UNIVERSIDAD NACIONAL DE CHIMBORAZO
FACULTAD DE INGENIERIA
CARRERA DE ELECTRÓNICA Y TELECOMUNICACIONES

ANÁLISIS E IMPLEMENTACIÓN DE UN PROTOTIPO DE RED SDN EN EL CAMPUS NORTE DE LA UNACH

Trabajo de Titulación para optar al título de Ingeniero en Electrónica y Telecomunicaciones

Autor:
Paucar Asqui Andrés Paolo

Tutor:
PhD. Rentería Bustamante Leonardo Fabián

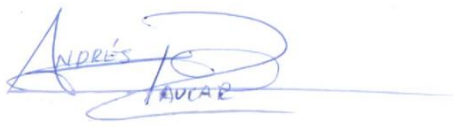
Riobamba, Ecuador.
Año 2022

1. DERECHOS DE AUTORÍA

Yo, Andrés Paolo Paucar Asqui, con cédula de ciudadanía 0603934233, autor (a) (s) del trabajo de investigación titulado: ANALISIS E IMPLEMENTACIÓN DE UN PROTOTIPO DE RED SDN EN EL CAMPUS NORTE DE LA UNACH, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 19 de mayo de 2022.



Andrés Paolo Paucar Asqui

C.I: 0603934233

2. DICTAMEN FAVORABLE DEL TUTOR;

Quien suscribe, PhD. Leonardo Fabián Rentería Bustamante, catedrático designado para la evaluación del trabajo de investigación ANALSIS E IMPLEMENTACION DE UN PROTOTIPO DE RED SDN EN EL CAMPUS NORTE DE LA UNACH por Andrés Paolo Paucar Asqui, con cédula de identidad número 060393423-3, certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha asesorado durante el desarrollo, revisado y evaluado el trabajo de investigación escrito y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 19 de mayo de 2022.

PhD. Leonardo Fabián Rentería Bustamante

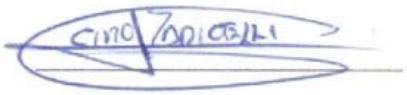
C.I: 1104064132

3. CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

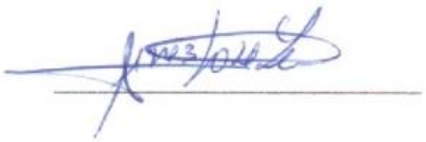
Quienes suscribimos, catedráticos designados, Miembros del Tribunal para la evaluación del trabajo de investigación ANALISIS E IMPLEMENTACION DE UN PROTOTIPO DE RED SDN EN EL CAMPUS NORTE DE LA UNACH, presentado por Andrés Paolo Paucar Asqui, con cédula de identidad número 060393423-3, bajo la tutoría de Dr. Leonardo Fabián Rentería Bustamante; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable, firmamos, en Riobamba 19 de mayo de 2022

Presidente del Tribunal de Grado
PhD. Ciro Diego Radicelli García



Miembro del Tribunal de Grado
Mgs. José Luis Jinez Tapia



Miembro del Tribunal de Grado
PhD. Luis Patricio Tello Oquendo

4. CERTIFICADO ANTIPLAGIO

C E R T I F I C A C I Ó N

Que, **PAUCAR ASQUI ANDRES PAOLO** con CC: **0603934233**, estudiante de la Carrera **ELECTRÓNICA Y TELECOMUNICACIONES,** Facultad de **INGENIERÍA**; ha trabajado bajo mi tutoría el trabajo de investigación titulado" **ANALISIS E IMPLEMENTACION DE UN PROTOTIPO DE RED SDN EN EL CAMPUS NORTE DE LA UNACH**", cumple con el 3 %, de acuerdo al reporte del sistema Anti plagio **Urkund**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 10 de mayo de 2022

PhD. Leonardo Fabián Rentería Bustamante.
TUTOR

5. DEDICATORIA

Le quiero dedicar a la persona que no pudo estar en mi caminar universitario, pero sé, que siempre estuvo a mi lado cuidándome y dándome las fuerzas para seguir adelante, "cuando estaba triste me decías que cerrara los ojos y pensara en cosas bonitas, hoy cierro los ojos y pienso en ti mamá", un día te dije que estarías orgullosa de mi, esto va por ti, Mamá.

Dedicarle al pedacito de cielo que tengo en la tierra, a mi hijo, Julián, que ha sido la inspiración terrenal para continuar con mis estudios hasta el final, que ha sido capaz de con un abrazo recargarme el corazón de amor para siempre seguir soñando.

Andrés P. Paucar A.

6. AGRADECIMIENTO

Primero le agradezco a Dios por darme salud y vida, para poder seguir cumpliendo mis sueños, agradezco a mi padre por luchar incansablemente, para que las metas de sus hijos se cumplan, y darme ese cariño de padre y de madre cuando ella se fue de este mundo. No te lo digo muchas veces pero "Gracias Papá".

A mi hermano que siempre ha estado a mi lado, apoyándome y consintiéndome, que este fruto de mi esfuerzo, no hubiera sido posible, sino estuvieras en mi vida, un día me diste un consejo que jamás olvidare y espero siempre aplicarlo en mi vida.

Al mejor amigo que la vida me pudo haber brindado, Cristian, gracias por todo, gracias por enseñarme, que por más oscura que este la noche siempre saldrá el sol, fuiste y serás un ejemplo de vida para mí, y estoy seguro que para muchos más

A mi familia y a mis queridos amigos, que son como mi familia, gracias por el apoyo tan grande que han sido en mi vida, espero poder brindar igual mi apoyo y seguir compartiendo momentos justos.

Andrés P. Paucar A.

7. ÍNDICE GENERAL

1. DERECHOS DE AUTORÍA 2
2. DICTAMEN FAVORABLE DEL TUTOR; 3
3. CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL 4
4. CERTIFICADO ANTIPLAGIO 5
5. DEDICATORIA 6
6. AGRADECIMIENTO 7
7. ÍNDICE GENERAL 8
8. ÍNDICE DE FIGURAS 10
9. RESUMEN 13
10. ABSTRACT 14
11. CAPÍTULO I. INTRODUCCIÓN. 15
    11.1 OBJETIVOS: GENERAL Y ESPECÍFICOS. 18
    11.2 PLANTEAMIENTO DEL PROBLEMA. 18
12. CAPÍTULO II. MARCO TEÓRICO. 20
    12.1 RED DE DATOS 20
    12.2 MODELO TCP/IP Y MODELO OSI 21
    12.3 RED SDN 22
        12.3.1 DEFINICION 22
        12.3.2 INTERFACES NORTHBOUND Y SOUTHBOUND 24
        12.3.3 COMPONENTES DE SDN 25
        12.3.4 OPENFLOW 26
    12.4 VIRTUALBOX 27
        12.4.1 INSTALACION. 28
    12.5 MININET 30
        12.5.1 DEFINNICION. 30
        12.5.2 CARACTERÍSTICAS. 30

12.5.3 INSTALACION. 31
12.5.4 COMANDOS DE MININET. 33
12.6 CONTROLADORES 41
12.6.1 OPENDAYLIGHT. 41
12.6.2 CONTROLADOR POX. 44
12.6.3 CONTROLADOR FLOODLIGHT 45
12.7 IMPLEMENTACION DEL PROTOTIPO DE RED SDN. 47
12.7.1 ANALSIS Y SELECCIÓN DEL CONTROLADOR SDN 47
12.7.2 DISEÑO EIMPLEMENTACION DE UN PROTOTIPO DE RED SDN PARA EL CAMPUS NORTE DE LA UNACH. 49
13. CAPÍTULO III. METODOLOGIA. 56
14. CAPÍTULO IV. RESULTADOS Y DISCUSIÓN 58
14.1 RESULTADOS DE WIRESHARK 58
14.2 PAQUETES ENVIADOS Y LATENCIA 59
14.3 DISCUSIÓN. 63
15. CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES 64
15.1 CONCLUSIONES. 64
15.2 RECOMENDACIONES. 65
16. BIBLIOGRAFIA 66
17. ANEXOS 67

8. ÍNDICE DE FIGURAS

Figura 1. Arquitectura de red actual. 16
Figura 2. Arquitectura de red SDN 17
Figura 3. Modelo TCP/IP y OSI 22
Figura 5. Arquitectura SDN 23
Figura 6. Red Tradicional / Red SDN. 24
Figura 7. Interfaces Northbound y Southbound 25
Figura 8. Componentes SDN. 26
Figura 9. Tabla de Flujo. 27
Figura 10. Instalación VirtualBox. 28
Figura 11. Ejecutar VirtualBox. 29
Figura 12. Instalación en proceso. 29
Figura 13. Instalación finalizada. 29
Figura 14. Simulación / Implementación 30
Figura 15. Ingreso al súper usuario. 32
Figura 16. Comando apt-get update 32
Figura 17. Comando git clone http://github.com/mininet/mininet 32
Figura 18. Selección de la versión de mininet. 32
Figura 19. Instalación final de mininet 33
Figura 20. Verificación de instalación. 33
Figura 21. Topología minimal. 34
Figura 22. Topología single. 35
Figura 23. Topología linear de n=4. 35
Figura 24. Topología Tree con n=3 36
Figura 25. Script de Python. 36
Figura 26. Ingresar miniedit. 37
Figura 27. Miniedit. 38
Figura 28. Herramienta de selección. 38

