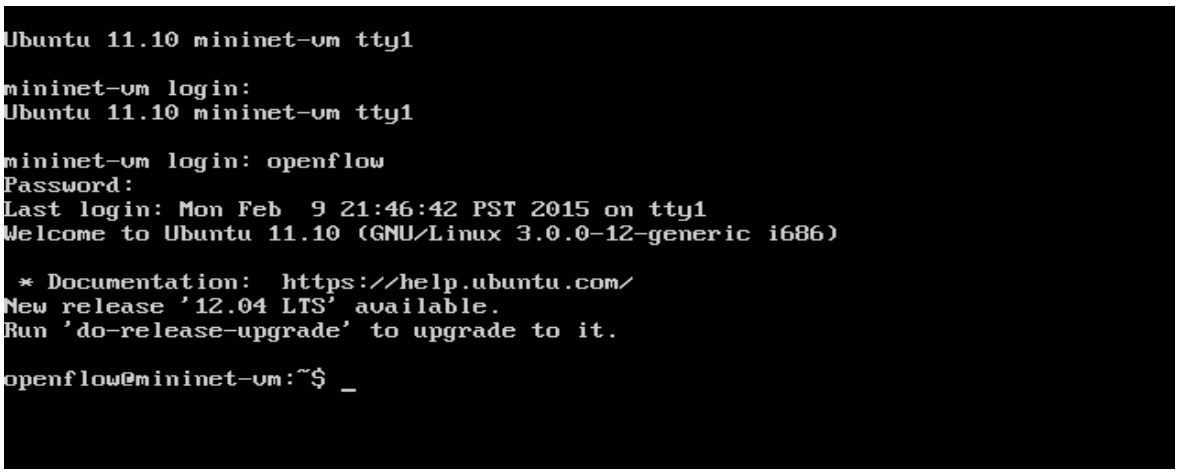


Figura 26. Acceso a Mininet

En la figura 27, se puede observar la pantalla que muestra el resultado de asignar una IP por defecto en la máquina virtual, mediante el comando:

- sudo dhclient eth1

Figura 27. Asignación IP a máquina virtual

Cabe indicar que la dirección IP por defecto corresponde a una de clase C privada 192.168.56.101 /24, la misma que permite acceder de forma local o remota, para la realización de pruebas de conectividad de la red virtual.

En la figura 28 se muestra la pantalla de la IP asignada mediante la ejecución del comando: ifconfig eth1, en donde se puede apreciar toda la información de la interface Ethernet 1 de la máquina virtual.

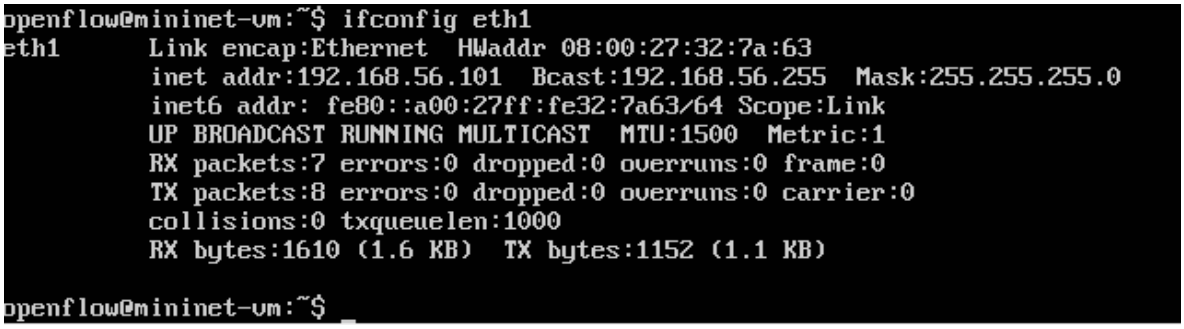


Figura 28. Comprobación de IP

Para iniciar la ejecución de la topología desarrollada en Python se debe cargar el script mediante WinSCP el cual realiza la transferencia segura de archivos entre dos sistemas, el local y el remoto mediante servicios SSH. En la figura 29 se muestra la pantalla de ingreso a la máquina virtual, mediante el usuario y la contraseña.

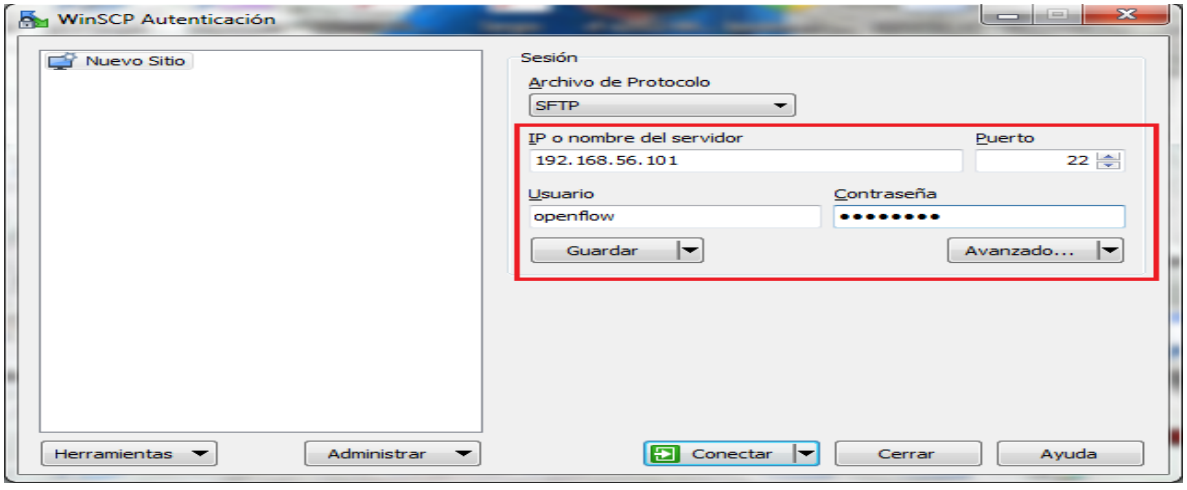


Figura 29. Acceso a Servidor WinSCP

Una vez conectado se realiza la transferencia de archivos, solo basta con encontrar la ruta a la que se copiará el script programado. En este caso la ruta es /home/openflow/mininet/custom y el nombre del script generado es ALU_UTE.py. En la figura 30, se puede observar la pantalla del archivo con el código fuente de la red que se transfirió correctamente hasta el servidor de la máquina virtual de Mininet.

Una vez creada y ejecutada la topología, se habilita la interfaz de comandos para sus múltiples pruebas de conectividad. Para cargar la topología se ejecuta el siguiente comando:

- sudo mn –custom ~/mininet/custom/ALU_UTE.py --topo sdn

La figura 31 muestra la pantalla del programa de la topología personalizada de la red SDN simulada, obtenida en el emulador Mininet, en donde se puede apreciar los diferentes host (h_i) y switch (s_j) creados.

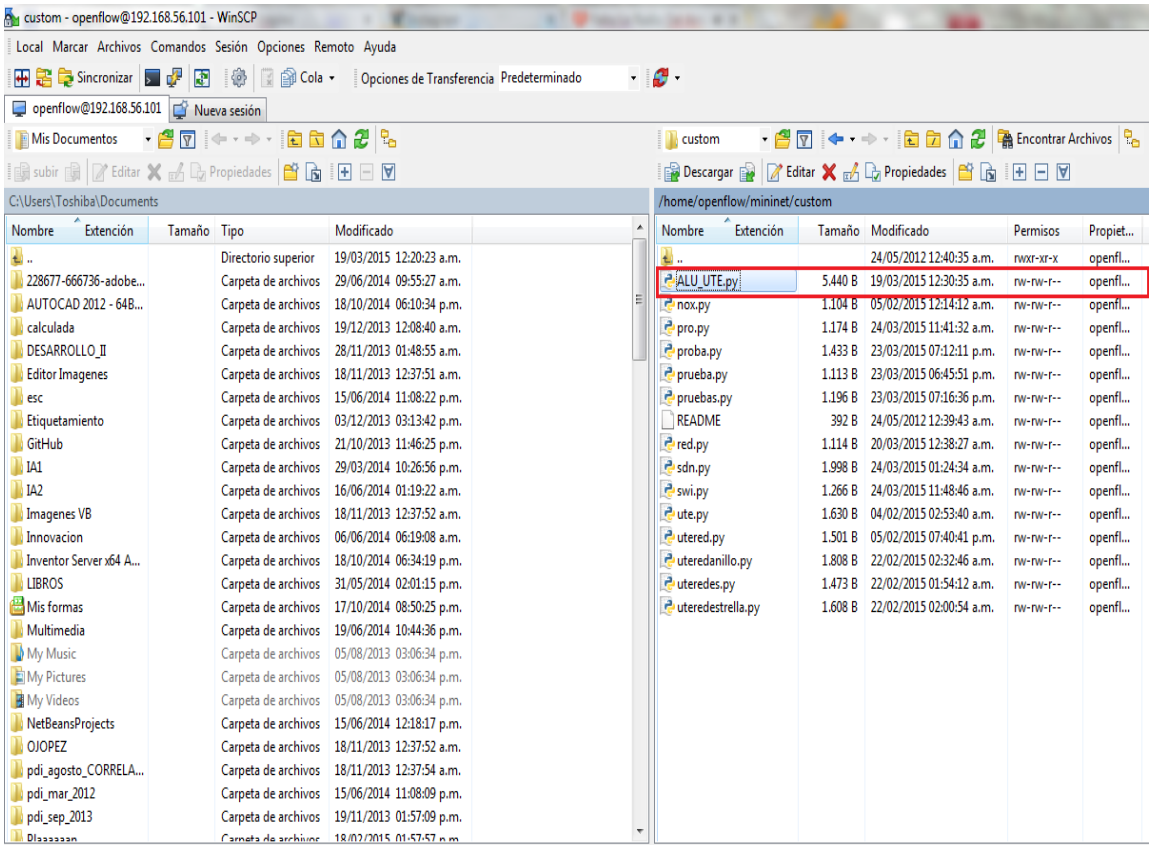


Figura 30. Transferencia de Archivo

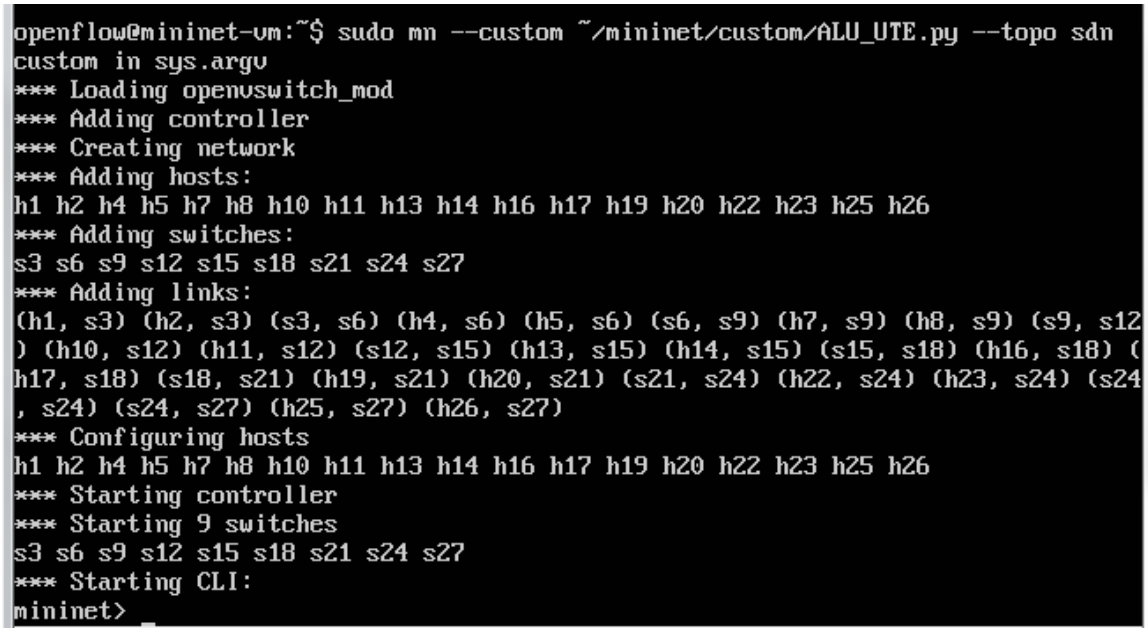


Figura 31. Topología de la red personalizada en Mininet

Una vez cargada la topología, desde la línea de comandos CLI de Mininet, se pueden realizar las pruebas de conectividad (ping) entre los diferentes host y probar la configuración de la red.

- **Comandos ejecutados en CLI de Mininet.**

Los siguientes comandos se ejecutarán en un entorno cliente de CLI Mininet, que es una interfaz de línea de comandos que permite y facilita la comunicación y la conexión entre los nodos de una red, y gracias a esta interfaz se pueden realizar las múltiples pruebas con éxito.

En la figura 32 se muestra la pantalla con la IP y máscara del host deseado, solo basta con ejecutar un comando que reflejará toda la información perteneciente a cada uno de ellos, se ejecuta de la siguiente manera:

- h1 ifconfig –a

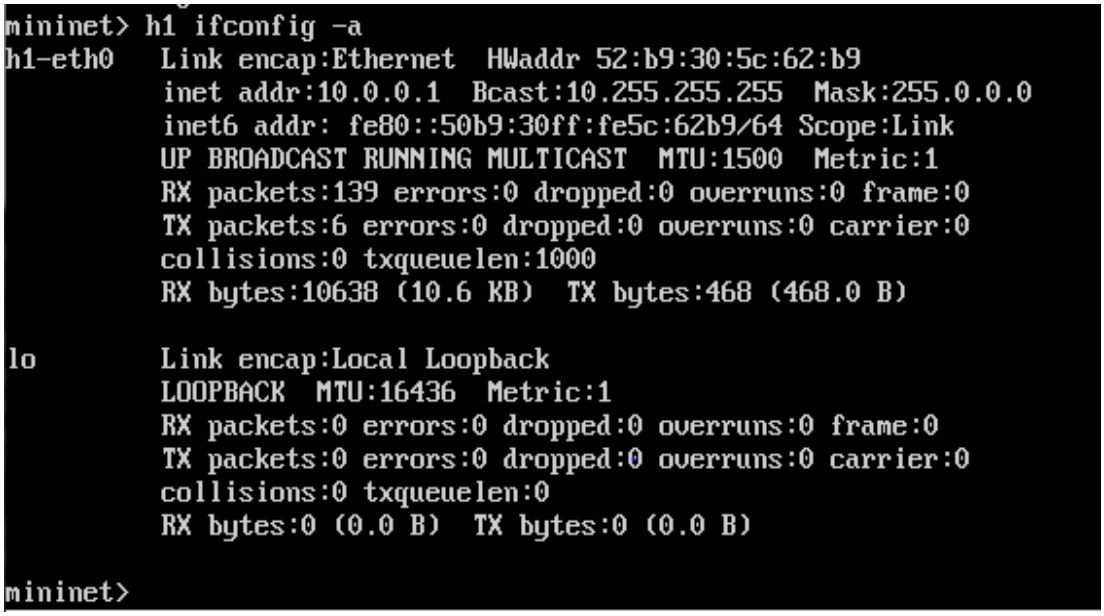


Figura 32. Configuración de la interfaz de red del host

En la figura 33 se muestra la pantalla de configuración de la interfaz de red del switch y se ejecuta mediante el comando:

- s9 ifconfig –a

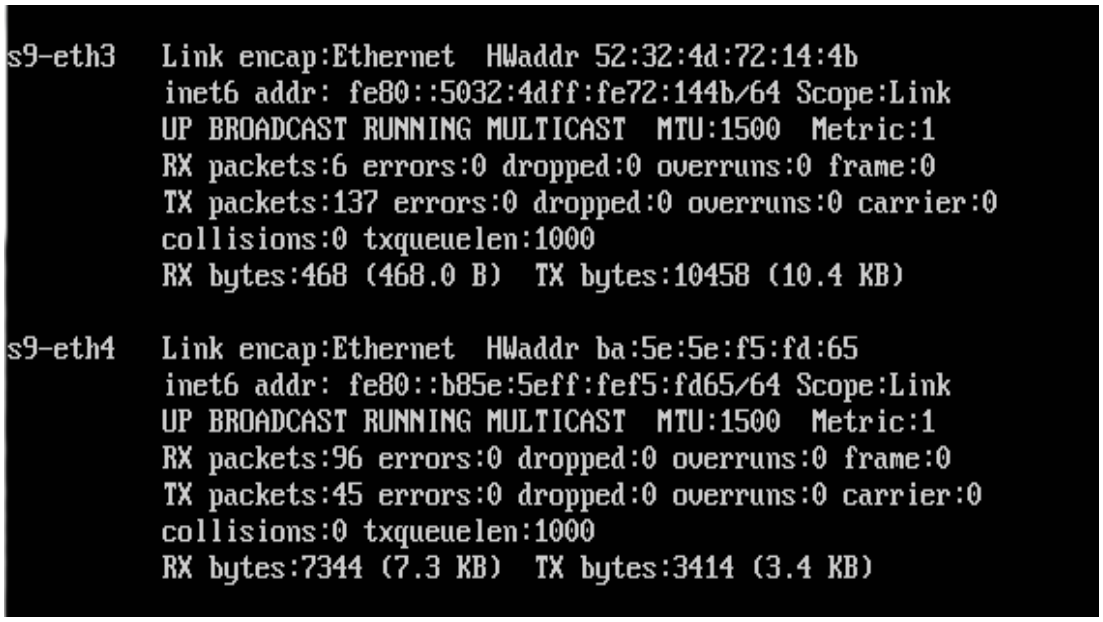


Figura 33. Configuración de la interfaz de red del switch

En la figura 34 se muestra la pantalla de conexión de los hosts finales a cada switch y se comprueba mediante la ejecución del siguiente comando:

- net 0

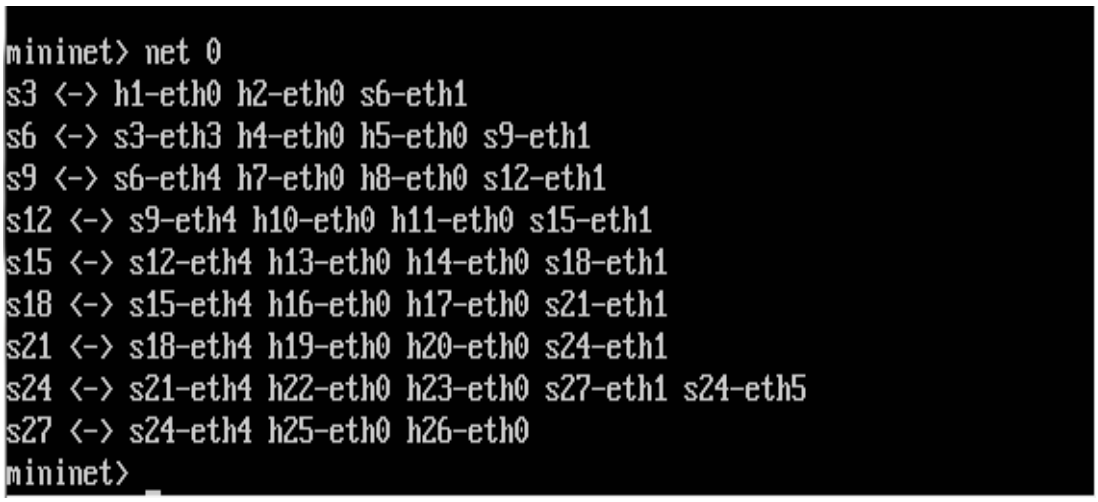


Figura 34. Conexión de hosts a switches

En la figura 35 se muestra la pantalla de conectividad entre los nodos de la red, se ejecuta el comando ping seguido del operador –c**N** (donde **N** indica el número específico de paquetes a transmitir), de la siguiente manera:

- h1 ping –c10 h2

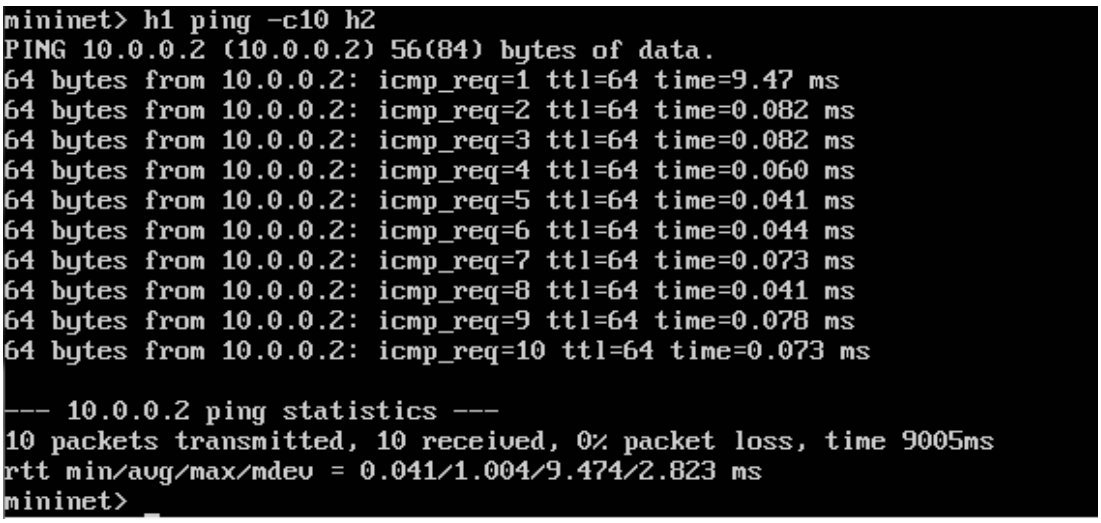


Figura 35. Comprobación individual de la conectividad entre nodos

En la figura 36 se muestra la pantalla de conectividad entre todos los host de la red, se ejecuta el comando pingall, de la siguiente manera:

- pingall

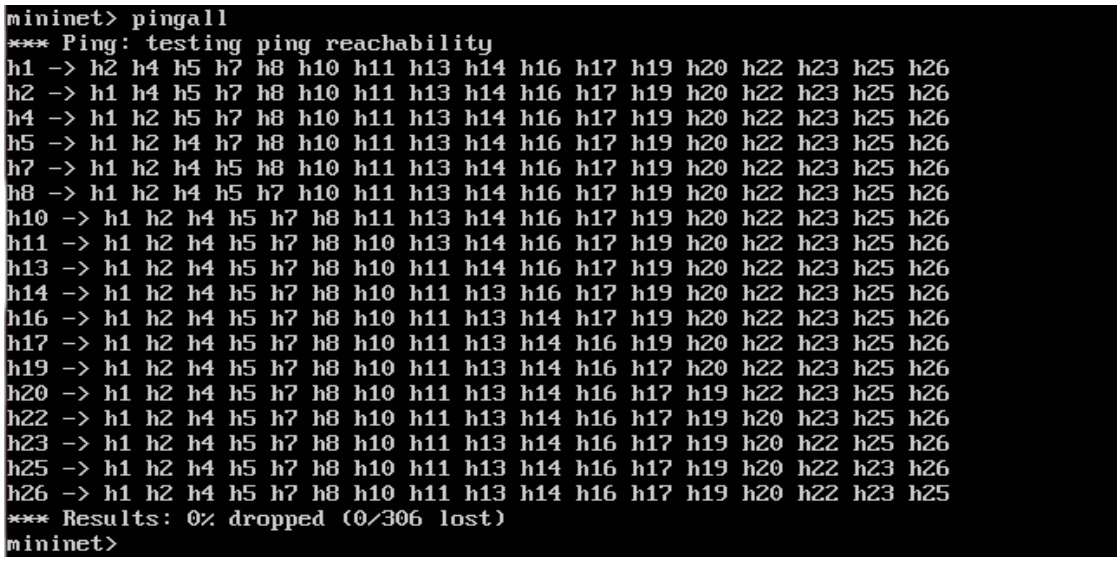


Figura 36. Comprobación de la conectividad entre todos los host

En la figura 37 se muestra el tipo de información de todos los nodos que forman parte de la topología de red, se ejecuta el siguiente comando:

- dump

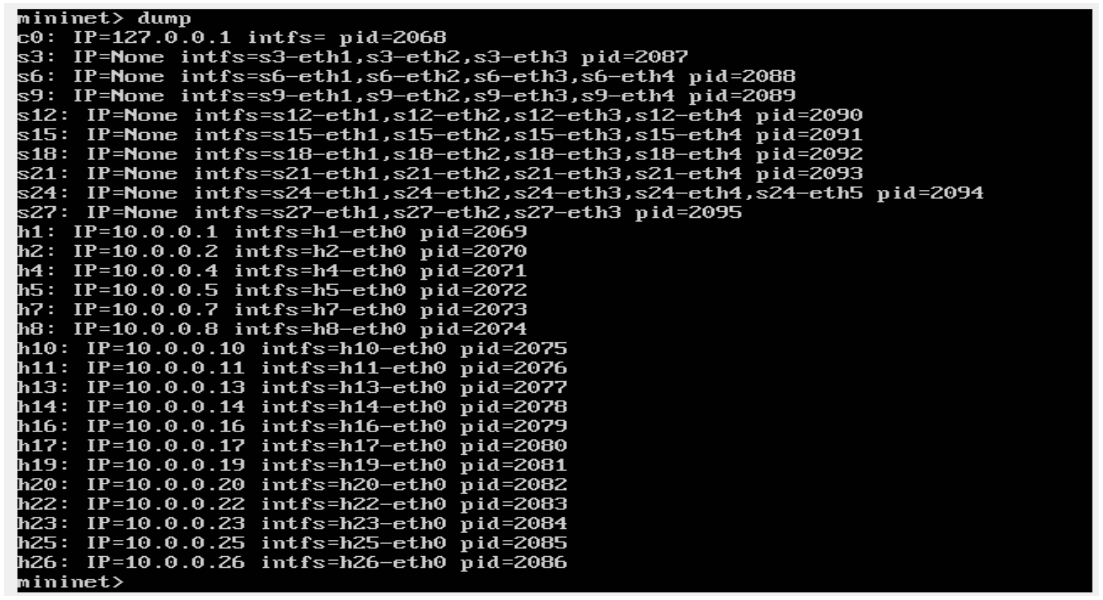


Figura 37. Información de Nodos

En la figura 38 se muestra la pantalla con la tabla de direcciones físicas del switch y en la figura 39 se muestra la pantalla de arp, para su visualización se ejecutan los siguientes comandos:

- s12 route

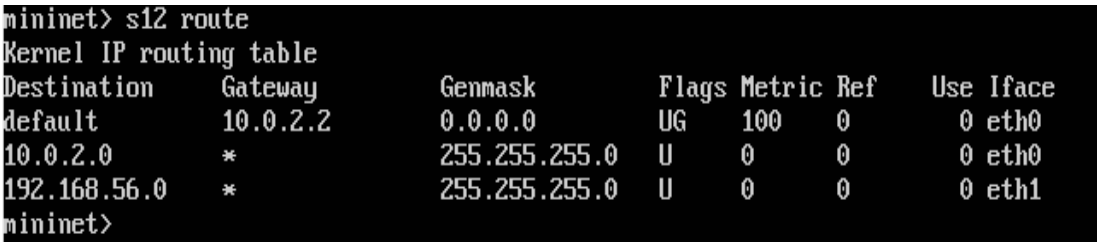


Figura 38. Tabla de routing del switch

- s12 arp

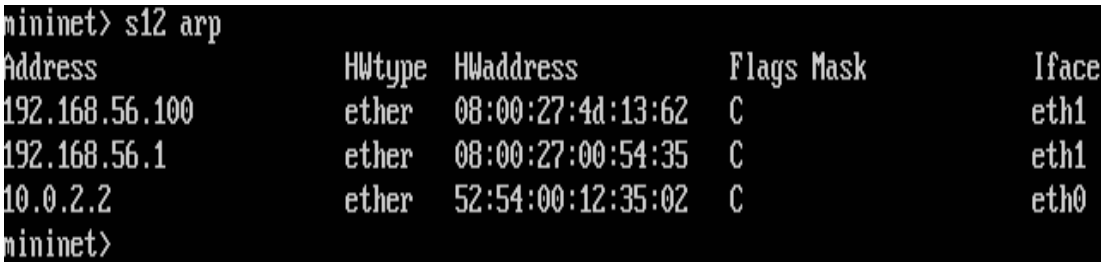


Figura 39. Tabla de arp del switch

4.4. Fase de Pruebas

- **Analizador de tráfico**

Para realizar esta acción es necesario conectarse mediante acceso remoto Putty, como punto inicial es necesario instalar un X-Manager, en el caso de este proyecto de titulación se ejecutó sobre en una máquina virtual en Windows y para esto se descargará e instalará un X-Manager Xming.

En la figura 40 se muestra la pantalla con las configuraciones iniciales que se deben realizar para poder utilizar xterm, el mismo que permite ejecutar pruebas en interfaces individuales para esto en VirtualBox es necesario

realizar ciertos cambios. Antes de ejecutar la máquina virtual cambiar lo siguiente:

- Configuración
- Sistema
- Procesador
- Habilitar características extendidas y habilitar PAE/NX.

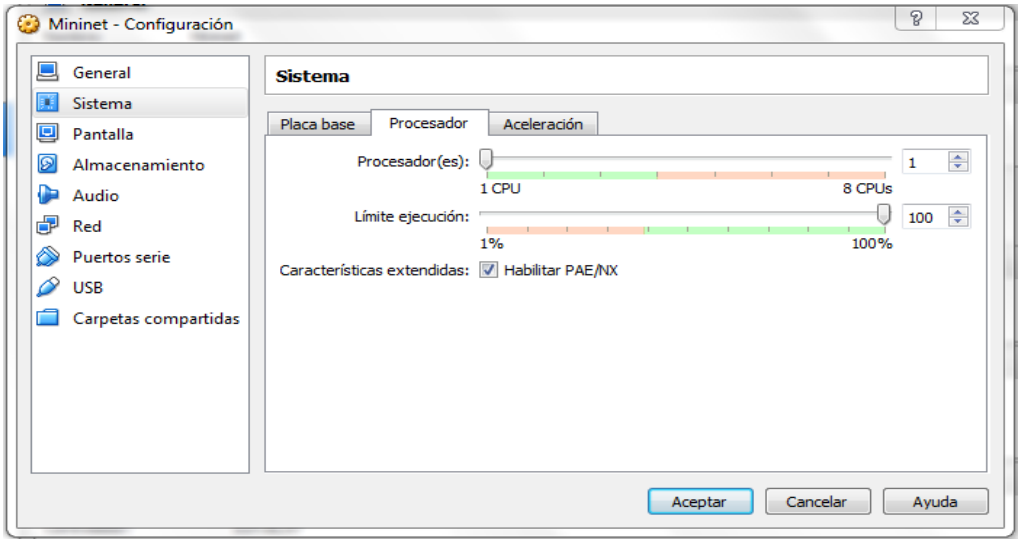


Figura 40. Configuración para Xming

Para proceder al acceso remoto se realizará desde un cliente SSH, en este proyecto se utilizará Putty, el cual previamente se debe tener instalado en la máquina, es necesario habilitar la opción X11 para que sea posible la conexión con el servidor Xming, la forma de acceder es con la misma dirección IP que se asignó a la máquina virtual de Mininet y posteriormente se ingresará un usuario y contraseña (openflow) que son los mismos que se ingresó en Mininet.

En la figura 41 se muestra la pantalla de la conexión mediante SSH para tener acceso remoto a la máquina virtual.

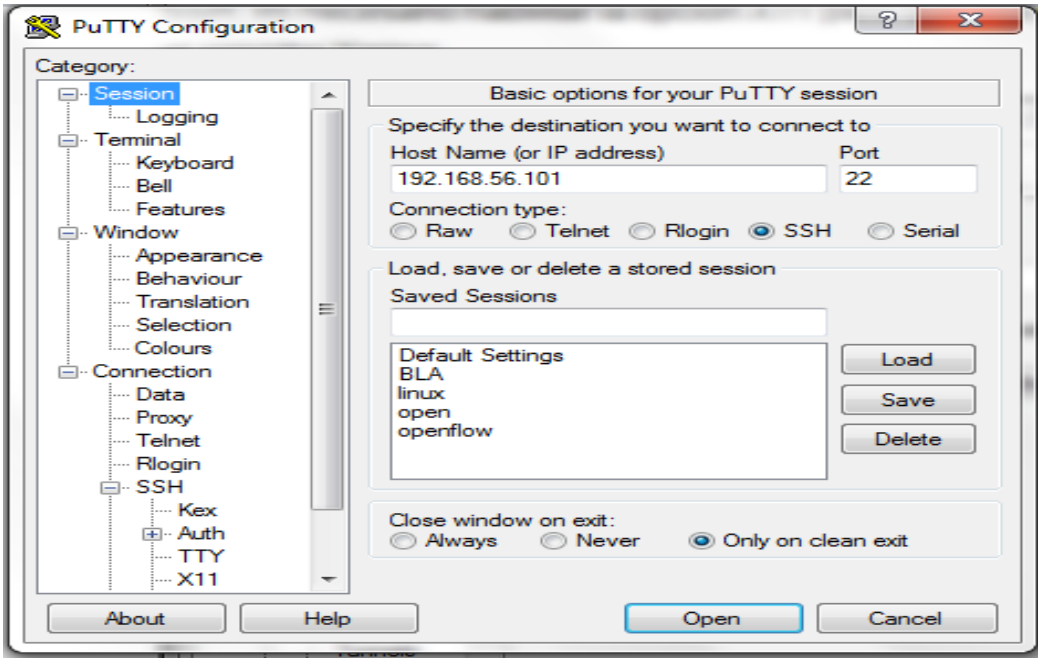


Figura 41. Conexión mediante SSH

En la figura 42 se muestra la pantalla con la opción que se debe habilitar X11 para que se puedan ejecutar las aplicaciones xterm y wireshark.

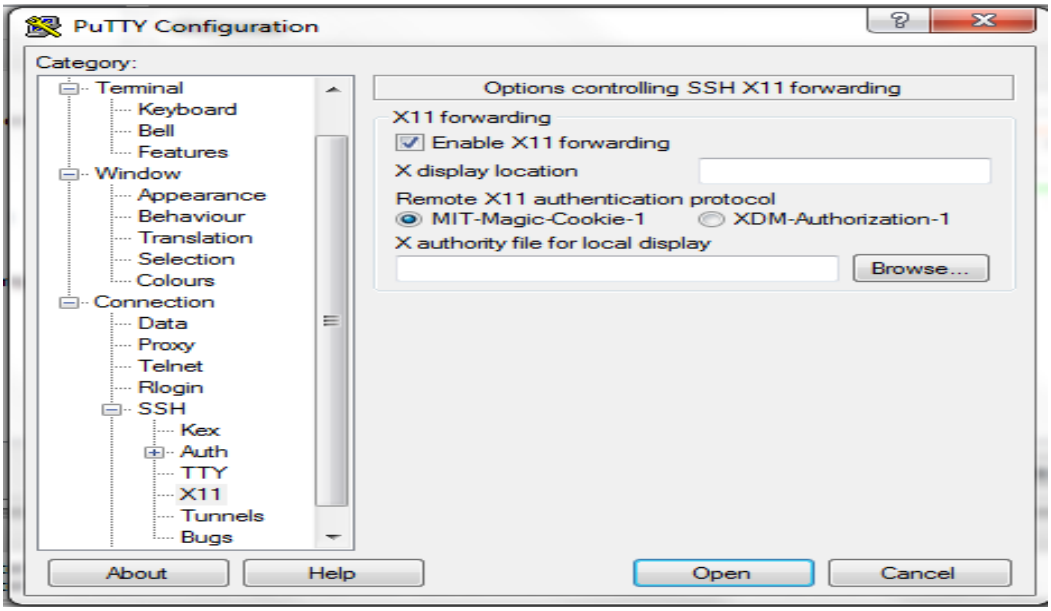


Figura 42. Habilitar X11

Después se ejecuta el X-Manager que se instaló previamente para que se puedan tener acceso a las aplicaciones exteriores de Mininet. En la figura 43 se muestra la pantalla del proceso para ejecutar Xming y consiste en ir a Inicio, todos los programas y buscar una carpeta llamada Xming, abrir la carpeta y ejecutar el servidor Xming, de inmediato se activará y se podrán ejecutar aplicaciones como wireshark y xterm para observar el tráfico de red.