Figura 29. Herramienta de Host.....38
Figura 30. Herramienta de conmutador.....39
Figura 31. Herramienta de conmutador tradicional.....39
Figura 32. Herramienta de router tradicional.....39
Figura 33. Herramienta de enlaces.....39
Figura 34. Herramienta de controlador.....40
Figura 35. Topología creada en miniedit.....40
Figura 36. Guardar topología de miniedit.....41
Figura 37. Instalación de jdk.....42
Figura 38. Instalación de la variable global.....42
Figura 39. Extracción de opendaylight.....43
Figura 40. Desempaquetado de opendaylight.....43
Figura 41. Opendaylight en funcionamiento.....43
Figura 42. Instalación de controlador pox.....44
Figura 43. Controlador Pox funcionando.....45
Figura 44. Página de descarga de flooding.....46
Figura 45. Descarga del archivo.....46
Figura 46. Máquina virtual flooding.....46
Figura 47. Flooding en funcionamiento.....47
Figura 48. Wireshark.....48
Figura 49. Envió de paquetes.....48
Figura 50. Ancho de banda.....48
Figura 51. Envió de paquetes Pox.....48
Figura 52. Ancho de banda en controlador Pox.....49
Figura 53. Envió de trafico Flooding.....49
Figura 54. Ancho de banda Flooding.....49
Figura 55. Diseño de prototipo de red SDN.....50
Figura 56. Inicialización de Opendaylight.....51
Figura 57. Resetear Mikrotik.....52

Figura 58. Ejecutar Winbox. 53
Figura 59. Herramienta Openflow en Mikrotik. 53
Figura 60. Asignación de puerto para controlador. 54
Figura 61. Implementación de puertos openflow. 55
Figura 62. Implementación Visualizada en el controlador ODL 56
Figura 63. Grafica del controlador ODL. 58
Figura 64. Grafica del controlador Flooding. 59
Figura 65. Grafica del controlador Pox. 59

9. RESUMEN

El surgimiento de nuevas tecnologías conlleva una gran demanda de tráfico de datos a nivel global, por lo que las arquitecturas actuales de red deben ser escalables, dinámicas y flexibles, esto impulsa a las empresas que ofertan servicio de internet a innovar y analizar sus arquitecturas de red tradicionales En el presente proyecto se realizó el análisis y diseño de un prototipo de red SDN (Software defined networking), con el cual se pretende adaptar una nueva arquitectura de red, la gestión de las redes SDN se realiza a través de 3 capas principales, capa de aplicación, capa de control y capa de infraestructura. La capa de aplicación o aplicaciones, es donde residen los programas encargados de comunicar el comportamiento deseado de la red al controlador SDN. La capa de control se encarga de configurar los diferentes routers y switch de la red, decide el camino que debe de tomar los paquetes para llegar a su destino, ya que gestiona la capa de aplicación y la capa de infraestructura usando interfaces. La capa de infraestructura es donde se localizan los switch interconectados que pertenecen a la red SDN. Se efectuó un análisis de los diferentes controladores SDN Open Source, mediante un software de simulación instalado en una máquina virtual, primero se debe seleccionar los controladores que van a ser analizados, en este proyecto se seleccionó los controladores, OPENDAYLIGHT, FLOODLIGHT Y POX, en una computadora se procedió a instalar el software Virtual Box, que es necesario para cargar una máquina virtual con el sistema operativo Ubuntu, una vez instalado el sistema operativo Ubuntu, instalamos la herramienta de simulación de red SDN, llamada Mininet, adicionalmente en la máquina virtual que se creó se instalan los controladores para posteriormente ser vinculados con Mininet, se crea una red y se implementan los controladores para realizar el análisis de los datos, El análisis de tráfico se realizó con el software Wireshark que permite ver los paquetes enviados y recibidos en tiempo real. Para la implementación del prototipo de red se seleccionó el controlador Opendaylight, los switchs utilizados fueron de la marca Mikrotik y para las conexiones se utilizó cables de red UTP.

Palabras claves: SDN, Mininet, Openflow, Opendaylight, Floodlight, Pox

10. ABSTRACT

The emergence of new technologies leads to a high demand for data traffic globally, so the current network architectures must be scalable, dynamic and flexible, this drives companies that offer internet service to innovate and analyse their traditional network architectures. In this project the analysis and design of a prototype SDN (Software defined networking) network was carried out, with which it is intended to adapt a new network architecture, the management of SDN networks is done through 3 main layers, application layer, control layer and infrastructure layer. The application layer is where the programs in charge of communicating the desired behavior of the network to the SDN controller reside. The control layer is in charge of configuring the different routers and switches of the network, it decides the path that the packets must take to reach their destination, as it manages the application layer and the infrastructure layer using interfaces. The infrastructure layer is where the interconnected switches belonging to the SDN network are located. An analysis of the different Open Source SDN controllers was carried out, using simulation software installed in a virtual machine, first the controllers to be analyzed must be selected, in this project the controllers OPENDAYLIGHT, FLOODLIGHT and POX were selected, in a computer the Virtual Box software was installed, which is necessary to load a virtual machine with the Ubuntu operating system, Once the Ubuntu operating system was installed, we installed the SDN network simulation tool, called Mininet, additionally in the virtual machine that was created the controllers were installed to later be linked with Mininet, a network was created and the controllers were implemented to carry out the data analysis. The traffic analysis was carried out with the Wireshark software that allows us to see the packets sent and received in real time. For the implementation of the network prototype, the Opendaylight controller was selected, Mikrotik switches were used and UTP network cables were used for the connections.

Keywords: SDN, Mininet, Openflow, Opendaylight, Floodkight, Pox.

Reviewed by:

Firmado electrónicamente por:
ANDREA
CRISTINA
RIVERA PUGLLA

Lic. Andrea Rivera

ENGLISH PROFESSOR

C.C 0604464008

11. CAPÍTULO I. INTRODUCCIÓN.

A lo largo de los años, el aumento de dispositivos móviles, la nube, servidores virtuales y la demanda masiva de tráfico, hace que las tecnologías avancen a pasos agigantados, teniendo que acomodarse al beneficio de usuarios y empresas. Este aumento, a la vez, impulsa a la industria de las Tecnologías de la Información y de las Comunicaciones (TICs) a mejorar con innovaciones y reexaminar las tecnologías tradicionales a nivel internacional. [1]

En la infraestructura de red actual existen 2 planos importantes, un plano de control y un plano de datos, que se encuentran juntos en cada dispositivo de red. La complejidad y dificultades radican no solo en la topología, sino también en la variedad de protocolos (MPLS, EIGRP, OSPF), que deben coexistir para asegurar la comunicación e interoperabilidad entre ellos, así como una gestión eficiente. Todo eso con lleva unos equipos más robustos de los que serían implementados en una red definida por software. [2]

Redes Definidas por Software (SDN, por sus siglas en inglés), surgen como una opción para las nuevas arquitecturas de red; una definición aceptable de las redes SDN, es, separar el plano de datos y el plano de control, para ser todo controlado por medio de una aplicación que se denomina controlador. Además, las redes no tienen manipulación por hardware, esto quiere decir, que ya no se debe configurar cada dispositivo y todo pasa a ser controlado por una aplicación, ya sea en un ordenador o en algún dispositivo móvil. Gracias a ello se soluciona los problemas de las redes actuales, dado que, el plano de control está situado en un servidor central que se encargará de configurar y gestionar todos los equipos de la infraestructura de red, ahorrando en personal costoso y sobre todo en tiempo de configuración de una red. [2]

En la actualidad, las redes SDN, están en crecimiento por sus amplias ventajas con respecto a las demás arquitecturas, muchas empresas, principalmente las que se encuentran en evolución, buscan mejorar su oferta de negocios, implementando nuevos servicios; no obstante, operadoras de gran 4 prestigio como son GOOGLE, FACEBOOK,

YAHOO!, están migrando sus redes actuales, a redes definidas por software, dado que afrontan cada día, miles de nuevos usuarios solicitando sus servicios de internet. Un gran problema con el que estas operadoras deben lidiar es la escalabilidad de sus redes. Cuando un usuario va a realizar una búsqueda de información en el portal de Google, el intercambio de datos entre los nodos de esta red, puede llegar a los 1000 Terabytes. Aquí radica la importancia de implementar las redes SDN, que brindarán mayor escalabilidad, proporcionando un eficiente rendimiento y mejorando la conectividad entre los cientos de miles de usuarios. [3]

El escenario de una red actual, como se ilustra en la Figura 1, observamos como existen varios protocolos de capa 2 y capa 3 en la misma red, lo que ocasiona que se debe tener equipos que puedan soportar todos estos protocolos, y el plano de control como el plano de datos se encuentran en cada dispositivo de esta red.