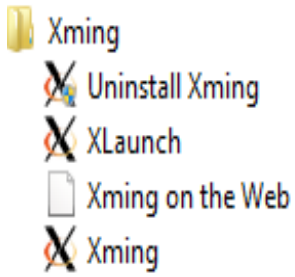


Figura 43. X-Manager Xming

En la figura 44 se muestra la pantalla de acceso mediante SSH a la máquina virtual de Mininet.

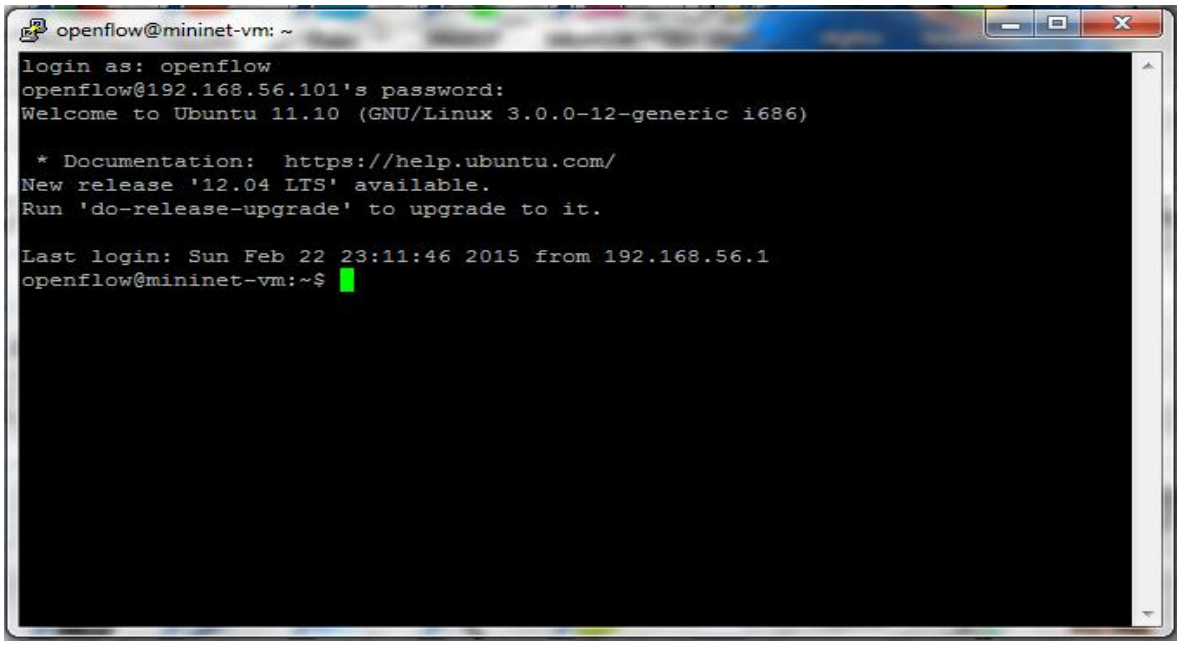


Figura 44. Máquina virtual mediante SSH

En la figura 45 se muestra la pantalla de ejecución del analizador de tráfico donde se podrá observar el protocolo OpenFlow de la siguiente manera:

- sudo wireshark &

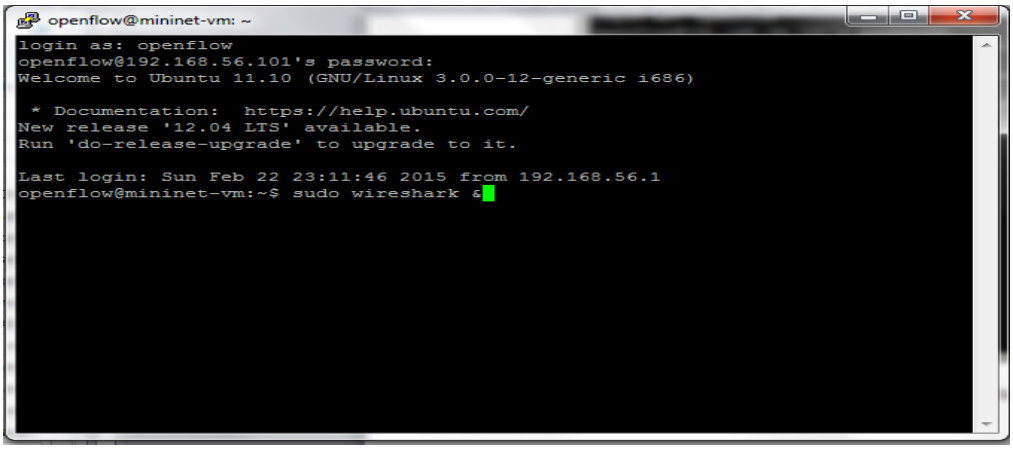


Figura 45. Ejecución de Wireshark

En la figura 46 se muestra la pantalla de interfaz del análisis de tráfico, a los dos mensajes que aparecen se selecciona OK y seleccionar la interfaz de LoopBack y se empieza a analizar el tráfico de la red. En la figura 47 se muestra la pantalla donde ya se refleja el protocolo OpenFlow (OFP), se puede realizar envío y recepción de paquetes en la red para tener un mayor volumen de tráfico y observar con mayor frecuencia el Protocolo OpenFlow.

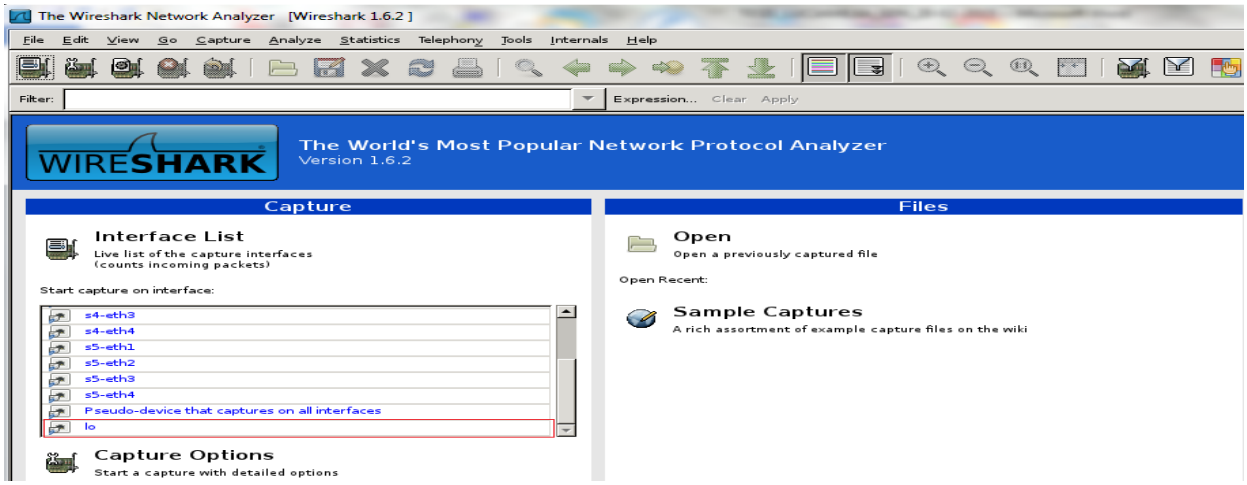
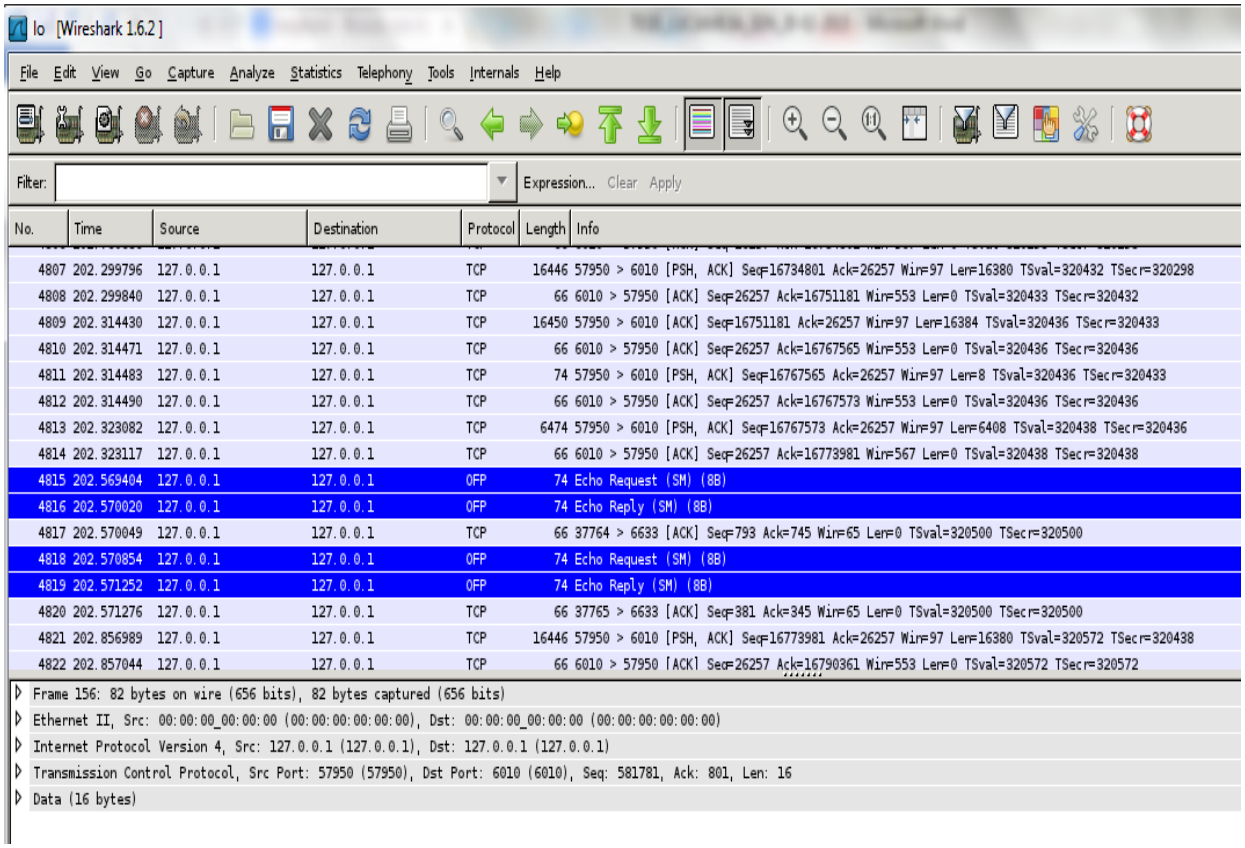


Figura 46. Selección de interfaz LoopBack



373	6.600169	127.0.0.1	127.0.0.1	OFP	74 Echo Request (SM) (8B)
374	6.600272	127.0.0.1	127.0.0.1	OFP	74 Echo Request (SM) (8B)
375	6.600368	127.0.0.1	127.0.0.1	OFP	74 Echo Request (SM) (8B)
376	6.600463	127.0.0.1	127.0.0.1	OFP	74 Echo Request (SM) (8B)
377	6.600558	127.0.0.1	127.0.0.1	OFP	74 Echo Request (SM) (8B)
378	6.601601	127.0.0.1	127.0.0.1	OFP	74 Echo Reply (SM) (8B)
379	6.601650	127.0.0.1	127.0.0.1	TCP	66 53185 > 6633 [ACK] Seq=17 Ack=17 Win=65 Len=0 TSval=3499500 TSecr=3499500
380	6.601822	127.0.0.1	127.0.0.1	OFP	74 Echo Reply (SM) (8B)
381	6.601854	127.0.0.1	127.0.0.1	TCP	66 53186 > 6633 [ACK] Seq=17 Ack=17 Win=65 Len=0 TSval=3499500 TSecr=3499500
382	6.602111	127.0.0.1	127.0.0.1	OFP	74 Echo Request (SM) (8B)
383	6.604508	127.0.0.1	127.0.0.1	OFP	74 Echo Reply (SM) (8B)
384	6.604523	127.0.0.1	127.0.0.1	TCP	66 53187 > 6633 [ACK] Seq=17 Ack=17 Win=65 Len=0 TSval=3499501 TSecr=3499501
385	6.604835	127.0.0.1	127.0.0.1	OFP	74 Echo Reply (SM) (8B)
386	6.666691	127.0.0.1	127.0.0.1	TCP	16450 41061 > 6010 [ACK] Seq=826837 Ack=2305 Win=97 Len=16384 TSval=3499516 TSecr=3499
387	6.666723	127.0.0.1	127.0.0.1	TCP	66 6010 > 41061 [ACK] Seq=2305 Ack=843221 Win=553 Len=0 TSval=3499516 TSecr=3499516
388	6.666822	127.0.0.1	127.0.0.1	TCP	70 41061 > 6010 [PSH, ACK] Seq=843221 Ack=2305 Win=97 Len=4 TSval=3499516 TSecr=349
389	6.666832	127.0.0.1	127.0.0.1	TCP	66 6010 > 41061 [ACK] Seq=2305 Ack=843225 Win=553 Len=0 TSval=3499516 TSecr=3499516
390	6.695554	127.0.0.1	127.0.0.1	TCP	214 41093 > 6010 [PSH, ACK] Seq=6841 Ack=481 Win=97 Len=148 TSval=3499524 TSecr=3499
391	6.695893	127.0.0.1	127.0.0.1	TCP	370 41093 > 6010 [PSH, ACK] Seq=6989 Ack=481 Win=97 Len=304 TSval=3499524 TSecr=3499
392	6.695927	127.0.0.1	127.0.0.1	TCP	66 6010 > 41093 [ACK] Seq=481 Ack=7293 Win=97 Len=0 TSval=3499524 TSecr=3499524
393	6.697361	127.0.0.1	127.0.0.1	TCP	98 6010 > 41093 [PSH, ACK] Seq=481 Ack=7293 Win=97 Len=32 TSval=3499524 TSecr=34995
394	6.697378	127.0.0.1	127.0.0.1	TCP	66 41093 > 6010 [ACK] Seq=7293 Ack=513 Win=97 Len=0 TSval=3499524 TSecr=3499524
395	6.773191	127.0.0.1	127.0.0.1	TCP	6230 41061 > 6010 [PSH, ACK] Seq=843225 Ack=2305 Win=97 Len=6164 TSval=3499543 TSecr=
396	6.773221	127.0.0.1	127.0.0.1	TCP	66 6010 > 41061 [ACK] Seq=2305 Ack=849389 Win=568 Len=0 TSval=3499543 TSecr=3499543
397	7.000976	127.0.0.1	127.0.0.1	TCP	16430 41061 > 6010 [PSH, ACK] Seq=849389 Ack=2305 Win=97 Len=16364 TSval=3499600 TSecr
398	7.001009	127.0.0.1	127.0.0.1	TCP	66 6010 > 41061 [ACK] Seq=2305 Ack=865753 Win=553 Len=0 TSval=3499600 TSecr=3499600
399	7.114590	127.0.0.1	127.0.0.1	TCP	16450 41061 > 6010 [ACK] Seq=865753 Ack=2305 Win=97 Len=16384 TSval=3499628 TSecr=3499
400	7.114626	127.0.0.1	127.0.0.1	TCP	66 6010 > 41061 [ACK] Seq=2305 Ack=882137 Win=553 Len=0 TSval=3499628 TSecr=3499628

Figura 47. Visualización de protocolo OpenFlow

- **Pruebas Xterm**

Mininet incluye una aplicación llamada Xterm que permite realizar pruebas desde cualquier host de manera independiente, para ejecutarlo solo basta con ejecutar en Putty la topología personalizada de manera similar que en la máquina virtual de Mininet. En la figura 48 se muestra la pantalla donde se ejecuta la topología desde Putty donde se puede apreciar los diferentes host (h_i) y switch (s_j) creados.

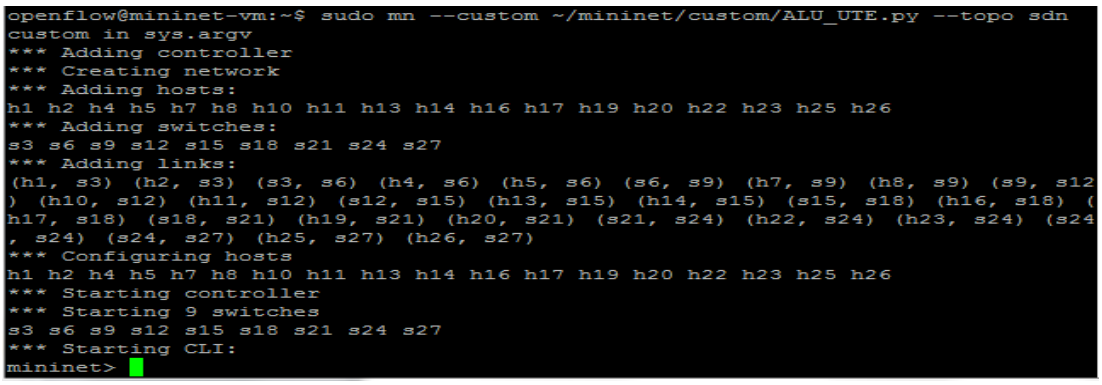


Figura 48. Topología de la red personalizada en Putty

En la figura 49 se muestra la pantalla donde se muestra la ejecución independiente del host 1.

- xterm h1

Figura 49. Ejecución Node: h1

En la figura 50 se muestra la pantalla de conectividad entre un host específico de la red y una dirección IP de un segundo host, mediante la ejecución del comando ping seguido del operador –c**N** (donde **N** indica el número específico de paquetes a transmitir), de la siguiente manera:

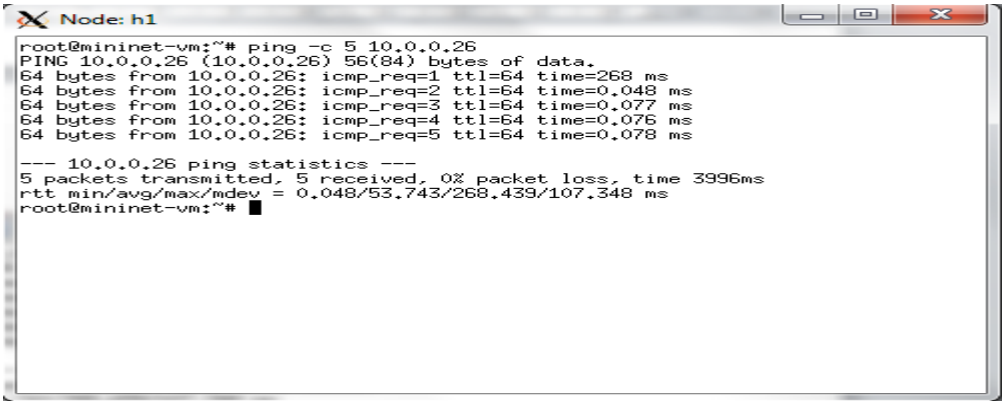


Figura 50. Envío y recepción de paquetes entre h1 y h26

En la figura 51 se muestra la pantalla que refleja el flujo de tráfico en Wireshark después de realizar el envío y recepción de paquetes entre h1 y la ip 10.0.0.26 que corresponde al host 26.

302	4.692927	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=496) (116B) => Echo (ping) request id=(
303	4.693397	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
304	4.693447	127.0.0.1	127.0.0.1	TCP	66 53194 > 6633 [ACK] Seq=357 Ack=193 Win=65 Len=0 TSval=3499023
305	4.700567	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=435) (116B) => Echo (ping) request id=(
306	4.700806	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
307	4.700819	127.0.0.1	127.0.0.1	TCP	66 53195 > 6633 [ACK] Seq=357 Ack=193 Win=65 Len=0 TSval=3499025
308	4.719025	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=433) (116B) => Echo (ping) request id=(
309	4.719286	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
310	4.719300	127.0.0.1	127.0.0.1	TCP	66 53196 > 6633 [ACK] Seq=357 Ack=193 Win=65 Len=0 TSval=3499030
311	4.735053	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=428) (116B) => Echo (ping) request id=(
312	4.735306	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
313	4.735320	127.0.0.1	127.0.0.1	TCP	66 53197 > 6633 [ACK] Seq=357 Ack=193 Win=65 Len=0 TSval=3499034
314	4.738045	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=424) (116B) => Echo (ping) request id=(
315	4.738245	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
316	4.738257	127.0.0.1	127.0.0.1	TCP	66 53198 > 6633 [ACK] Seq=357 Ack=193 Win=65 Len=0 TSval=3499034
317	4.771100	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=416) (116B) => Echo (ping) request id=(
318	4.771374	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
319	4.771388	127.0.0.1	127.0.0.1	TCP	66 53199 > 6633 [ACK] Seq=357 Ack=193 Win=65 Len=0 TSval=3499043
320	4.771919	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=412) (116B) => Echo (ping) request id=(
321	4.772113	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
322	4.772125	127.0.0.1	127.0.0.1	TCP	66 53200 > 6633 [ACK] Seq=349 Ack=185 Win=65 Len=0 TSval=3499043
323	4.800688	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=409) (116B) => Echo (ping) request id=(
324	4.800962	127.0.0.1	127.0.0.1	OFP	146 Flow Mod (CSM) (80B)
325	4.800976	127.0.0.1	127.0.0.1	TCP	66 53201 > 6633 [ACK] Seq=349 Ack=185 Win=65 Len=0 TSval=3499050
326	4.801694	127.0.0.1	127.0.0.1	TCP	16450 41061 > 6010 [PSH, ACK] Seq=655149 Ack=2305 Win=97 Len=16384
327	4.801763	127.0.0.1	127.0.0.1	TCP	66 6010 > 41061 [ACK] Seq=2305 Ack=671533 Win=553 Len=0 TSval=345
328	4.840750	10.0.0.1	10.0.0.26	OFP+ICMP	182 Packet In (AM) (BufID=392) (116B) => Echo (ping) request id=(

Figura 51. Flujo de tráfico después de realizar ping entre h1 y h26

- **Pruebas de conectividad**

Para el desarrollo de pruebas de conectividad entre host de la red simulada se utilizó el comando ping entre un host inicial, central y lejano. En la figura 52 se muestra la pantalla con la primera prueba, se envían paquetes entre los host h1 a h2 que son los más cercanos.

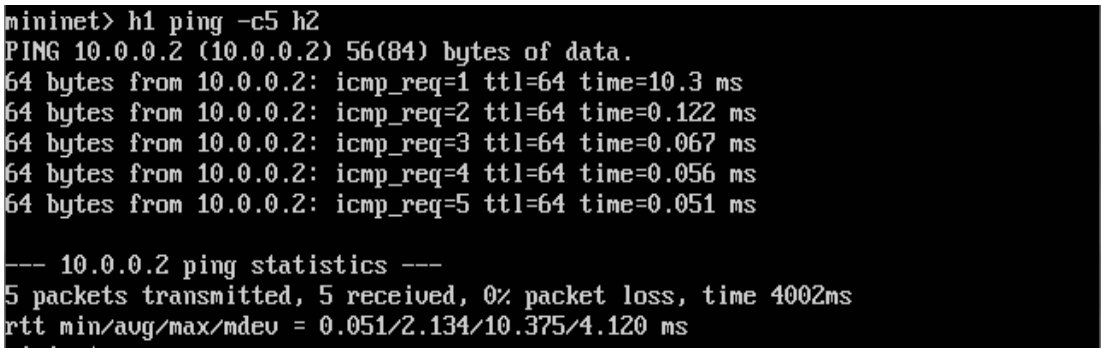


Figura 52. Envío de paquetes desde h1 hasta h2

En la figura 53 se muestra la pantalla con la segunda prueba, se envían paquetes entre los host h1 a h14.

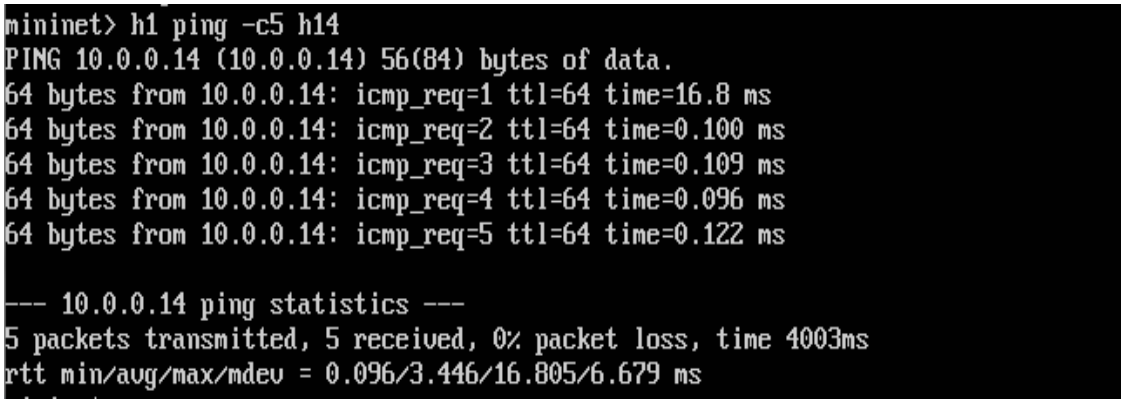


Figura 53. Envío de paquetes desde h1 hasta h14

En la figura 54 se muestra la pantalla con la tercera prueba, se envían paquetes entre los host h1 a h26.

```
mininet> h1 ping -c5 h26
PING 10.0.0.26 (10.0.0.26) 56(84) bytes of data.
64 bytes from 10.0.0.26: icmp_req=1 ttl=64 time=41.8 ms
64 bytes from 10.0.0.26: icmp_req=2 ttl=64 time=0.080 ms
64 bytes from 10.0.0.26: icmp_req=3 ttl=64 time=0.146 ms
64 bytes from 10.0.0.26: icmp_req=4 ttl=64 time=0.118 ms
64 bytes from 10.0.0.26: icmp_req=5 ttl=64 time=0.075 ms

--- 10.0.0.26 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.075/8.456/41.863/16.703 ms
```

Figura 54. Envío de paquetes desde h1 hasta h26

La tabla 11 presenta muestra los tiempos de retardo y número de paquetes que se envían a través de la red en las tres pruebas respectivamente.

Tabla 11. Tabla de envío de paquetes en la red

N° de Paquetes	Host	1	2	3	4	5
Tiempo (ms)	h1-h2	10,3	1,22	0,067	0,056	0,051
	h1-h14	16,8	0,1	0,109	0,096	0,122
	h1-h26	41,8	0,08	0,146	0,118	0,075

Como se puede apreciar en la figura 55 se comprueba la conectividad entre los host se puede observar que el tiempo de respuesta al primer paquete es alto, a medida que se estabiliza la red los próximos paquetes que se envían con menor retardo y mayor rapidez.

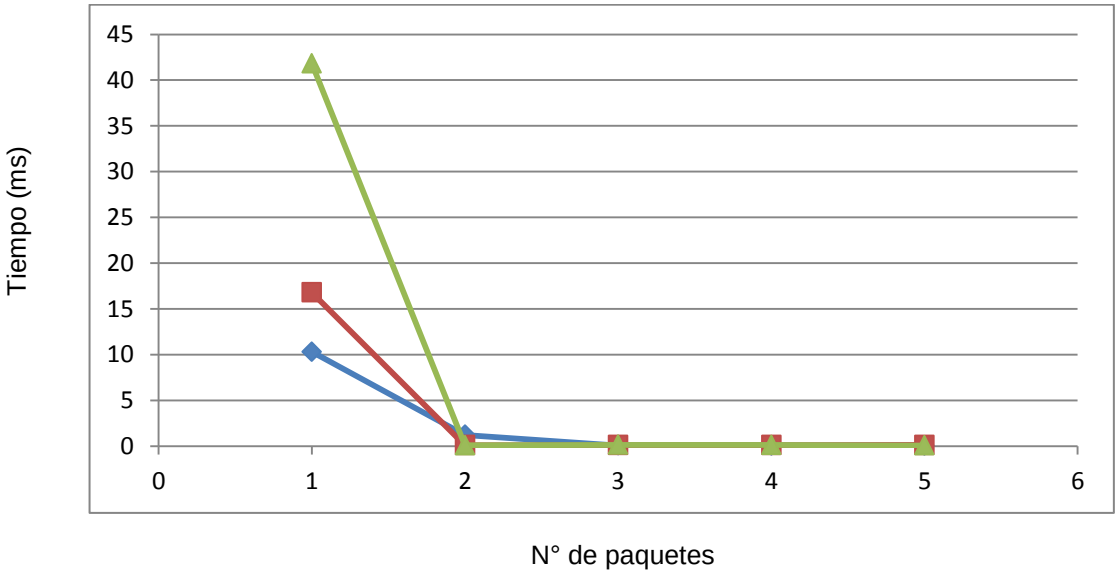


Figura 55. Tiempo de respuesta Vs. N° de paquetes en la red

CONCLUSIONES Y RECOMENDACIONES

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES.

- Para el diseño de redes SDN, es importante conocer sus características principales, en especial su arquitectura ya que esto nos permitirá seleccionar las herramientas adecuadas para implementación y configuración de las diferentes topologías a través de la programación de las mismas en el lenguaje Python. Cabe indicar que la interfaz Mininet constituye el núcleo central de una red SDN ya que en este entorno se crea y emula el comportamiento lógico de una de red física.

- Para el diseño de la topología lógica de Switches de red SDN de los laboratorios de computación del IDIC, se realizó un análisis de la topología física actual para definir el número de switches y hosts que funcionan y que conformaron esta nueva topología, posteriormente se elaboró una planimetría bifilar donde se definieron claramente la comunicación y conexión entre los diferentes dispositivos.

- Una vez diseñada la red SDN del primer piso de los laboratorios de computación del IDIC, para simular el funcionamiento de la misma, se elaboró un programa en Python que emula el comportamiento lógico de la red en base a la creación de dispositivos y enlaces en concordancia con la topología definida. Luego se ejecutó el script desde Mininet, se creó la red, se asignó las direcciones IP y máscaras en los diferentes host, y a través del uso de las CLI se realizó las pruebas de conectividad.

- Para el desarrollo de la topología de red SDN, es necesario conocer la estructura física de la red que se quiere modelar para luego, a partir de ésta realizar el desarrollo, programación e implementación del código fuente de la topología de la red. En el emulador Mininet no es posible la personalización de la red con direccionamiento ip, ya que esta máquina virtual posee un rango de ip's establecidas para sus topologías, por esta razón no se puede definir direcciones directamente, esto se podrá implementar en los equipos que se configuren como controladores SDN, de esta manera se podrá establecer la implementación y aplicación de subredes y máscara según el administrador lo considere necesario. Adicional a esto tampoco se pueden definir direcciones mac ya que el servidor de Mininet inicia automáticamente con direcciones mac al azar, cada vez que se ejecuta la topología o se inicia Mininet las direcciones cambian y por esta razón no se puede definir direcciones específicas para cada host.

- En este proyecto de simulación de redes SDN como parte de metodología se seleccionó el ciclo de vida en cascada ya al igual que cualquier proyecto se siguen varias fases y etapas hasta comprobar su funcionalidad y validación. Esto ha permitido generar una solución de bajo costo que ayudará a mejorar la administración de tráfico en la red de datos de una empresa o institución.

- Para el desarrollo de la topología de red SDN personalizada se seleccionó el lenguaje de programación Python versión 2.7 por ser el más compatible con redes SDN en la actualidad, posee librerías compatibles con Mininet y en estas se programan directamente la clase principal y subclases necesarios para la creación de switches, nodos y enlaces de red.

- Se realizaron múltiples pruebas de conectividad enviando y recibiendo paquetes e intercambiando mensajes entre todos los nodos de red sin problemas de pérdida de los mismos. Gracias al analizador de tráfico Wireshark se pudo observar y capturar el flujo de tráfico existente en la red simulada, y mediante una aplicación **xterm** se comprobó la funcionalidad de cada host de manera independiente realizando pruebas individuales. Esto ha permitido demostrar el cumplimiento de los objetivos planteados al inicio de este proyecto, ya que se han aplicado los conocimientos de redes SDN, Mininet, Python y otros en la simulación de una Red Definida por Software.

5.2. RECOMENDACIONES.

- A los responsables de nuevos proyectos relacionados con SDN, se recomienda primeramente hacer una revisión sobre los fundamentos, arquitectura, dispositivos, medios, protocolos y estándares que rigen a este tipo de redes de última generación, así como también sobre virtualización de servidores y software de administración de redes, ya que esto permitirá tener una visión clara sobre las nuevas herramientas para la gestión para este tipo de redes.

- A los Docentes del Área de Redes, incluir en sus sílabos temáticas sobre redes SDN en especial sobre sus características, arquitectura, dispositivos, controladores (NOX, POX, Beacon, Floodlight), protocolo OpenFlow, Simuladores (Mininet), requerimientos de hardware y software, analizadores de tráfico, entre otros. Esto permitirá fortalecer en el estudiante el desarrollo de sus competencias en el campo de las redes de datos.

- A la gestores de conocimiento de la Universidad Tecnológica Equinoccial, proponer otros proyectos relacionados con SDN, en especial alguna aplicación práctica, que permita implementar esta metodología y conocer con mayor detalle el comportamiento de una red SDN real, con usuarios, seguridades, aplicaciones y otras, para tener una mejor visión de su diseño, implementación, configuración y administración de la misma para lograr expertía y conocimiento sobre redes de última generación.

- A los administradores de red que estén pensando en implementar y configurar de forma real una red definida por software deben tener presente que se necesitan disponer de equipos físicos compatibles con SDN y con su protocolo OpenFlow, ya que en el *firmware* de estos equipos se encontrará toda la lógica de la topología de red, de

esta manera se podrá actualizar y gestionar la nueva red. La teoría recomienda asegurarse de que todos los dispositivos sean de la misma marca para que exista compatibilidad entre ellos, caso contrario, pueden presentar problemas de inconsistencias y conectividad de la misma.

- Si actualmente se desearía realizar una implementación de este tipo sería una decisión muy temprana ya que actualmente las redes SDN se encuentran en estudios y no se cuenta con soluciones de red específicas en cuanto a seguridad, por lo que esto es un punto de vulnerabilidad y preocupación, dejando una brecha para los atacantes, por ello mientras se afianzan estas seguridades es mejor esperar un poco para su implementación en empresas que manejen información relevante.

- A los Administradores de la red de datos de la UTE se recomienda en un futuro tomar a consideración este proyecto de titulación como ayuda y referencia para la posible implementación de una red SDN real en la red de la UTE.

- Una vez que el país se cuente con todos los equipos de tecnología SDN y OpenFlow y ya se pueda implementar de forma real una red SDN se recomienda utilizar diferentes lenguajes de programación tales como: C++, Java, Python para configurar sus controladores.

GLOSARIO DE TÉRMINOS

ANSI: (*American National Standards Institute*), Instituto Nacional Americano de Normalización, organización que supervisa el desarrollo de estándares para productos, servicios, procesos, networking y sistemas en los Estados Unidos.

API: (*Application Programming Interface*), es una interfaz de programación de aplicaciones donde se programan métodos, procedimientos o funciones de software que permiten realizar una acción específica.

Beacon: Este software está basado en Java donde al igual que los demás controladores permite desarrollar políticas y reglas de flujo de manera más fácil y dinámica.

CIDR: (*Classless Inter-Domain Routing*), Enrutamiento entre dominios sin clases permite flexibilidad al dividir rangos de direcciones IP en subredes separadas.