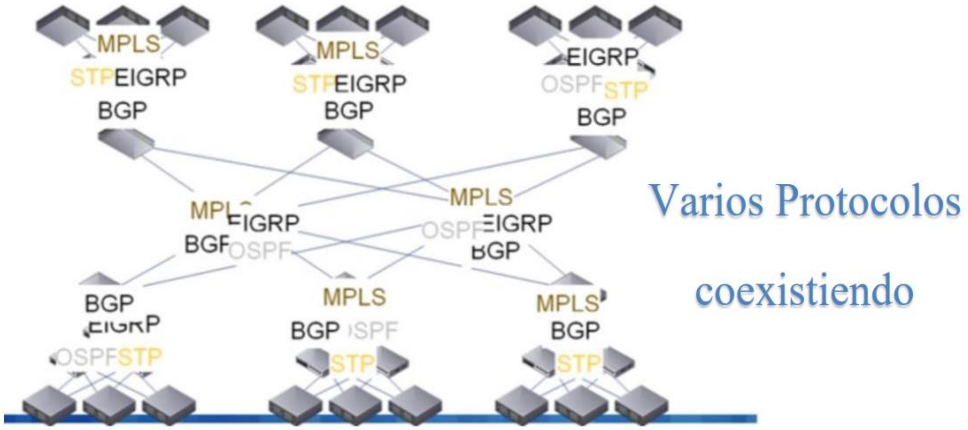


Figura 1. Arquitectura de red actual.

En el escenario de una red SDN, como se ilustra en la Figura 2, observamos como en esta red, el plano de control y plano de datos están separados, para conseguir un control centralizado mediante un controlador SDN, con ello se consigue abaratar costos de equipo, personal y tiempo.

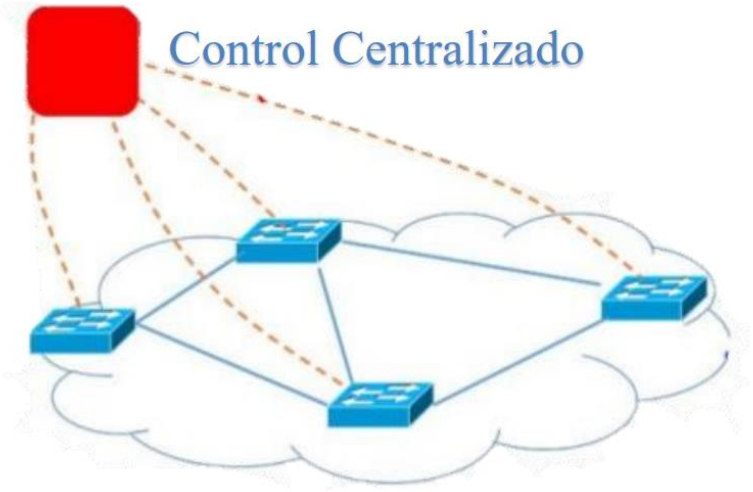


Figura 2. Arquitectura de red SDN

La gestión de las redes SDN se realizará, a través de la virtualización de red, independizándola de la infraestructura física subyacente y creando redes lógicas. Para ello se utilizan protocolos existentes, por ejemplo: OpenFlow, Border Gateway Protocol (BGP), NETCONF, protocolos de Gestión de Base de Datos Open vSwitch (OVSDB), Protocolo de Presencia y Mensajería Extensible (XMPP). Actualmente, la especificación más popular para crear una red SDN, es el estándar de código abierto OpenFlow [4]

OpenFlow es uno de los primeros estándares para una red SDN, constituyen la base en el área de las redes abiertas definidas por software. En el año 2007 se desarrolló este protocolo con la ayuda de las universidades de Stanford y California en primera instancia, en la actualidad se tiene la versión 1.5, que fue creada en 2014. Los conmutadores o Switches con protocolo OpenFlow deben tener 3 características principales para su funcionamiento, entre ellos se encuentra la tabla de flujos, asocia a cada entrada de la tabla e indica al conmutador como debe de procesar esa información, el canal seguro, permite el envío de comandos y paquetes entre el conmutador y el controlador usando el protocolo OpenFlow, y el controlador, añade y elimina entradas en las tablas de flujo.

11.1 OBJETIVOS: GENERAL Y ESPECÍFICOS.

OBJETIVO GENERAL.

- Evaluar controladores Opensource de red SDN y seleccionar el más adecuado para implementar un prototipo de red SDN para el Campus Norte de la Universidad Nacional de Chimborazo (UNACH).

OBJETIVOS ESPECIFICOS.

- Analizar el rendimiento de 3 controladores SDN Opensource, en una infraestructura que simule el tráfico de una red.
- Diseñar una red SDN utilizando el controlador más adecuado, según el análisis previo en el campus norte de la UNACH.
- Implementar un prototipo de red SDN en el campus norte de la UNACH, utilizando un controlador Opensource.

11.2 PLANTEAMIENTO DEL PROBLEMA.

Las arquitecturas de red en la actualidad no son flexibles y la complejidad es alta, dado que se basa en la transmisión de datos, minimizando agilidad, flexibilidad y escalabilidad. La gestión es poco automatizada, y para poder hacer la configuración de un dispositivo, se debe hacer manualmente y a su vez puede producirse más errores, por error de quien realiza la configuración, aparte los servicios de aplicaciones han ido incrementando considerablemente. Pero la arquitectura que soporta dichas aplicaciones no ha evolucionado con la misma rapidez, convirtiéndose en una limitante. El surgimiento de nuevas aplicaciones y las industrias que impulsan el uso de las mismas, ya sea para dispositivos móviles, ordenadores portátiles, aplicaciones web, para beneficio de negocios, ocio, agencias bancarias, videoconferencias o videojuegos en línea, demanda una gran cantidad de información. Es por esto que surge la necesidad de permitir que la transmisión de datos, sea de una forma ágil y contar con elementos cuya infraestructura sea lo suficientemente idónea para la administración de los datos de una forma eficiente. [5]

Los controladores SDN, poseen una visión completa de la red y pueden programarse para tomar decisiones óptimas; equivale al sistema operativo de la red que controla todas las comunicaciones entre las aplicaciones y los dispositivos. El controlador se encarga de traducir las necesidades o requisitos de la capa aplicación a los elementos de la red, y de proporcionar información relevante a las aplicaciones SDN, pudiendo incluir estadísticas y eventos.

En estos controladores se encuentra el plano de control, que está encargado de varias funciones principales de la red, como el enrutamiento de rutas, topología, estado y comportamiento de las redes adyacentes, optimiza la configuración del sistema de administración y gestiona el intercambio de paquetes, este plano, indica a la capa de control la dirección de cada uno de los paquetes que entran y la dirección que debe tomar para salir. En los demás dispositivos de la red se encuentra el plano de datos, es mucho más sencillo, su función radica en procesar los datos en función a la trama de enrutamiento.

En cuanto a los tipos de controladores SDN, en el mercado existen varios controladores, tanto Opensource como propietarios. En este proyecto se trabajará con controladores Opensource, ya que son de uso libre y no se tiene que pagar por ello. La principal diferencia de los controladores que usaremos en las simulaciones es, el lenguaje de programación que se emplea, para definir las reglas, políticas y plataforma en la que trabajan, puesto que la funcionalidad es la misma. Todos utilizan el estándar OpenFlow para comunicarse con los demás equipos de red SDN. Al existir varios controladores se debe analizar diferentes parámetros para cada uno de ellos, como virtualización de red, funcionalidad, escalabilidad, rendimiento, programación de red, confiabilidad, seguridad de red, monitorización centralizada y procesamiento. Esto permitirá desarrollar nuevas alternativas para disminuir costos, optimizar los recursos, agilizar los procesos, además de lograr un análisis comparativo de los diferentes controladores, orientado a la infraestructura de red que se encuentra implementada en la UNACH. [6]

La importancia de este proyecto consiste en evaluar la flexibilidad de la red con un sistema de control, donde se podrá cambiar el comportamiento de la red de manera

dinámica y automatizada, obteniendo una red adaptable. Además de aprovechar la utilización de software libre. El objetivo es seleccionar el controlador que mejor se ajuste a los requerimientos y tráfico que se intercambia. Seleccionado el controlador, se iniciará el diseño de un prototipo de red SDN y se presentará una propuesta, para una futura implementación, tomando en cuenta el costo de implementación de una red SDN.

La evaluación de la red, se desarrollará en un ambiente virtual, que permita emular la propuesta basada en la tecnología SDN, se usará simuladores para el escenario de la red y herramientas que genere flujos de datos UDP y/o TCP para medir el rendimiento, además se usará funciones de los simuladores que nos permitan, visualizar, analizar y comparar las características entre una red tradicional actual y una red SDN.