CLOUD COMPUTING: (Computación en la nube), servidores desde Internet encargados de atender las peticiones en cualquier momento. Se puede tener acceso a su información o servicio, mediante una conexión a internet desde cualquier dispositivo móvil o fijo ubicado en cualquier lugar.

DNS: (*Domain Name System*), Sistema de nombres de dominio permite traducir los nombres de dominios en las redes en protocolos de internet para conocer la maquina donde se encuentra un dominio al cual se desee acceder.

Floodlight*:* Este controlador así se encuentra escrito en Java permitiendo que se ejecute sobre su propia máquina virtual y facilitando una fácil configuración con dependencias mínimas.

FTP: (*File Transfer Protocol*), Protocolo de transferencia de archivos entre clientes conectados a una red mediante la arquitectura cliente-servidor.

HAN: (*Home Area Network*), Red de área domestica son redes de comunicación que facilitan la interconexión entre diferentes dispositivos digitales o equipos existentes en una casa.

HTTP: (*HyperText Transfer Protocol*), Protocolo de transferencia de hipertexto es un protocolo que se utiliza para la transferencia de hipertextos en las transacciones *de world wide web* (www).

IEEE: (*Institute of Electrical and Electronics Engineers*), Instituto de Ingenieros Eléctricos y Electrónicos, es una organización de profesionales especializados en el desarrollo de estándares y normas de comunicaciones y redes.

IP: Etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz dentro de una red que utilice el protocolo IP.

IPv4: Es un protocolo de Internet de 32 bits, que se utiliza para enviar información entre dispositivos, este sistema asigna una serie de cuatro números decimales separados por puntos, cada uno de los cuales está comprendido entre 0 y 255 a cada dispositivo.

IPv6: Es una nueva versión del Protocolo de Internet diseñado para reemplazar al IPv4, tiene una longitud de 128 bits, así esta versión soporta un mayor espacio de direcciones.

ISDN: (*Integrated Services Digital Network*), Red digital de servicios integrados facilita conexiones digitales de extremo a extremo para proporcionar servicios a la que los usuarios pueden acceder a través de un conjunto de interfaces normalizadas.

ISO: (*International Organization for Standardization*), Organización de Estándares Internacionales, organización encargada de promover el desarrollo de normas internacionales de fabricación entre ellas normas referentes al *neworking*.

ITU: Unión Internacional de Telecomunicaciones, organismo encargado de regular las telecomunicaciones.

LAN: (*Local Area Network*), Red de área local es un conjunto de dispositivos que conforman una red de computadoras de pequeña extensión, tales como oficinas.

MAN: (*Metropolitan Area Network*), Red de área metropolitana es un conjunto de dispositivos interconectados dentro de un área metropolitana y zona geográfica extensa, tales como municipios o ciudades.

Middlebox: Es una red de computadoras o dispositivos que inspecciona y manipula el tráfico con fines de reenvío de paquetes.

Mininet: Es un programa emulador de red que maneja virtualmente un conjunto de dispositivos finales, switches, routers y enlaces en un solo núcleo de Linux.

NOX: Es una plataforma de código abierto programado en C++, este permite desarrollar la creación de redes definidas por software, en donde este será el controlador de la red, permitiendo gestionar y administrar la red SDN.

OpenFlow: Protocolo abierto de comunicaciones que permite a un servidor de software determinar la ruta y el mejor camino de reenvío de paquetes que debería seguir una red de switches.

OSI: (*Open System Interconnection*) Modelo de interconexión de sistemas abiertos como referencia para la definición de arquitecturas en la interconexión de los sistemas de comunicaciones.

PAN: (*Personal Area Network*), Red de área personal un conjunto de dispositivos terminales personalizados ubicados dentro de un domicilio, casa.

PDU: (*Protocol Data Unit*), Unidad de datos de protocolo se utilizan para el intercambio de datos entre unidades disparejas, dentro de una capa del modelo OSI.

POX: De la misma manera que NOX es una plataforma de código abierto pero además de utilizar C++ este ya integra nuevas funcionalidades en una API de Python que permite desarrollar de manera amigable reglas de flujo y lógica de la red SDN.

Python*:* Lenguaje de programación multiplataforma que soporta programación orientada a objetos que permite desarrollar aplicaciones de una manera fácil y legible.

Script*:* Programa o archivo de órdenes de texto plano que permite realizar tareas específicas, interactuar con el sistema operativo y el usuario.

SDN*:* (*Software Defined Network*), conjunto de técnicas relacionadas con el área de redes computacionales, cuyo objetivo es facilitar la implementación e implantación de servicios de red de manera personalizada gestionando la red mediante un controlador.

SMTP: (*Simple Mail Transfer Protocol*), Protocolo para la transferencia simple de correo electrónico que se utiliza para la transferencia de correo electrónico entre computadores y dispositivos.

SNMP: (*Simple Network Management Protocol*), Protocolo simple de administración de red permite el intercambio de cierta información entre equipos de red.

SSH: (*Secure Shell*), Intérprete de órdenes seguras, es un protocolo que permite acceder de manera remota a máquinas a través de una red.

SSL: (*Secure Socket Layer*), Capa de conexión segura es un conjunto de reglas relacionadas con seguridad que se aplica para una conexión segura entre cliente y un servidor.

TCP: (*Transmission Control Protocol*), Protocolo de control de transmisión es uno de los protocolos fundamentales en Internet. En las redes de comunicación se utiliza TCP para crear conexiones entre sí, a través de las cuales puede enviarse un flujo de datos.

TCP/IP: Conjunto de guías generales de diseño e implementación de protocolos de red específicos para permitir que un equipo pueda comunicarse en una red.

TFTP: (*Trivial file transfer Protocol*), protocolo de transferencia de archivos trivial muy simple que se utiliza para transferir pequeños archivos entre ordenadores desde un servidor de red.

UDP: (*User Datagram Protocol*), Protocolo de datagrama de usuario es protocolo del nivel de transporte basado en el intercambio de datagramas que trabaja en la capa 4 Modelo OSI.

WAN: (*Wide Area Network*), Red de área extendida es un conjunto de computadoras y dispositivos terminales de gran extensión que puede comunicarse entre ciudades y países.

Wireshark: Analizador de tráfico de red que muestra los diferentes protocolos utilizados para realizar análisis y solucionar problemas en redes de comunicaciones.

WLAN: (*Wireless Area Network*), Red de área local inalámbrica son redes que utilizan como medio de transmisión el aire y no necesitan de medios físicos, como por ejemplo Wi-Fi.

Xterm: Es un emulador de terminal para el sistema de ventanas X Window System.

BIBLIOGRAFÍA

- Andreu Cabezón. F., Pellejero. I., & Lesta Amaia. (2006). Fundamentos y aplicaciones de seguridad en redes WLAN, España: MARCOBOMBO S.A.

- Barceló Ordinas, J. M., Iñigo Griera, J., Martí Escalé, R., Peig Olivé, E., & Perramon Tornil, X. (2004). *Software Libre.* Recuperado el 16 de Septiembre de 2014, de http://ocw.uoc.edu/computer-science-technology-and-multimedia/computer-networks/computer-networks/XP06_M2105_01496.pdf

- Barcenilla, C. A., & Eijo, A. (2004). *Universidad Tecnológica Nacional.* Recuperado el 10 de Septiembre de 2014, de http://www4.ipv6.frlp.utn.edu.ar/downloads/Propuesta%20direccionamiento%20IPv6%20UTN.pdf

- Blandón Gómez, D. (2013). OPENFLOW EL PROTOCOLO DEL FUTURO. Recuperado el 20 de Octubre de 2014, de https://www.google.com.ec/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&uact=8&ved=0CCsQFjAC&url=http%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F4897871.pdf&ei=XB46VZTwF_WOsQTQvYDoBg&usg=AFQjCNGwGimg7lrz2svXELyMSYoiS9GlTw&bvm=bv.91427555,d.cWc

- Cruz Álvarez, Melo Quiñónez, & Rodríguez Sierra. (2008). Instituto Politécnico Nacional. Recuperado el 05 de Octubre de 2014, de http://itzamna.bnct.ipn.mx/dspace/bitstream/123456789/438/1/LUCERNA.pdf

- Erickson D., What is beacon.(2013). Recuperado el 20 de Febrero de 2015, de https://openflow.stanford.edu/display/Beacon/Home

- Esparza Morocho, J. (2013). Escuela Politécnica Nacional. Recuperado el 10 de Octubre de 2014, de http://bibdigital.epn.edu.ec/bitstream/15000/6056/1/CD-4785.pdf

- Heller, B. (2011). Openflow. w e b d e O p e n fl o w , Recuperado el 14 de Octubre de 2014, de http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial

- Laboratorio Nacional de Calidad del Software. (2009). INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA. España.

- Marzal, A., Gracia, I.(2009). Introducción a la programación con Python. Recuperado el 05 de Mayo de 2015, de http://www.uji.es/bin/publ/edicions/ippython.pdf

- ORACLE, Guía de administración del sistema: servicios IP, 2010. Recuperado el 22 de Febrero de 2015, de http://docs.oracle.com/cd/E24842_01/html/820-2981/index.html.

- Siamak, A. (2013). Software Defined Networking with OpenFlow. UK: Packt Publishing.

- Stallings, William. (2004). Comunicaciones y redes de computadores. Madrid: Pearson Educacion.

- TANENBAUM, ANDREW S. (2003). Redes de computadoras. México: Pearson Educación.

- Thomas D., Nadeau,. & Ken Gray. (2013). SDN: Software Defined Networks. Recuperado el 22 de Agosto de 2014, de http://cdn.oreillystatic.com/oreilly/booksamplers/9781449342302_sampler.pdf

- Velásquez, W. (2014). Emulación de una red definida por software utilizando MiniNet. Recuperado el 05 de Agosto de 2014, de https://www.academia.edu/5730624/Emulaci%C3%B3n_de_una_red_definida_por_software_utilizando_MiniNet

- Vivek. T. (2013). SDN and OPENFLOW FOR BEGINNERS WITH HANDS ON LABS. Recuperado el 30 de Junio de 2014, de http://2doubleccies.com/downloads/SDN-and-Openflow.pdf

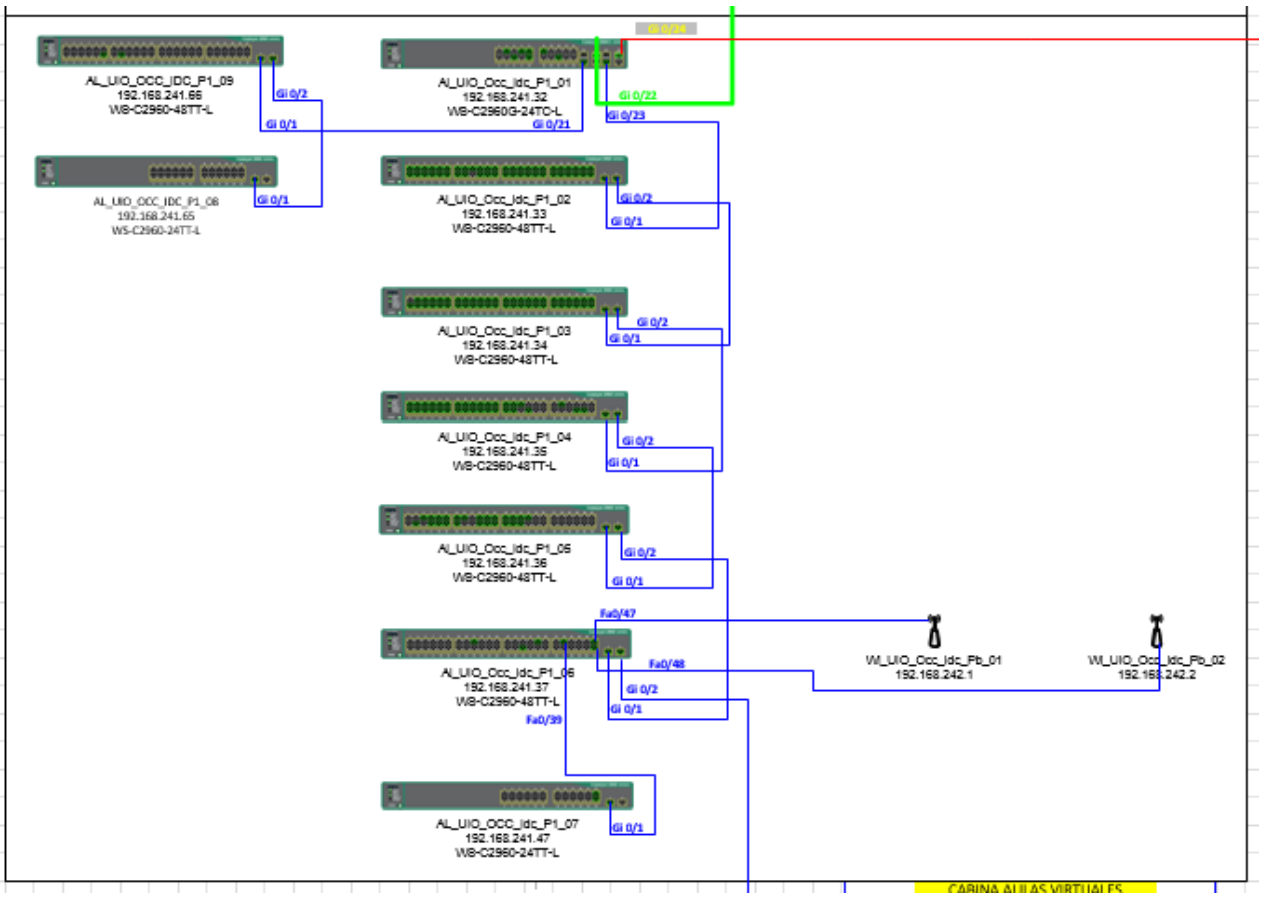
- VMWARE, Virtualización de servidores. Recuperado el 10 de Marzo de 2015, de http://www.vmware.com/es/virtualization

ANEXOS

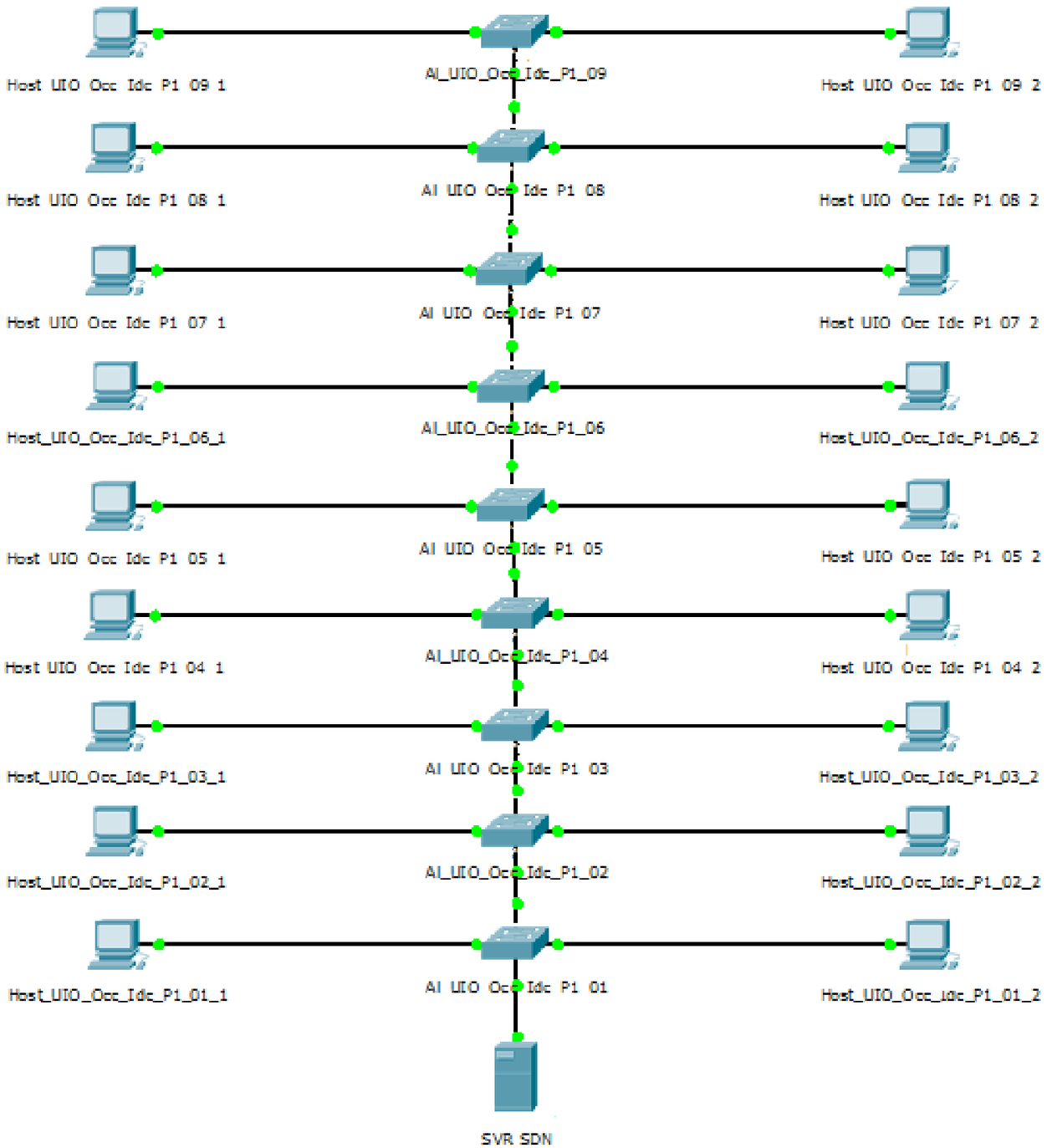
Anexo 1: **Plano de la red física actual de la red de switches de la Universidad Tecnológica Equinoccial.**

ANEXO 1

Plano de la red física actual de la red de switches de la Universidad Tecnológica Equinoccial.