12. CAPÍTULO II. MARCO TEÓRICO.

12.1 RED DE DATOS

Con la evolución de la civilización humana se ha avanzado en la forma de comunicación entre personas, sea de forma escrita, oral, o por señas, y en las últimas décadas de forma digital, llegando a conectar personas que se encuentran a miles de kilómetros de distancia de manera casi instantánea. La industria ha promovido el uso de masivo de internet, empresas como, Facebook, YouTube, Netflix, tienen una gran cantidad de usuarios, lo que con lleva, una gran demanda de datos, una gran demanda de equipos que soporten el flujo de información y una arquitectura que permita el envío y recepción de datos de manera óptima.

Los equipos, los medios físicos y lógicos que realizan la comunicación de información entre dispositivos o usuarios a cualquier distancia se denomina red de transmisión de datos, entre las principales redes se encuentran, en el ámbito local LAN y en el ámbito global WAN. [1]

Características principales

Métodos de conmutación de datos

- **Conmutación por circuitos:** Transmisión de datos orientada a la conexión, puesto que la sesión no puede ser interrumpida, si la sesión es interrumpida la comunicación expirara. Es poco flexible y poco escalable. Un ejemplo claro es la red telefónica tradicional.

  Características.

  - Debe tener una conexión de extremo a extremo antes de transmitir la señal.
  - Los datos pueden ser analógico o digital dependiendo de la red, la conexión es full dúplex normalmente.
  - Para la desconexión del circuito, se debe propagar las señales correspondientes a los nodos con los que se estableció la conexión, para que liberen los recursos que se utilizaron.

- Conmutación por paquetes: La transmisión de datos es No orientada a la conexión, los datos enviados son segmentados en paquetes más pequeños, por lo que a cada paquete se le incluye una cabecera con la información necesaria para llegar a su destino, además los paquetes pueden emplear distintos caminos disponibles en la red.

12.2 MODELO TCP/IP Y MODELO OSI

Comprender la conmutación de circuitos y paquetes, es importante para entender cómo se transmiten los datos, otro aspecto importante son los modelos de referencia: TCP/IP y OSI.

TCP/IP es un protocolo que va dirigido a la información transferida a través de internet. Permite la comunicación entre equipos, sin importar si los equipos son de diferentes sistemas operativos, sean de área local (LAN) o de área extensa (WAN),

el modelo se compone de 4 capas, las cuales son: capa aplicación, capa transporte, capa internet y acceso a la red. [7]

OSI sirve como fundamento teórico para la interconexión de sistemas abiertos, está compuesto por 7 capas, en la que los datos se empaquetan y trasmiten desde una aplicación emisora hacia otra receptora, el nombre de las capas son: capa de aplicación, capa de sesión, capa de transporte, capa de red, capa de enlace de datos y capa física. [7]

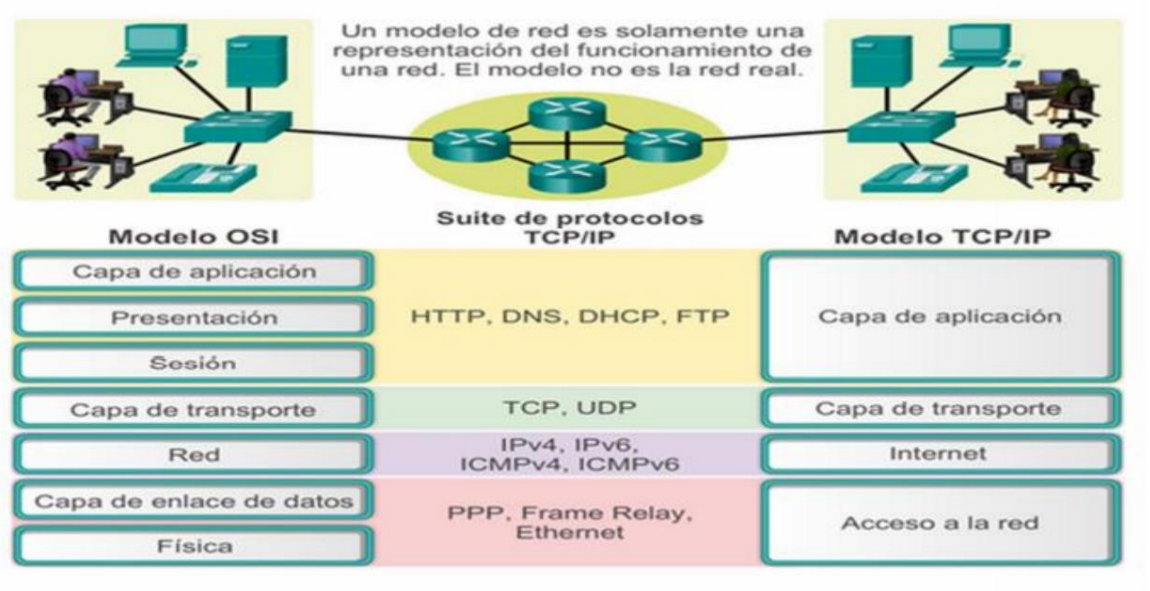


Figura 3. Modelo TCP/IP y OSI

12.3 RED SDN.

12.3.1 DEFINICION

Es una arquitectura diseñada para abastecer el ritmo de crecimiento de las aplicaciones en la actualidad, su funcionamiento se basa en separar la administración de la red de la infraestructura de red subyacente, para así conseguir un flujo de tráfico más dinámico, con ello reducir la complejidad de las redes definidas estáticamente, automatice las funciones de la red y simplifiquen la implementación.

La red SDN principalmente se compone de 3 capas, en las que se encuentra la capa de aplicaciones, la capa de control y la capa de infraestructura, conectados con unas API de comunicación. La capa de aplicación incluye aplicaciones y funciones de red, la capa de control, administra el flujo de tráfico de la red y la capa infraestructura contiene los switch físicos de la red. [8]

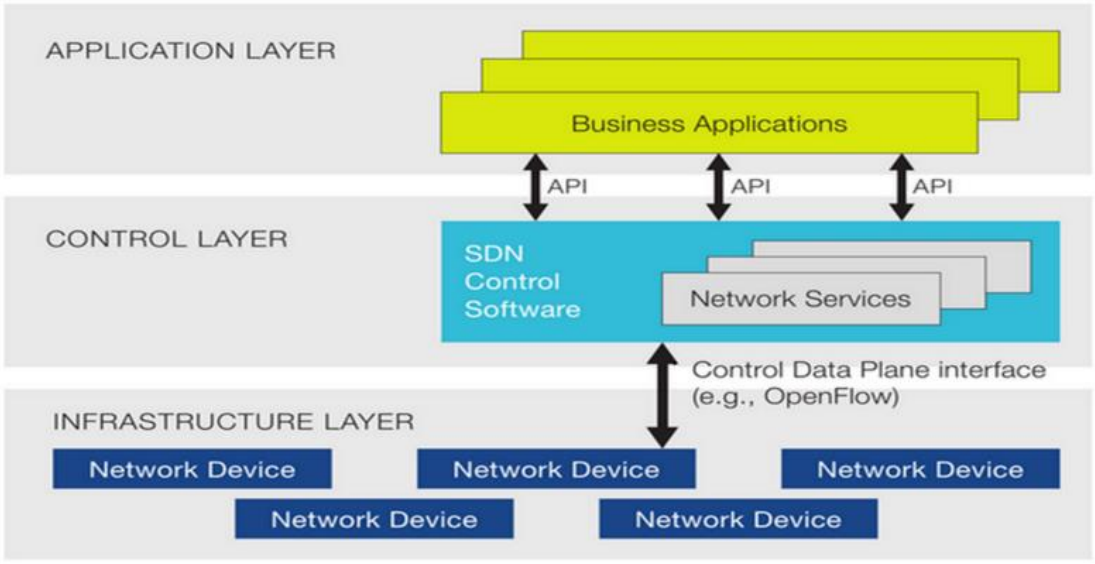


Figura 5. Arquitectura SDN

El plano de control está separado del plano de datos en los equipos de red, es la principal diferencia entre la red tradicional y una red SDN además de añadir el controlador que es el elemento que administra a los conmutadores, y gestiona el tráfico que por cada uno de ellos pasa, se podría decir que el controlador es el cerebro de la red SDN.

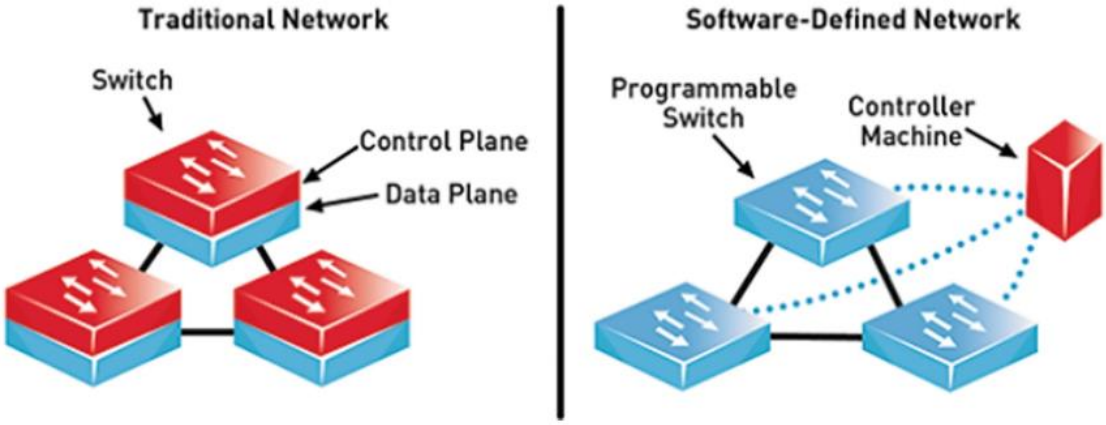


Figura 6. Red Tradicional / Red SDN.

12.3.2 INTERFACES NORTHBOUND Y SOUTHBOUND

Las interfaces Northbound (API) son las aplicaciones que permite ver a los equipos interconectados en la red y permite la comunicación con el controlador, la desventaja principal de este protocolo, es precisamente el protocolo, ya que no existe una estandarización y cada fabricante selecciona su API, lo que conlleva a que se debe conocer el funcionamiento de las APIs del fabricante con el que se va a operar. [9]

Las interfaces Southbound son los protocolos que se encarga de comunicar el controlador con el plano de datos de los schitch que componen la red. El principal protocolo opensource y el que se empleó en este proyecto fue Openflow, dado que es un protocolo libre y no necesita ninguna licencia para su implementación. [9]

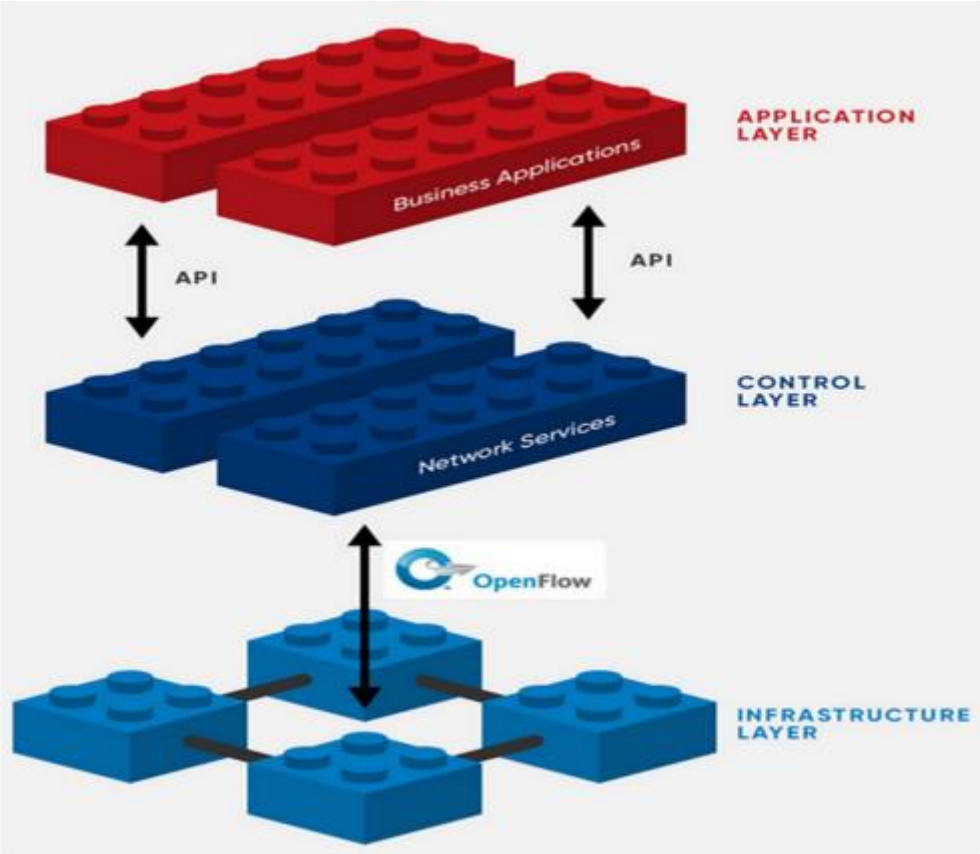


Figura 7. Interfaces Northbound y Southbound

12.3.3 COMPONENTES DE SDN

En este apartado se despliega una lista de los principales componentes de una infraestructura de red SDN.

- **Controlador SDN,** componente principal que gestiona y comunica los dispositivos interconectados de una red, establece las políticas de enrutamiento y en el que se encuentra el plano de control
- **Dispositivos y equipos de red**, pueden ser físicos o virtuales, están situados en el plano de datos de una SDN
- **Interfaces Northbound y Southbound**, son las APIs y el protocolo para comunicar los equipos de red entre ellos.
- **NOS (Network Operating System),** es el sistema operativo que se va a usar en la red, ya puede ser de código abierto o de propietario.

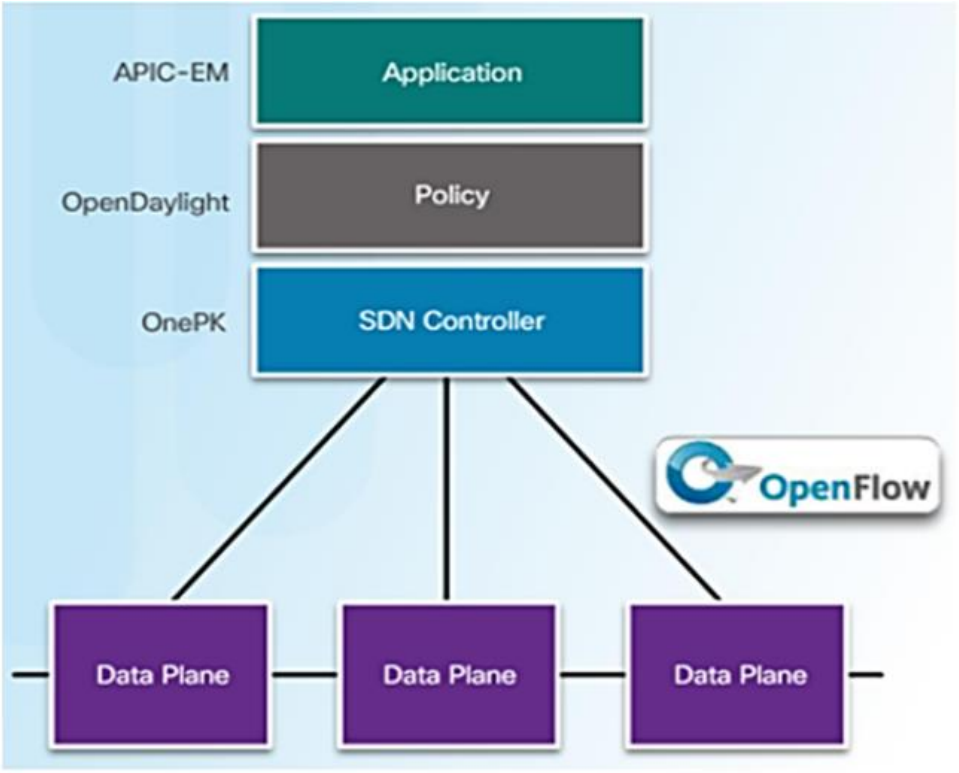


Figura 8. Componentes SDN.

12.3.4 OPENFLOW

Opoenflow es un protocolo de código abierto y su función principal es la interconexión de los equipos, en la arquitectura tradicional, el plano de control y el plano de datos se encuentran en el mismo equipo (switch), por lo que cada equipo tiene sus propias tablas de enrutamiento con las que seleccionan por donde de ir cada paquete de datos, en cambio, con el protocolo openflow, el controlador es quien decide por donde enviar el tráfico dado que se encuentra en el plano de control, mientras que los switch openflow se encuentran en el plano de datos y su función es enviar el tráfico por donde le guie el controlador.

Openflow no es el único protocolo que tiene SDN, existen varios otros como por ejemplo, Border Gateway Protocol (BGP), Protocolo de Gestión de Base de Datos Open vSwitch (OVSDB) entre otros, pero en este trabajo de

investigación se trabajó con openflow, por ser de código abierto, no se necesita ninguna licencia para su uso y el software que utilizaremos para la simulación de la red SDN también admite este protocolo en sus herramientas. [10]

12.3.4.1 CARACTERISTICAS.

Openflow permite la comunicación entre el controlador y los switchs, así como el intercambio de información de los paquetes enviados y los recibidos, la tabla de flujo es la que recoge las acciones que se debe realizar con el tráfico que llega a los switchs.

El canal seguro se conecta con el controlador para poder enviar comandos y paquetes a través del protocolo openflow, y a su vez el protocolo provee un estándar de comunicación entre el conmutador y el controlador, y así ser capaz de modificar, eliminar e insertar entradas de nuevas tablas de flujos.