Anexo 2: **Plano de la red Rediseño de la red de Switches de la Universidad Tecnológica Equinoccial, aplicando Redes Definidas por Software (SDN) para el piso 1 del Bloque IDIC del Campus Matriz Quito Av. Mariana de Jesús y Occidental.**



SIMBOLOGÍA

SVR SDN

Host

Switch

Cable cruzado

Cable directo

DISPOSITIVO	NOMBRE	IP	MÁSCARA
h1	Host_UIO_Occ_Idc_P1_01_1	10.0.0.1	255.0.0.0
h2	Host_UIO_Occ_Idc_P1_01_2	10.0.0.2	255.0.0.0
h4	Host_UIO_Occ_Idc_P1_02_1	10.0.0.4	255.0.0.0
h5	Host_UIO_Occ_Idc_P1_02_2	10.0.0.5	255.0.0.0
h7	Host_UIO_Occ_Idc_P1_03_1	10.0.0.7	255.0.0.0
h8	Host_UIO_Occ_Idc_P1_03_2	10.0.0.8	255.0.0.0
h10	Host_UIO_Occ_Idc_P1_04_1	10.0.0.10	255.0.0.0
h11	Host_UIO_Occ_Idc_P1_04_2	10.0.0.11	255.0.0.0
h13	Host_UIO_Occ_Idc_P1_05_1	10.0.0.13	255.0.0.0
h14	Host_UIO_Occ_Idc_P1_05_2	10.0.0.14	255.0.0.0
h16	Host_UIO_Occ_Idc_P1_06_1	10.0.0.16	255.0.0.0
h17	Host_UIO_Occ_Idc_P1_06_2	10.0.0.17	255.0.0.0
h19	Host_UIO_Occ_Idc_P1_07_1	10.0.0.19	255.0.0.0
h20	Host_UIO_Occ_Idc_P1_07_2	10.0.0.20	255.0.0.0
h22	Host_UIO_Occ_Idc_P1_08_1	10.0.0.22	255.0.0.0
h23	Host_UIO_Occ_Idc_P1_08_2	10.0.0.23	255.0.0.0
h25	Host_UIO_Occ_Idc_P1_09_1	10.0.0.25	255.0.0.0
h26	Host_UIO_Occ_Idc_P1_09_2	10.0.0.26	255.0.0.0
s3	AI_UIO_Occ_Idc_P1_01	NO IP	NO MASK
s6	AI_UIO_Occ_Idc_P1_02	NO IP	NO MASK
s9	AI_UIO_Occ_Idc_P1_03	NO IP	NO MASK
s12	AI_UIO_Occ_Idc_P1_04	NO IP	NO MASK
s15	AI_UIO_Occ_Idc_P1_05	NO IP	NO MASK
s18	AI_UIO_Occ_Idc_P1_06	NO IP	NO MASK
s21	AI_UIO_Occ_Idc_P1_07	NO IP	NO MASK
s24	AI_UIO_Occ_Idc_P1_08	NO IP	NO MASK
s27	AI_UIO_Occ_Idc_P1_09	NO IP	NO MASK

Fecha: 2015/04/01	UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL
Dibujado por: Lucia Mejía	**REDISEÑO DE LA RED DE SWITCHES DE LA UNIVERSIDAD TECNOLÓGICA EQUINOCCIAL, APLICANDO REDES DEFINIDAS POR SOFTWARE (SDN) PARA EL PISO 1 DEL BLOQUE IDIC DEL CAMPUS OCCIDENTAL.**
Revisado por: Ing. Bolívar Jácome	

Anexo 3: **Manual de Instalación y configuración de Virtual Box y Mininet, para la simulación de la red de Switches de la Universidad Tecnológica Equinoccial aplicando redes SDN.**

ANEXO 3

Manual de Instalación y configuración de VirtualBox y Mininet, para la simulación de la red de Switches de la Universidad Tecnológica Equinoccial aplicando redes SDN.

A continuación se hace una explicación de todo el proceso que se debe realizar para la instalación e implementación de un prototipo de red SDN, en este caso para el rediseño de la red de Switches de la Universidad Tecnológica Equinoccial aplicando redes definidas por software (SDN) para el piso 1 del Bloque IDIC del Campus Matriz Quito Av. Mariana de Jesús y Occidental.

- **Pasos para una instalación Máquina Virtual VirtualBox:**

1.- Descargar e instalar VirtualBox (libre) para Windows o Linux, disponible en el sitio: https://www.virtualbox.org/wiki/Downloads.

2.- Una vez descargado el instalador, hacer doble click y seleccionar la opción ejecutar.

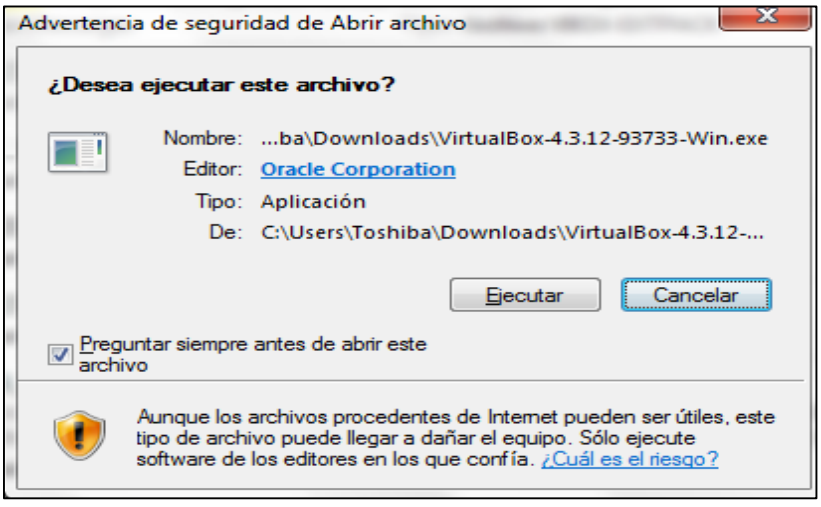


Figura 1. Ejecución del asistente de instalación de la máquina virtual

3.- Después de ejecutar el instalador se abrirá la primera pantalla donde se debe seleccionar next.

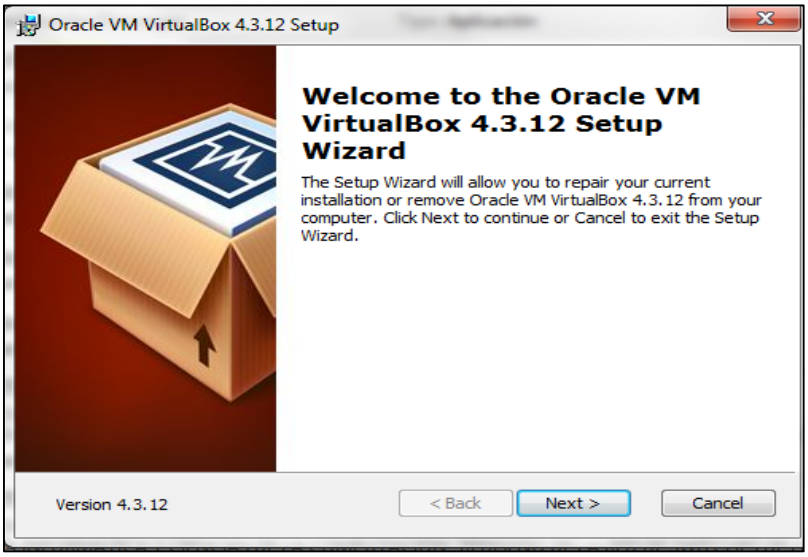


Figura 2. Setup de instalación

4.- En la siguiente pantalla se eligen los componentes y la ruta donde se instalará la máquina virtual, seleccionar next.

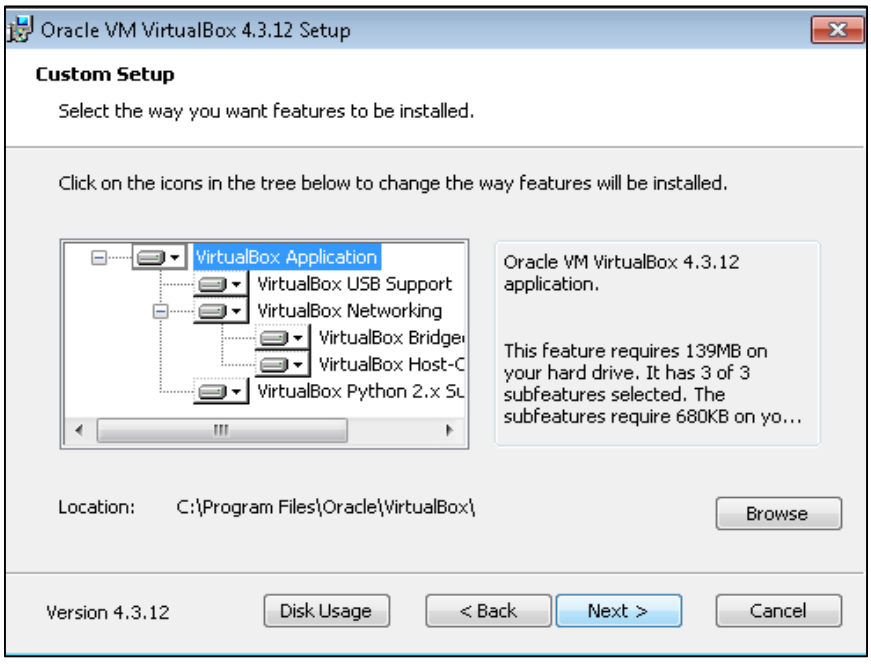


Figura 3. Componentes y ruta de instalación

5.- En la siguiente pantalla se eligen los accesos directos que desean configurar, seleccionar next.

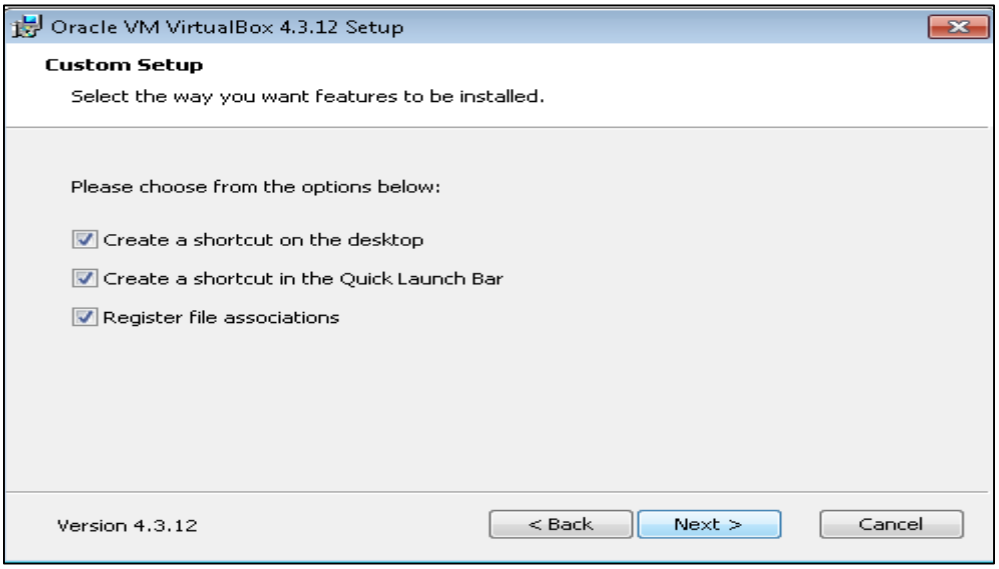


Figura 4. Accesos directos de la máquina virtual

6.- En la siguiente pantalla que se van a copiar los archivos del programa, se reiniciarán las tarjetas de red y se instalará VirtualBox, seleccionar install e inmediatamente se procede a la finalización de la instalación.

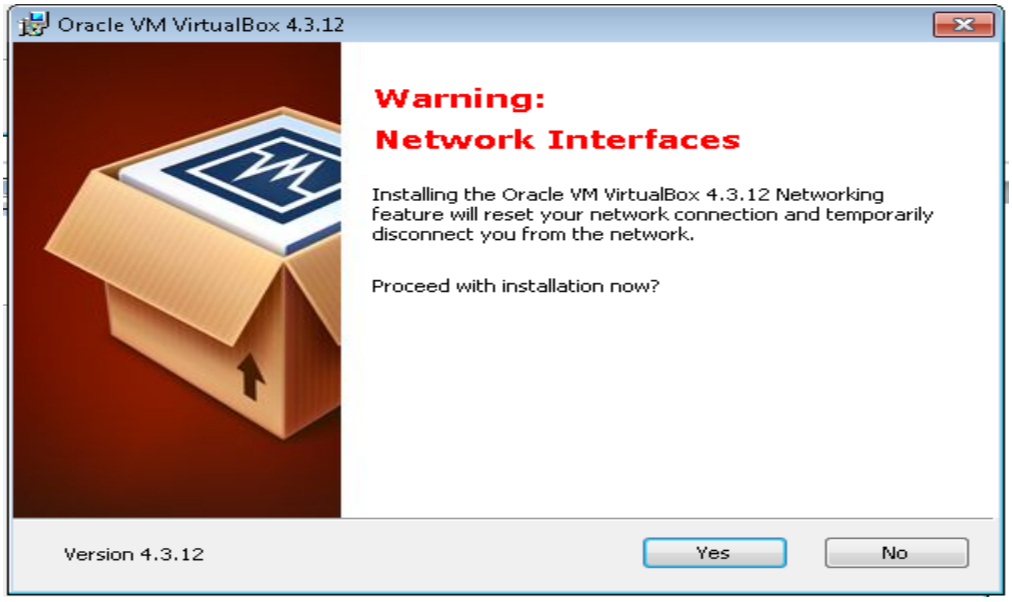


Figura 5. Reiniciar tarjetas de red

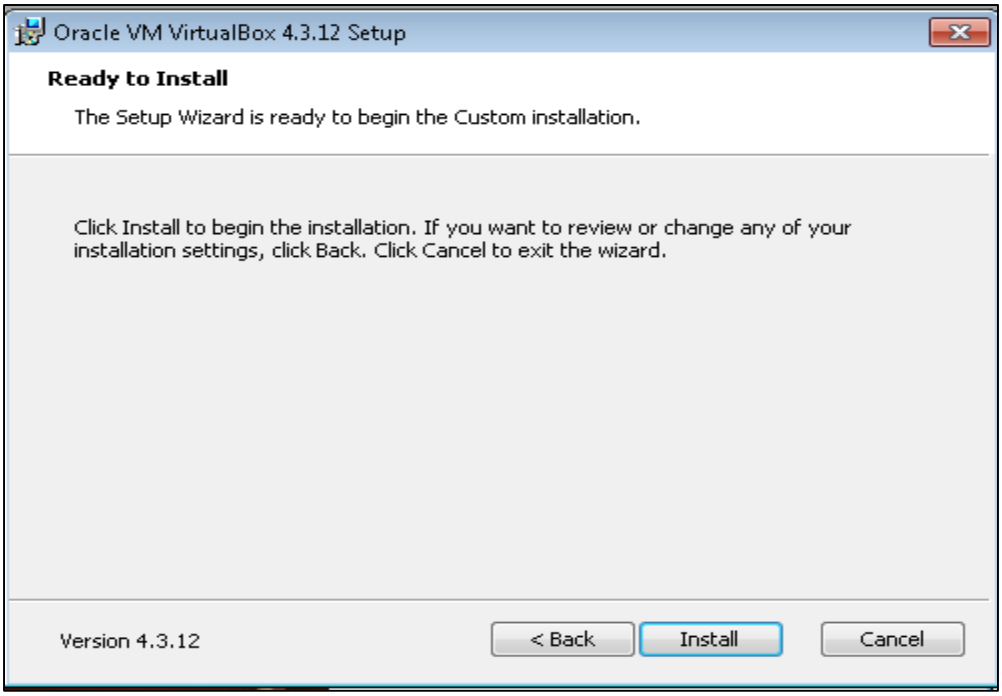


Figura 6. Instalación de la máquina virtual

7.- A continuación se indican las pantallas del proceso de la instalación, tardará pocos minutos hasta su finalización.