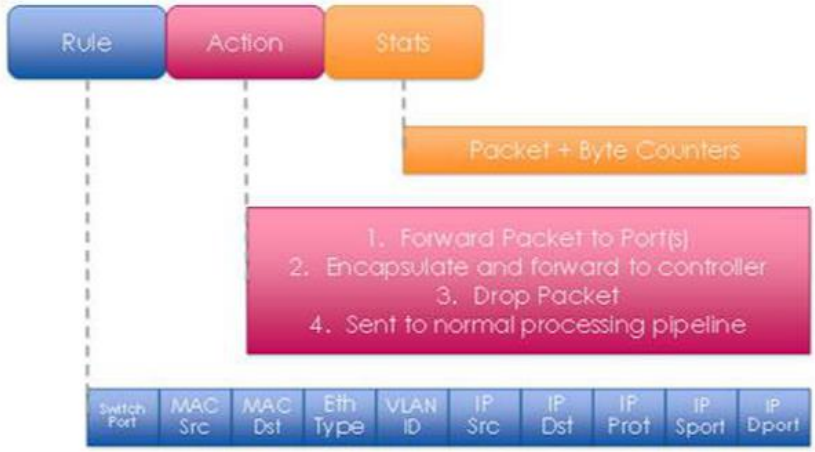


Figura 9. Tabla de Flujo.

12.4 VIRTUALBOX

Es un software libre que permite la instalación de varios sistemas operativos dentro de otro, es decir, se puede instalar el sistema operativo Linux dentro de Windows o incluso en MacOS, su principal funcionalidad es, no desinstalar el sistema operativo

actual de una computadora para instalarle uno nuevo, ya que con ello se perdería la información o se tendría que comprar otra computadora para instalar el sistema operativo que se desee.

12.4.1 INSTALACION.

Para la instalación de virtualbox se debe ir a la página oficial, y seguir los siguientes pasos.

- Enlace de la página oficial: https://www.virtualbox.org/
- Descargar el archivo dependiendo el sistema operativo donde se vaya a instalar virtualbox, en este proyecto se descargó virtualbox para Windows.
- Una vez descargado el archivo lo ejecutamos como administrador.

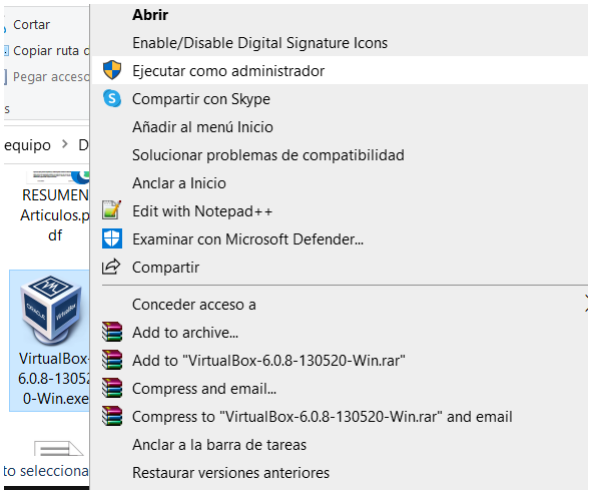


Figura 10. Instalación VirtualBox

- Al iniciar se abre una pantalla donde se pondrá siguiente.

Figura 11. Ejecutar VirtualBox.

- Para continuar con la instalación se seleccionara "next" a las siguientes pantallas emergentes, hasta que se instale por completo e iniciemos virtualbox.

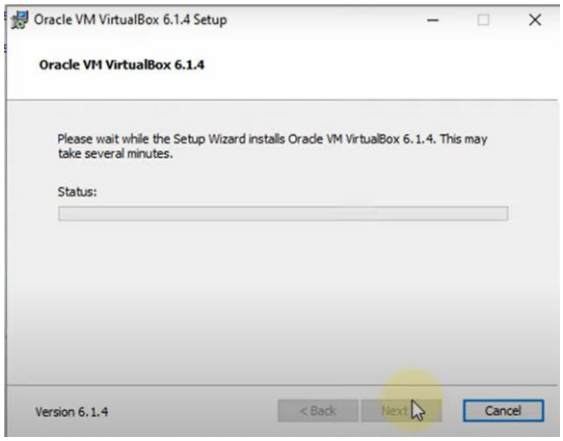


Figura 12. Instalación en proceso.

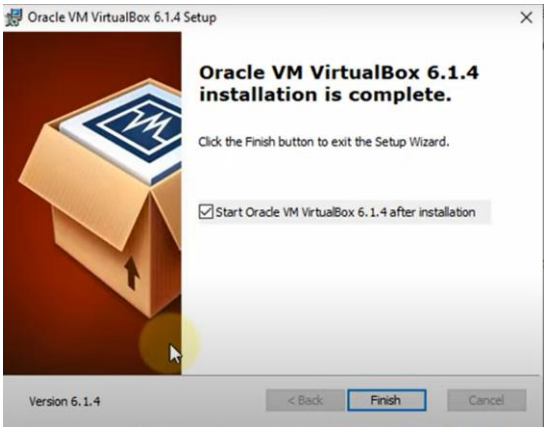


Figura 13. Instalación finalizada.

12.5 MININET

12.5.1 DEFINNICION.

Mininet es un software para la emulación de una red SDN, en sus herramientas contiene hots, switch, controlador, además de tener el protocolo openflow que es el cual se utilizó en la investigación, mininet se instaló en un sistema operativo Linux, en concreto en el sistema operativo Ubunto 20.4, instalado en virtualbox creando una máquina virtual.

Mininet fue uno de los primeros emuladores para SDN, permite ejecutar la simulación de redes virtuales, con todos los dispositivos de red, incluyendo hots, switchs y controlador, y siendo capaz de generar tráfico para reflejar el comportamiento de la SDN. [11]

12.5.2 CARACTERÍSTICAS.

A día de hoy, es fundamental primero simular el proyecto que se quiere implementar para poder llevarlo a cabo, con más seguridad de que funcione y no existan errores al momento final de la ejecución. [3]

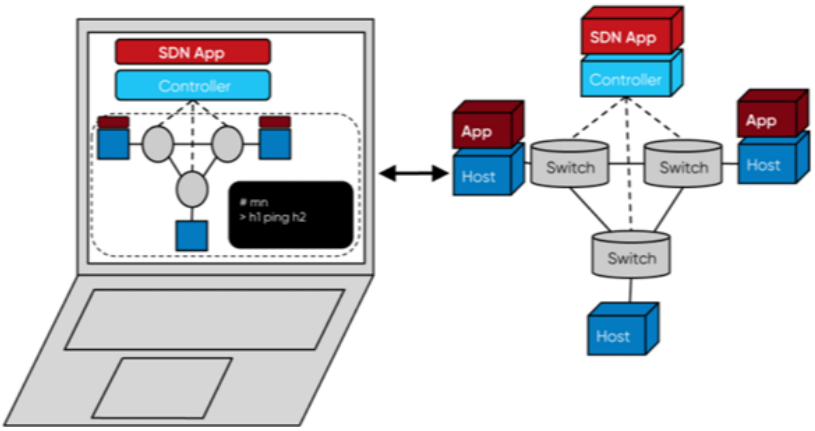


Figura 14. Simulación / Implementación

- Mininet es un emulador basado en Linux y permite crear redes SDN, además de incluir una interfaz gráfica para poder visualizar la red que se desea crear.
- Software opensource, por lo que no se debe pagar para su instalación ni su uso, además de ser la principal ventaja por lo que se usó para el desarrollo de esta investigación.
- Permite la conexión con otro software externos, como Wireshark que permite la visualización de tráfico.
- Permite la conexión con los controladores que se seleccionó para el análisis al ser implementados en una SDN.
- Permite la emulación en Python scripts para la creación de nuevas topologías para ser implementadas y simuladas en mininet.

12.5.3 INSTALACION.

Para la instalación de mininet existen varios métodos que se pueden utilizar, entre ellos están, instalación mediante una imagen en virtualbox, otra forma es desde la fuente instalación nativa, otra forma es la instalación mediante paquetes y por último esta la instalación de la actualización de mininet que se encuentra ya instalado en la computadora. Se deja el link de la página de instalación de mininet http://mininet.org/download/

Para la instalación de mininet en el proyecto de investigación se hizo de la segunda manera de instalación, Native installation from source.

- Primero se accede al terminal de Ubuntu usando el comando Ctrl + Alt + T. y se ingresa al superusuario usando el comando “sudo su” y se introduce la contraseña.

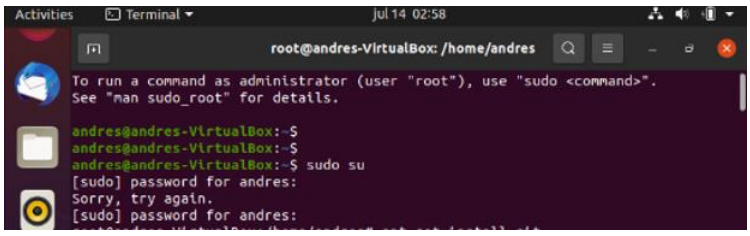


Figura 15. Ingreso al súper usuario.

- A continuación usamos el comando “apt-get update”.

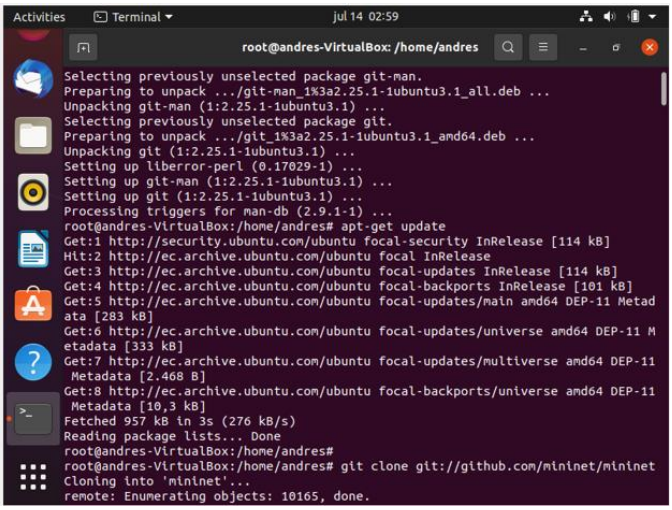


Figura 16. Comando apt-get update

- Se procede a ingresar el comando “git clone http://github.com/mininet/mininet” para poder clonar los archivos para ser instalados.

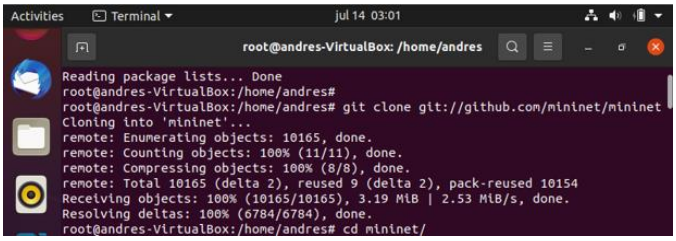


Figura 17. Comando git clone http://github.com/mininet/mininet

- Se usa el comando “git tag” para verficar la versión que se va a instalar de mininet.

Figura 18. Selección de la versión de mininet.

- En este proyecto se utilizó la versión 2.3.0 de mininet y se procedió a su instalación final con el comando “git checkout –b mininet-2.3.0 2.3.0”

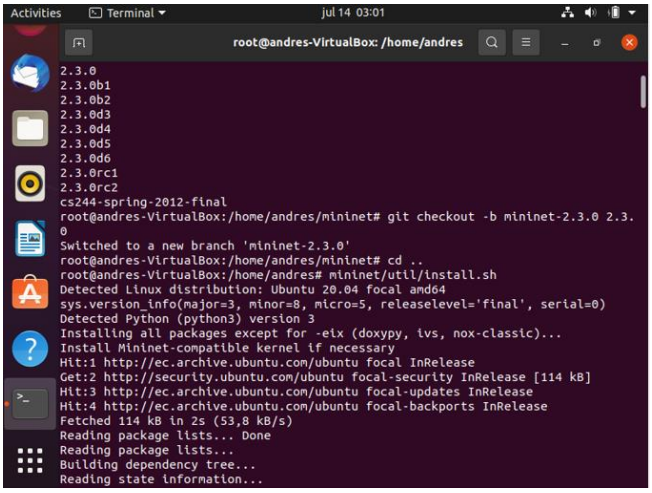


Figura 19. Instalación final de mininet

- Una vez finalizada su instalación se revisa que mininet este corriendo con normalidad creando una topología por defecto al usar el comando “sudo mn –switch ovsbr –test pingall”

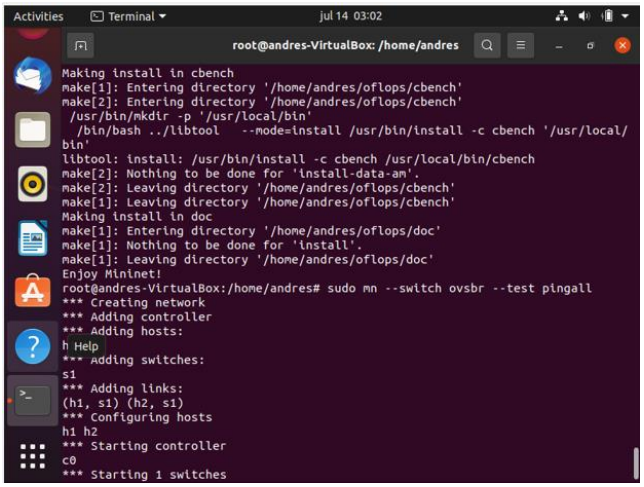


Figura 20. Verificación de instalación.

Una vez comprobada la instalación de mininet ya se puede realizar topologías y simulaciones de red SDN, pero al no haber un controlador, el software utiliza un controlador creado por defecto.

12.5.4 COMANDOS DE MININET.

12.5.4.1 TOPOLOGIA EN MININET.

Para empezar a ejecutar simulaciones en miniet, lo primero que se necesita saber son los comandos que se deben usar para crear una topología, hay diferentes formas de crear topologías, la primera es mediante los comandos por defecto que vienen incluidas en mininet. La siguiente es a través de API de Python y la otra alternativa es mediante la interfaz gráfica llamada miniedit. [12]

12.5.4.1.1 COMANDOS POR DEFECTOS

A continuación veremos los comandos predeterminados en mininet.

Topología Minimal, esta topología crea un switch openflow y dos hosts a él conectados, el comando que se utiliza es "mn –topo minimal"

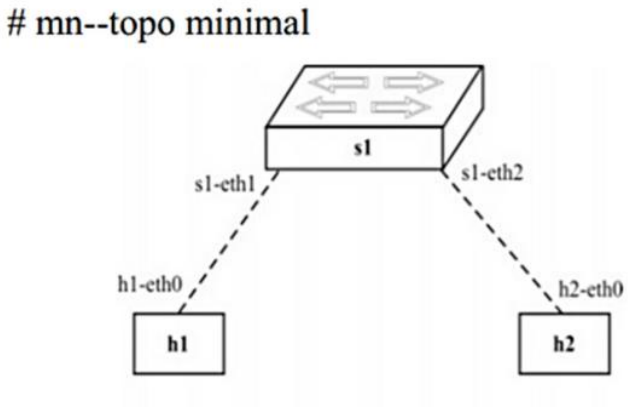


Figura 21. Topología minimal.

Topología Single, en esta topología es parecida a la minimal, pero en esta se puede incluir más cantidades de hots en el switch openflow y el comando que utilizamos es "mn –topo single, 4".

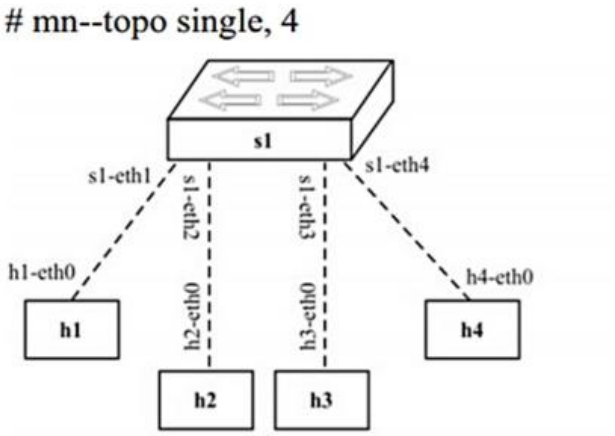


Figura 22. Topología single.

Topología Linear, en esta topología se puede crear más cantidades de switchs y más cantidades de hots conectados, el comando que se utiliza es "mn –topo linear,n" donde n es la cantidad de swirch, que se puede conectar en la red.

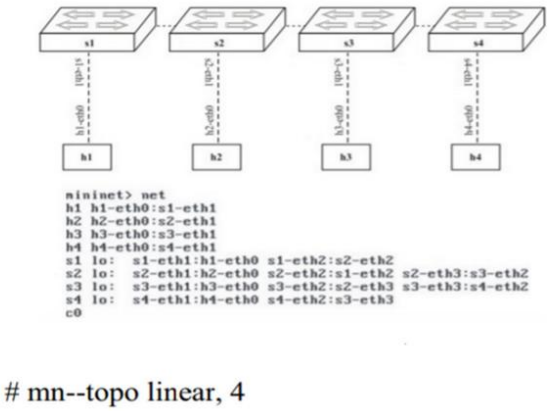


Figura 23. Topología linear de n=4.

Topología Tree, esta topología es de árbol y su comando es " mn –topo tree,n" donde n es la cantidad de niveles que se desea tener.