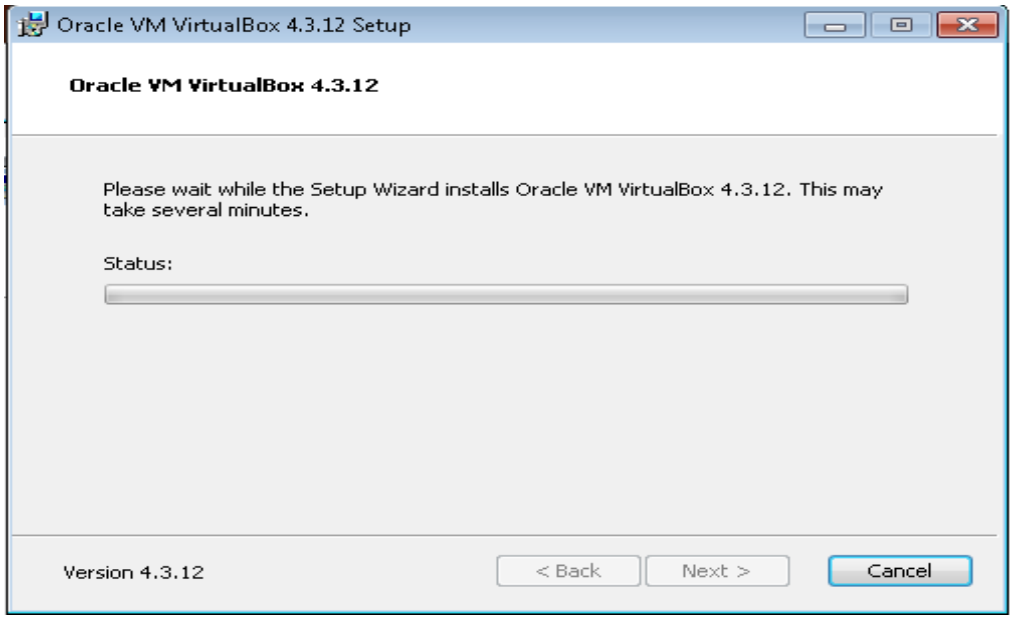


Figura 7. Proceso de instalación

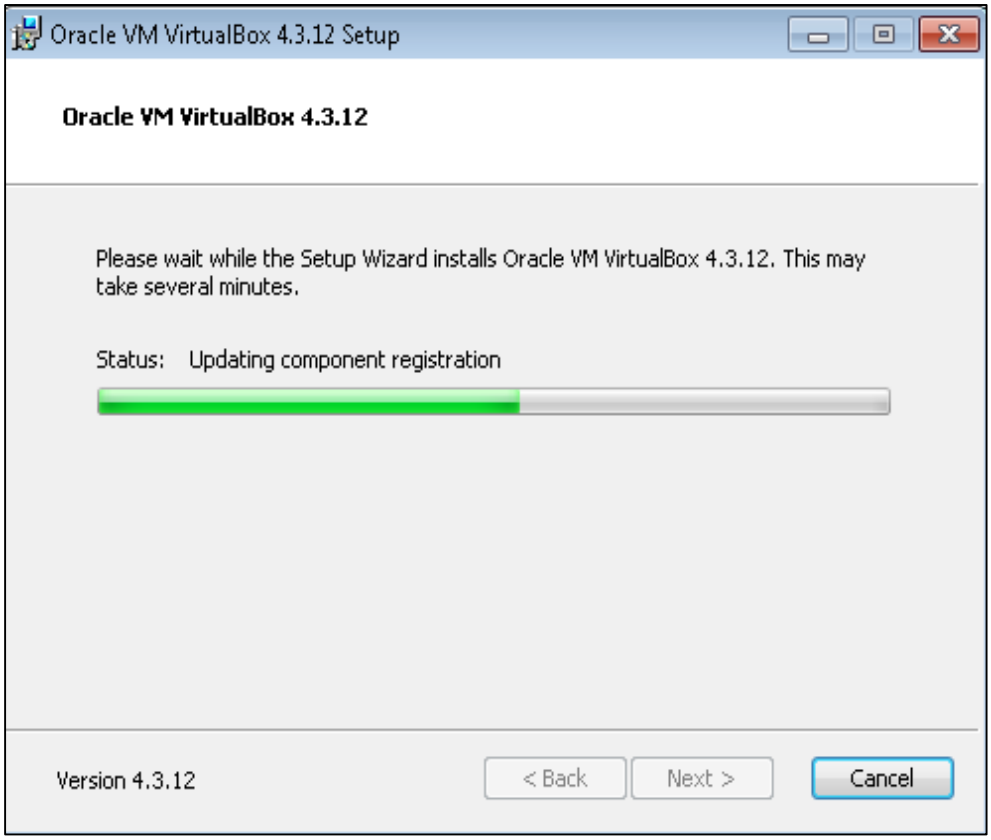


Figura 8. Proceso de instalación

8.- En siguiente pantalla se indica que se deben instalar dispositivos necesarios para la máquina virtual, seleccionar instalar.

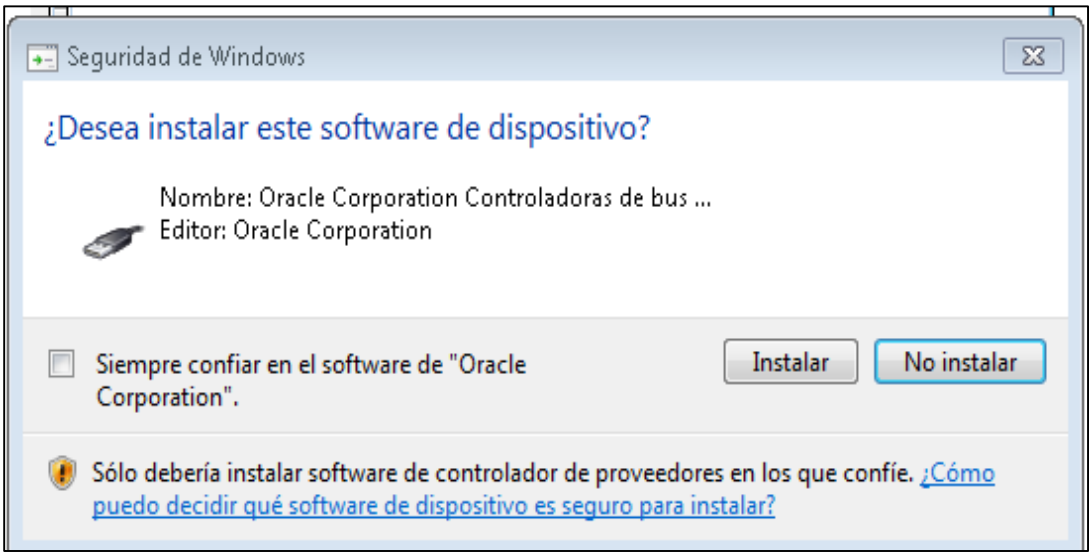


Figura 9. Dispositivos de instalación

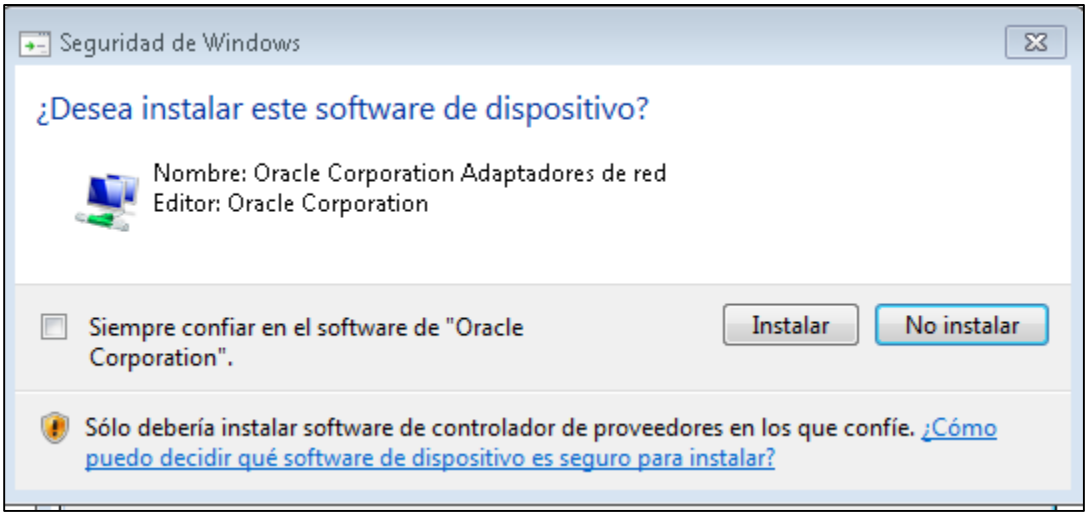


Figura 10. Dispositivos de instalación

9.- Finalmente esperar que el asistente de instalación termine, y luego click en Finish.

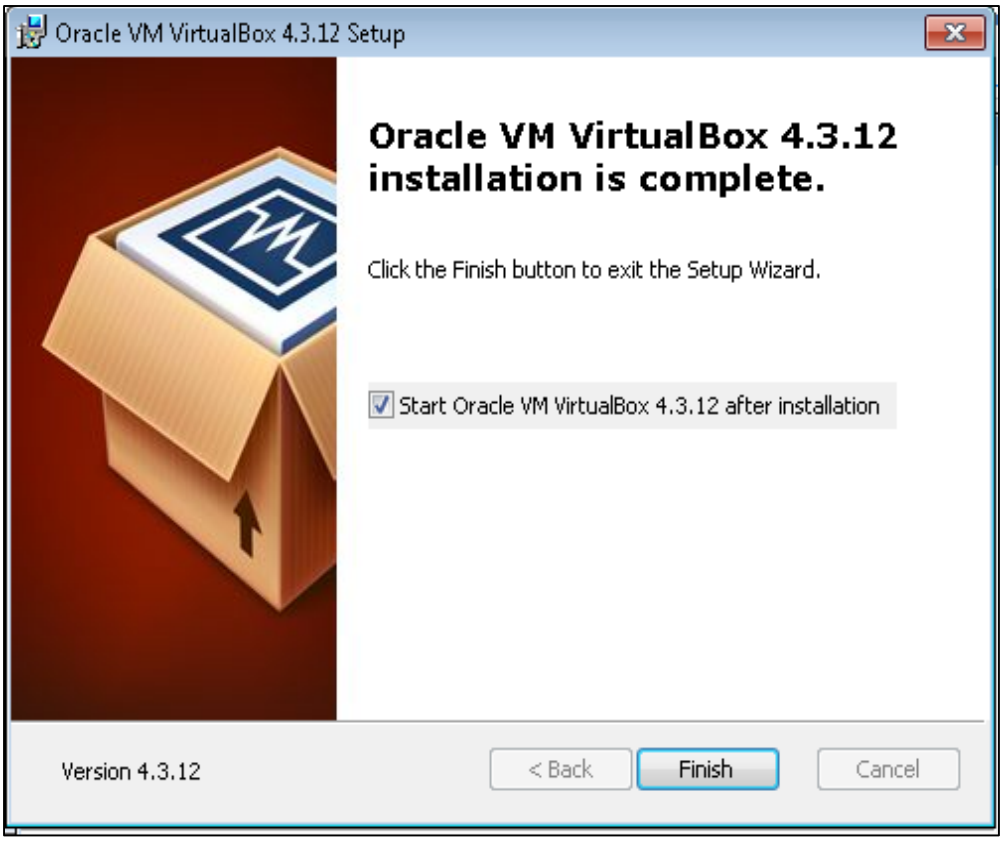


Figura 11. Finalización de la instalación

- **Pasos para la instalación de Mininet:**

A continuación se explica paso a paso el proceso de instalación y configuración de la máquina virtual Mininet.

1.- Descargar una imagen del emulador Mininet, disponible en el sitio: https://github.com/downloads/Mininet/Mininet/Mininet-vm-ubuntu11.10-052312.vmware.zip. El archivo comprimido tiene un tamaño de 885 MB, hay que descomprimirlo para su correcta importación.

2.- Una vez instalado VM VirtualBox se procede a la importación de la máquina virtual Mininet, para esto se selecciona el archivo **Mininet-VM.vmdk** descargado previamente como una imagen de disco duro y posteriormente iniciar el programa.

3.- Hacer click en el botón Nueva, para crear la nueva máquina virtual.

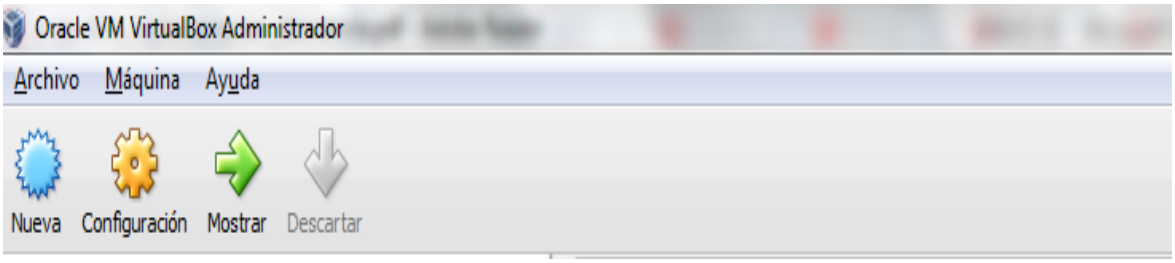


Figura 12. Creación nueva máquina virtual

4.- A continuación aparece una ventana de asistencia en donde se deben colocar las instrucciones indicadas en la ventana y seleccionar Next.

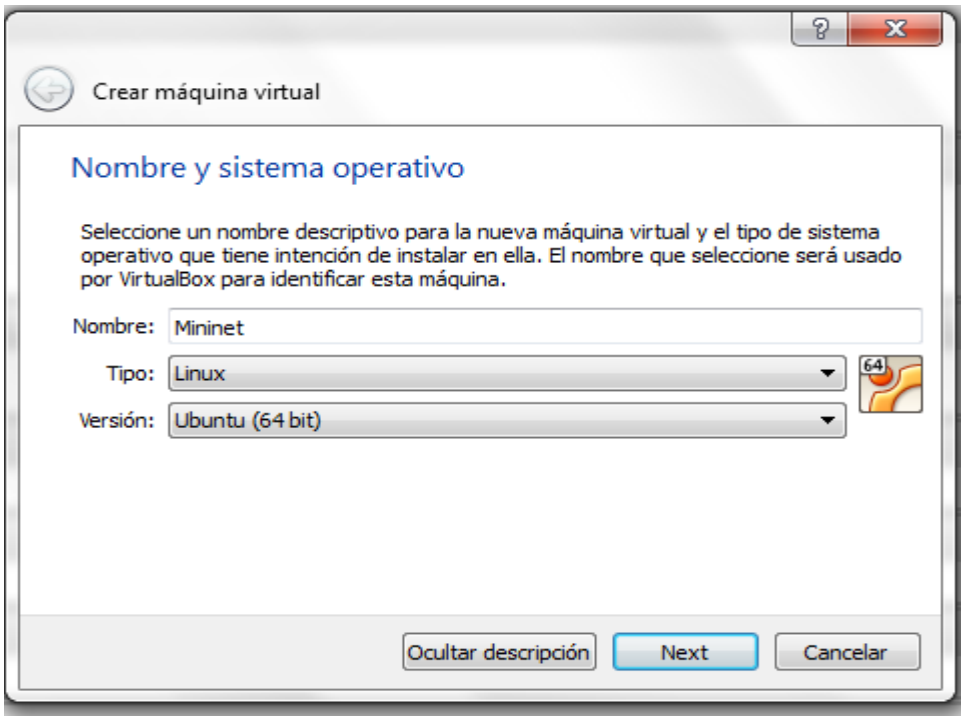


Figura 13. Creación Mininet

5.- El siguiente paso es definir el tamaño de memoria, el cual se asignó a 1024 MB de RAM, el tamaño mínimo que se puede asignar es de 512 MB.

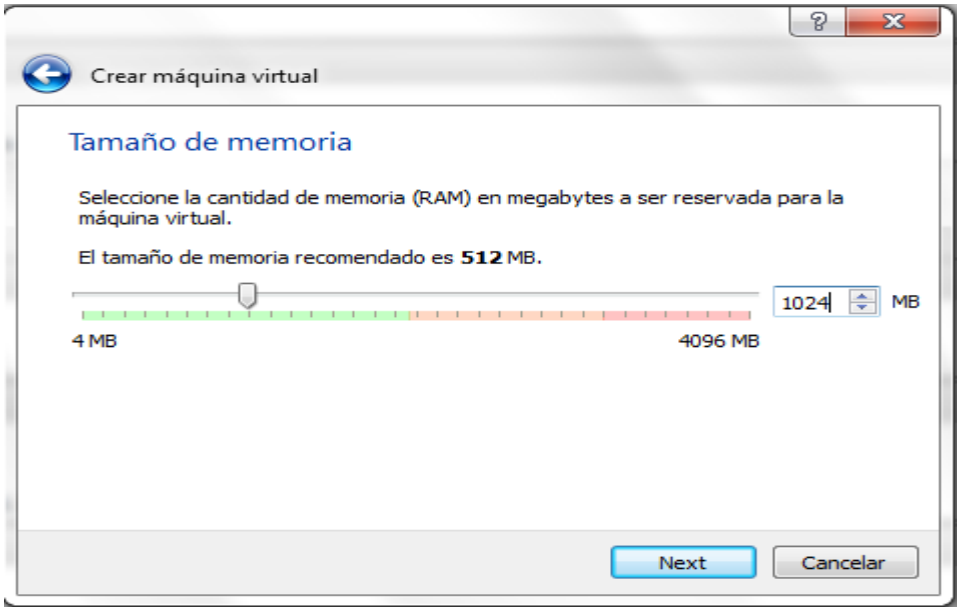


Figura 14. Asignación tamaño

6.- Se seleccionará la unidad de disco, para lo cual se ubica el archivo **Mininet VM.vmdk.** que se descargó previamente y seleccionar crear.

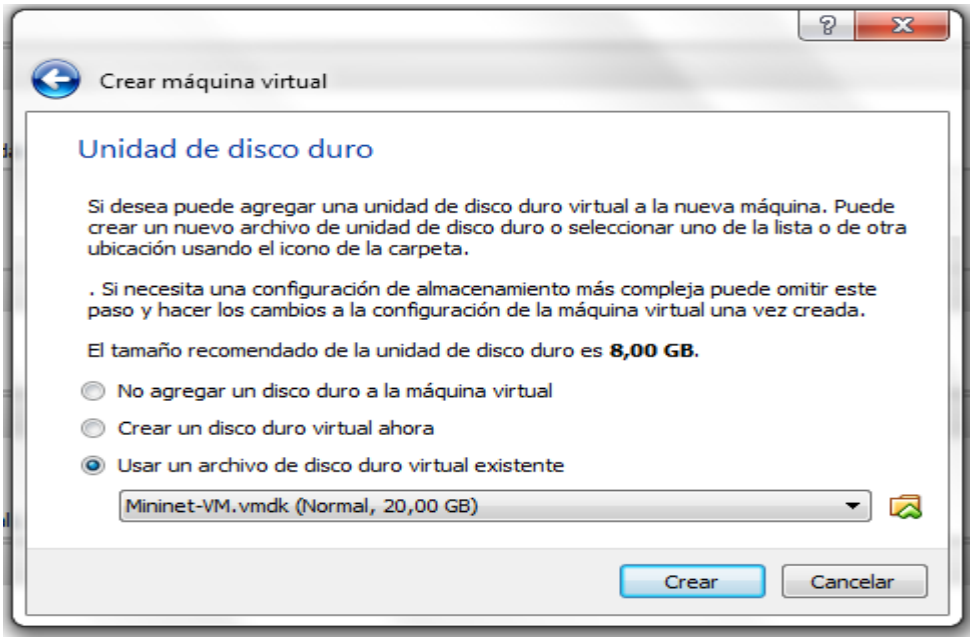


Figura 15. Ubicación archivo Mininet Mininet VM.vmdk

7.- A continuación se crea la máquina virtual de Mininet y se deben realizar modificaciones para la correcta ejecución y creación de la red SDN, se deben habilitar una nueva tarjeta de red, para esto los pasos a seguir son los siguientes:

- Configuración
- Red
- Adaptador 2
- Seleccionar conectado a Adaptador solo-anfitrión.
- Y VirtualBox Host-only Ethernet Adapter

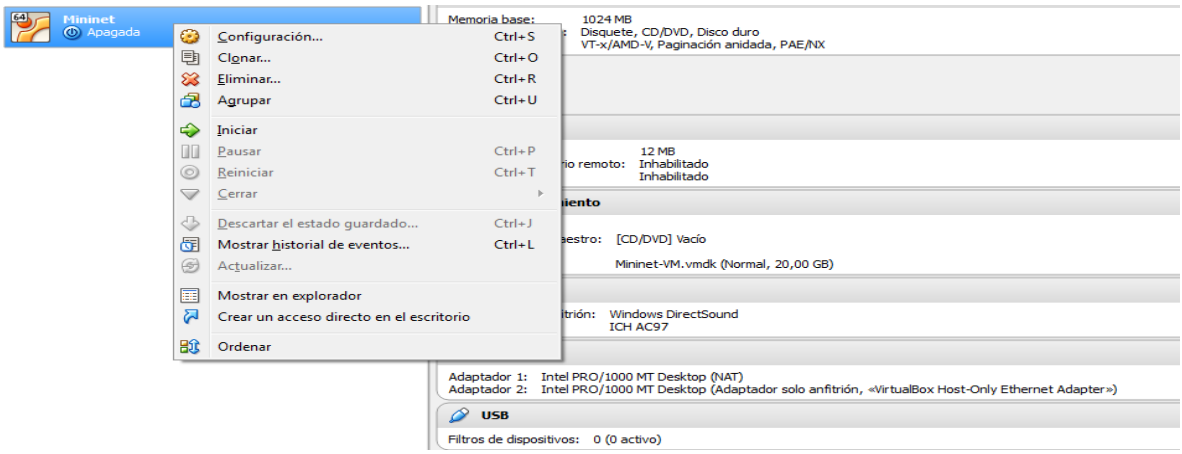


Figura 16. Máquina virtual Mininet creada

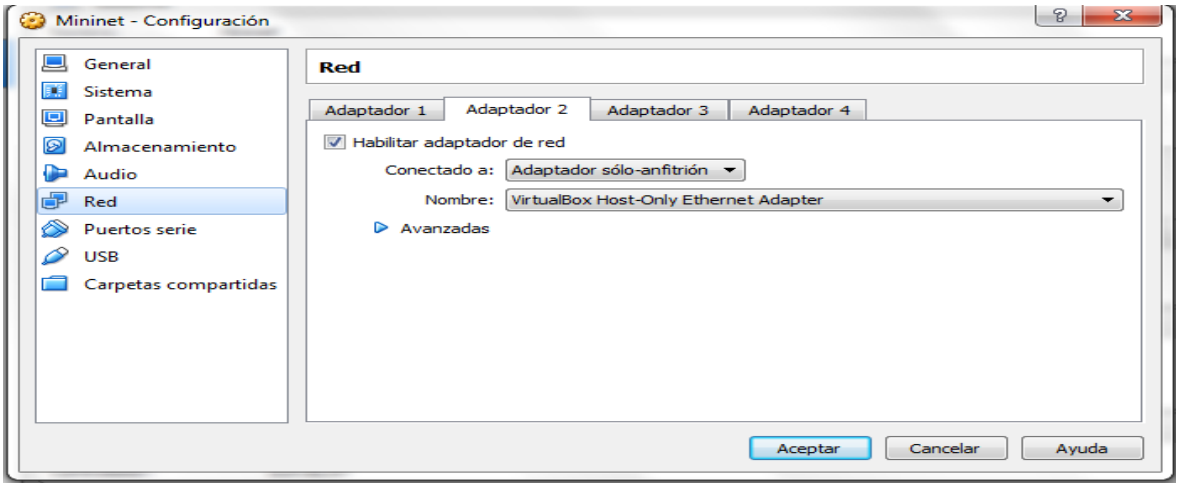


Figura 17. Configuración de red para Mininet

8.- Para que las aplicaciones externas tales como, Wireshark y xterm se puedan ejecutar se debe habilitar la opción PAE/NX que son características extendidas de Mininet, para esto los pasos a seguir son los siguientes:

- Configuración
- Sistema
- Procesador
- Y en el cuadro de Características extendidas habilitar PAE/NX.

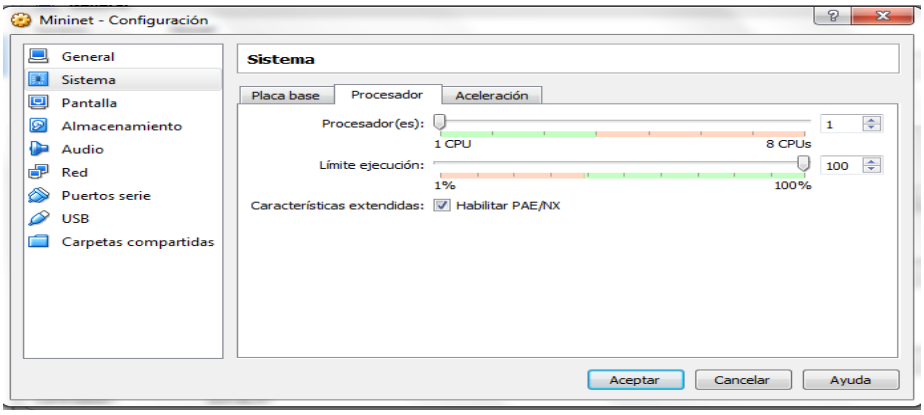


Figura 18. Habilitar características extendidas

9.- Se ingresará a la máquina virtual de Mininet con el login y password **openflow** y está lista para usarse y realizar las diferentes ejecuciones de topologías de red SDN.

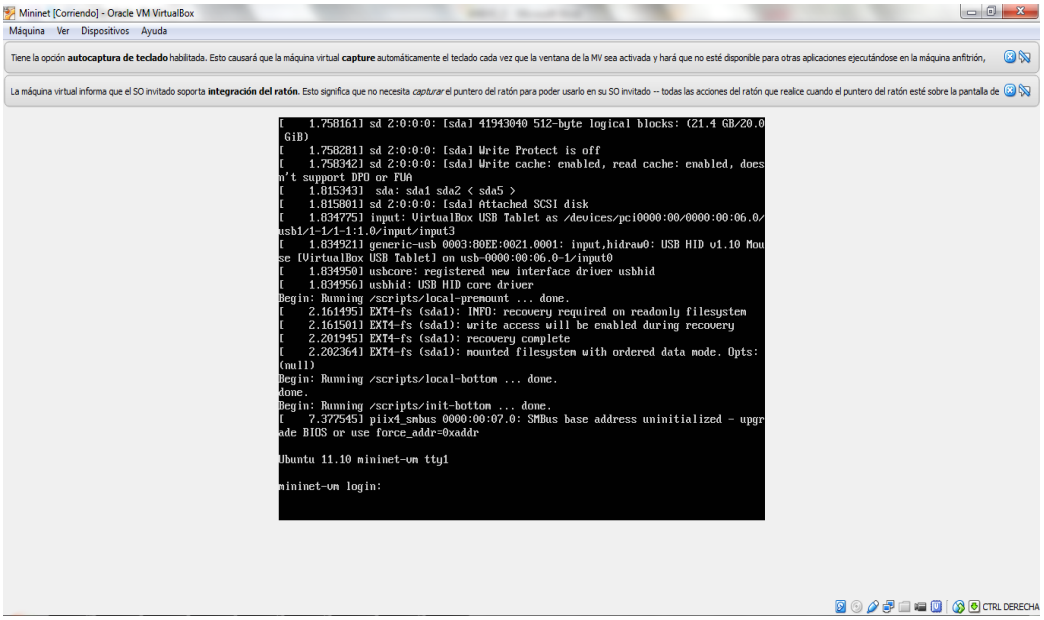


Figura 19. Ingreso a Mininet

10.- Y finalmente una vez ingresado se realizarán todos los procesos y pruebas detallados en el capítulo de Análisis y Resultados.

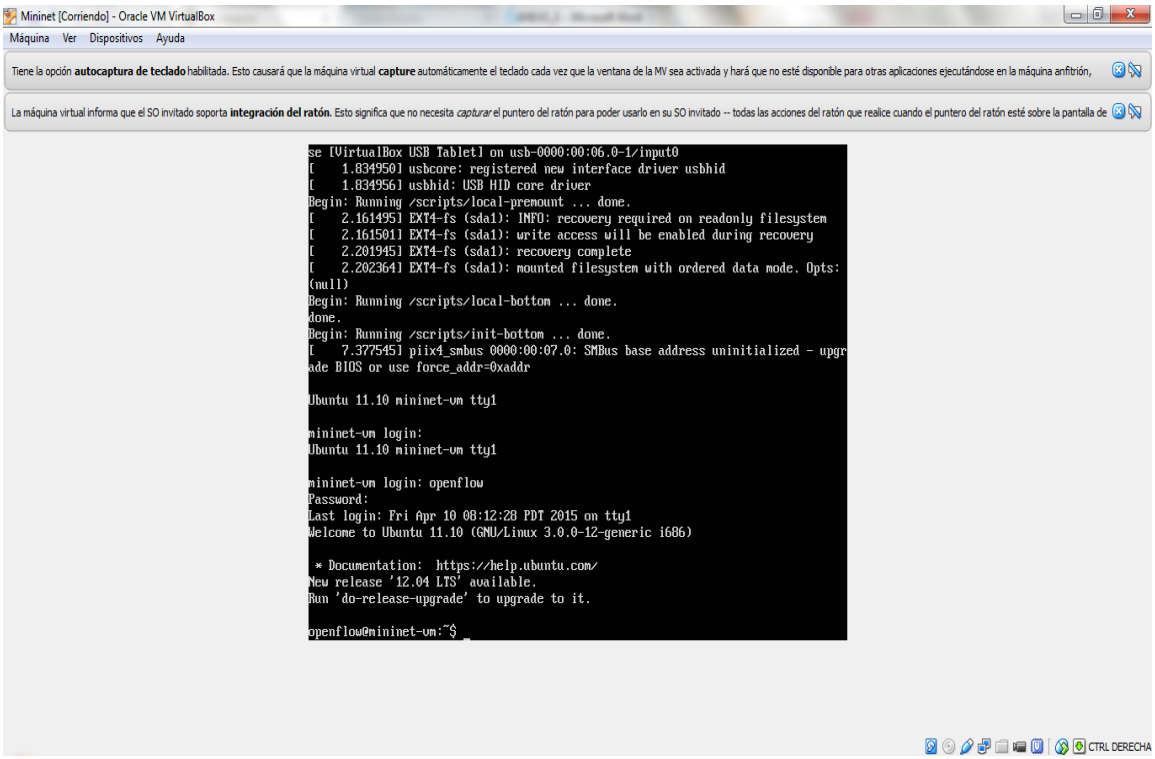


Figura 20. Ingreso a Mininet

Anexo 4: **Código fuente completo de la topología de red de Switches de la Universidad Tecnológica Equinoccial aplicando redes SDN para el piso 1 de los laboratorios del bloque IDIC Campus Matriz Quito Av. Mariana de Jesús y Occidental.**

ANEXO 4

Programa de desarrollo e implementación de la simulación del Rediseño de la red de Switches de la Universidad Tecnológica Equinoccial aplicando redes SDN.

```
from mininet.topo import Topo, Node

class RedUte( Topo ):

   "REDISENO DE LA RED DE SWITCHES DE LA UTE APLICANDO REDES DEFINIDAS POR SOFTWARE."


   def __init__( self, enable_all = True ):

      "REDISENO DE LA TOPOLOGIA PERSONALIZADA."


      # Add default members to class.

      super( RedUte, self ).__init__()


      # En cada host y switch se asigna un id

      Host_UIO_Occ_Idc_P1_01_1 = 1

      Host_UIO_Occ_Idc_P1_01_2 = 2

      Al_UIO_Occ_Idc_P1_01 = 3


      Host_UIO_Occ_Idc_P1_02_1 = 4

      Host_UIO_Occ_Idc_P1_02_2 = 5

      Al_UIO_Occ_Idc_P1_02 = 6


      Host_UIO_Occ_Idc_P1_03_1 = 7
```

Host_UIO_Occ_Idc_P1_03_2 = 8

Al_UIO_Occ_Idc_P1_03 = 9

Host_UIO_Occ_Idc_P1_04_1 = 10

Host_UIO_Occ_Idc_P1_04_2 = 11

Al_UIO_Occ_Idc_P1_04 = 12

Host_UIO_Occ_Idc_P1_05_1 = 13

Host_UIO_Occ_Idc_P1_05_2 = 14

Al_UIO_Occ_Idc_P1_05 = 15

Host_UIO_Occ_Idc_P1_06_1 = 16

Host_UIO_Occ_Idc_P1_06_2 = 17

Al_UIO_Occ_Idc_P1_06 = 18

Host_UIO_Occ_Idc_P1_07_1 = 19

Host_UIO_Occ_Idc_P1_07_2 = 20

Al_UIO_Occ_Idc_P1_07 = 21

Host_UIO_Occ_Idc_P1_08_1 = 22

Host_UIO_Occ_Idc_P1_08_2 = 23

Al_UIO_Occ_Idc_P1_08 = 24

Host_UIO_Occ_Idc_P1_09_1 = 25

Host_UIO_Occ_Idc_P1_09_2 = 26

Al_UIO_Occ_Idc_P1_09 = 27

```
self.add_node(  Al_UIO_Occ_Idc_P1_01, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_02, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_03, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_04, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_05, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_06, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_07, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_08, Node( is_switch=True ) )
self.add_node(  Al_UIO_Occ_Idc_P1_09, Node( is_switch=True ) )
self.add_node( Host_UIO_Occ_Idc_P1_01_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_01_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_02_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_02_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_03_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_03_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_04_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_04_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_05_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_05_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_06_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_06_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_07_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_07_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_08_1, Node(is_switch=False))
```

```
self.add_node( Host_UIO_Occ_Idc_P1_08_2, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_09_1, Node(is_switch=False))
self.add_node( Host_UIO_Occ_Idc_P1_09_2, Node(is_switch=False))

# Add edges

self.add_edge( Host_UIO_Occ_Idc_P1_01_1, Al_UIO_Occ_Idc_P1_01 )
self.add_edge( Host_UIO_Occ_Idc_P1_01_2, Al_UIO_Occ_Idc_P1_01 )
self.add_edge( Al_UIO_Occ_Idc_P1_01, Al_UIO_Occ_Idc_P1_02 )
self.add_edge( Al_UIO_Occ_Idc_P1_02, Host_UIO_Occ_Idc_P1_02_1 )
self.add_edge( Al_UIO_Occ_Idc_P1_02, Host_UIO_Occ_Idc_P1_02_2 )
self.add_edge( Al_UIO_Occ_Idc_P1_02, Al_UIO_Occ_Idc_P1_03 )
self.add_edge( Al_UIO_Occ_Idc_P1_03, Host_UIO_Occ_Idc_P1_03_1 )
self.add_edge( Al_UIO_Occ_Idc_P1_03, Host_UIO_Occ_Idc_P1_03_2 )
self.add_edge( Al_UIO_Occ_Idc_P1_03, Al_UIO_Occ_Idc_P1_04 )
self.add_edge( Al_UIO_Occ_Idc_P1_04, Host_UIO_Occ_Idc_P1_04_1 )
self.add_edge( Al_UIO_Occ_Idc_P1_04, Host_UIO_Occ_Idc_P1_04_2 )
self.add_edge( Al_UIO_Occ_Idc_P1_04, Al_UIO_Occ_Idc_P1_05 )
self.add_edge( Al_UIO_Occ_Idc_P1_05, Host_UIO_Occ_Idc_P1_05_1 )
self.add_edge( Al_UIO_Occ_Idc_P1_05, Host_UIO_Occ_Idc_P1_05_2 )
self.add_edge( Al_UIO_Occ_Idc_P1_05, Al_UIO_Occ_Idc_P1_06 )
self.add_edge( Al_UIO_Occ_Idc_P1_06, Host_UIO_Occ_Idc_P1_06_1 )
self.add_edge( Al_UIO_Occ_Idc_P1_06, Host_UIO_Occ_Idc_P1_06_2 )
self.add_edge( Al_UIO_Occ_Idc_P1_06, Al_UIO_Occ_Idc_P1_07 )
self.add_edge( Al_UIO_Occ_Idc_P1_07, Host_UIO_Occ_Idc_P1_07_1 )
self.add_edge( Al_UIO_Occ_Idc_P1_07, Host_UIO_Occ_Idc_P1_07_2 )
```

```
        self.add_edge( Al_UIO_Occ_Idc_P1_07, Al_UIO_Occ_Idc_P1_08 )
        self.add_edge( Al_UIO_Occ_Idc_P1_08, Host_UIO_Occ_Idc_P1_08_1 )
        self.add_edge( Al_UIO_Occ_Idc_P1_08, Host_UIO_Occ_Idc_P1_08_2 )
        self.add_edge( Al_UIO_Occ_Idc_P1_08, Al_UIO_Occ_Idc_P1_09 )
        self.add_edge( Al_UIO_Occ_Idc_P1_09, Host_UIO_Occ_Idc_P1_09_1 )
        self.add_edge( Al_UIO_Occ_Idc_P1_09, Host_UIO_Occ_Idc_P1_09_2 )
        self.add_edge( Al_UIO_Occ_Idc_P1_08, Al_UIO_Occ_Idc_P1_08 )

# Consider all switches and hosts 'on'

        self.enable_all()

topos = { 'sdn': ( lambda: RedUte() ) }
```

Anexo 5: **Datasheet de Switch con tecnología SDN.**

El Datasheet completo se puede descargar del siguiente link:
https://support.necam.com/kbtools/sdocs.cfm?id=fcbdcb3e-45fa-4ec4-9311-215bd9ab9f81

Contents