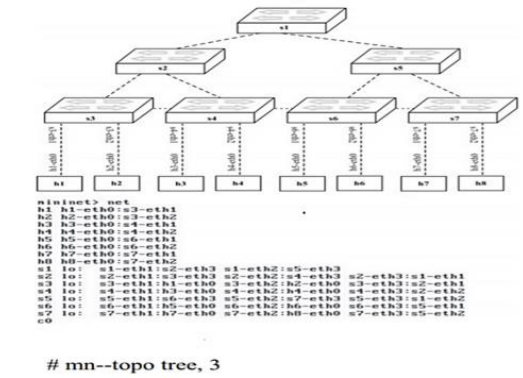


Figura 24. Topología Tree con n=3

12.5.4.2 API DE PYTHON

Ahora si desea crear topologías con la API de Python, se debe crear el script para programar la topología que deseamos.

nivelalto.py

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.topo import Topo
from mininet.log import setLogLevel, info
from mininet.util import quietRun

class PruebaTopo( Topo ):
    "Topologia de Prueba Alto Nivel"

    def build(self):
        self.addHost("h1")
        self.addHost("h2")
        self.addSwitch("s1")
        self.addSwitch("s2")
        self.addLink("h1","s1")
        self.addLink("h2","s2")
        self.addLink("s1","s2")

net = Mininet(topo=PruebaTopo())
net.start()
net.pingAll()
net.stop()

```

Python Tab Width: 8 Ln 25, Col 1 INS

Figura 25. Script de Python.

En este apartado se debe crear absolutamente toda la topología, desde las conexiones que deben hacer los hots, como los switch openflow, también se debe asignar el nombre de los equipos y las direcciones ip que se van a usar, los comandos son:

- **Topo,** como constructor, indica una topología a crear en mininet.
- **AddHost,** sirve para agregar los hots en la topología
- **addLink,** sirve para agregar los enlaces entre los nodos de la red.
- **addNode,** sirve para agregar los nodos a la topología.
- **addPort,** sirve para el mapeo de puertos en la conexión.
- **addSwitch,** sirve para agregar los switchs en la topología.

12.5.4.3 MINIEDIT.

Es otra forma de crear topologías en mininet, es mucho más didáctica que las anteriores y se puede crear una red personalizada, esta herramienta no necesita que se acuerde de ningún comando y tampoco que se domine el lenguaje de Python para poder ser utilizada.

Para poder ingresar a la carpeta de miniedit y poder ingresar a la interfaz gráfica, se debe abrir un terminal en Ubuntu, e ir ingresando a las carpetas de /mininet, después a la carpeta de /examples, escribir el comando "ls" que permite visualizar los documentos y carpetas existentes, por último se debe escribir el comando "./miniedit.py" para ingresar a miniedit.

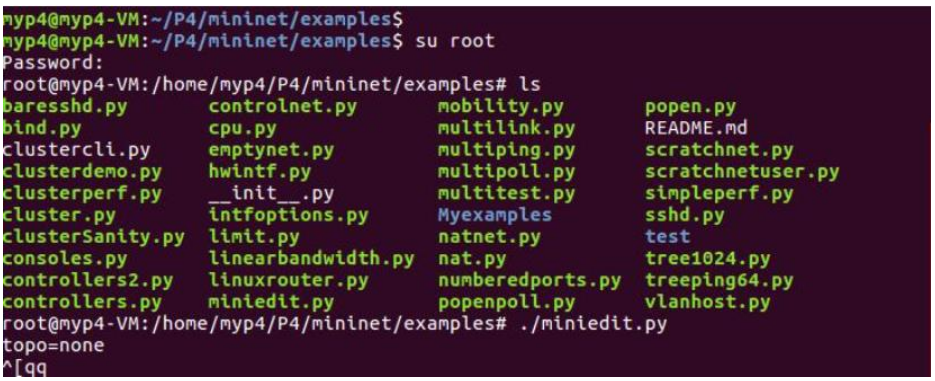


Figura 26. Ingresar miniedit.

Se abrirá una ventana con la interfaz gráfica de miniedit.

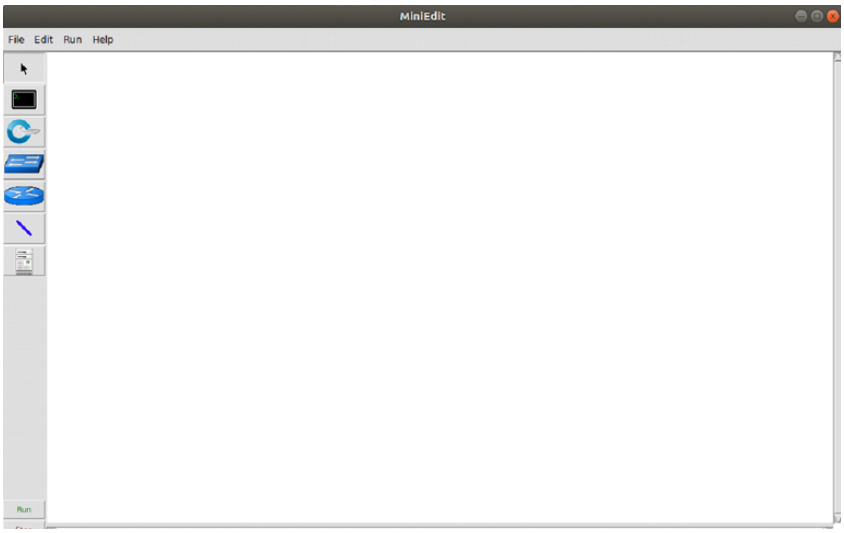


Figura 27. Miniedit.

En esta ventana se puede crear la topología que se desee, para poder correr una SDN, pero antes de iniciar es importante conocer cómo se utiliza las herramientas de miniedit que podemos observar en la parte izquierda de la ventana.

- **Herramienta de selección** que permite mover los nodos en la ventada de miniedit y para poder seleccionar al momento que se desee entrar a la configuración de algún equipo de la red SDN.

Figura 28. Herramienta de selección.

- **Herramienta de Host**, nos permite seleccionar uno o varios host, para implementarlos en la red, y además pueden ser configurados como si fuera un host físico.

Figura 29. Herramienta de Host.

- **Herramienta de conmutador,** esta herramienta sirve para seleccionar un open switch, que es el que fue usado para las pruebas, dado que este switch contiene el protocolo openflow.

Figura 30. Herramienta de conmutador.

- **Herramienta de conmutador tradicional,** sirve para crear switch tradicionales, los cuales vienen con una configuración predeterminada y no se puede configurar desde miniedit, no se usara en este proyecto.

Figura 31. Herramienta de conmutador tradicional.

- **Herramienta de router tradicional,** sirve para agregar un router tradicional en miniedit, no puede ser configurado desde miniedit.

Figura 32. Herramienta de router tradicional.

- **Herramienta de enlaces,** esta herramienta sirve para realizar los enlaces entre los switchs con el controlador, entre los mismos switch y para conectar los host con los switch.

Figura 33. Herramienta de enlaces

.

- **Herramienta de controlador,** la herramienta sirve para implementar controladores a la red, que por defecto viene con un controlador que ejecuta openflow, se puede configurar otro tipo de controlador configurándolo desde el menú dando click derecho en el controlador.

Figura 34. Herramienta de controlador.

Podemos visualizar una red ya creada en la interfaz gráfica de miniedit donde se puede observar que se encuentra un controlador openflow, varios host y conectados a los host y controlador los switch openflow.

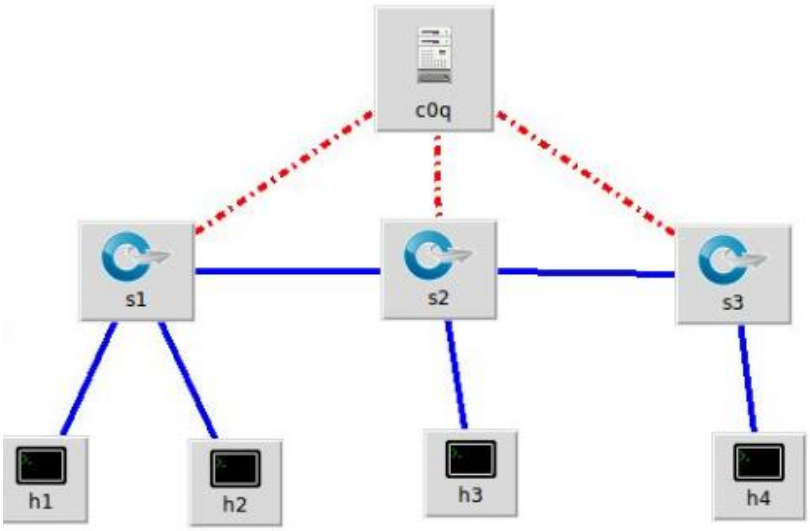


Figura 35. Topología creada en miniedit.

Un aspecto importante es que se puede guardar las topologías para luego poderlas obtener al instante o simplemente correrlas sin necesidad de abrir miniedit, para ello vamos a la parte superior de la ventana de miniedit y se le da click en el apartado de “File” se abre submenú en el que se debe poner “export level 2 scrip”, el cual desplegara una ventana donde se guardara la topología, la extensión con la que se guarda en “.py” es decir un script de Python. [13]

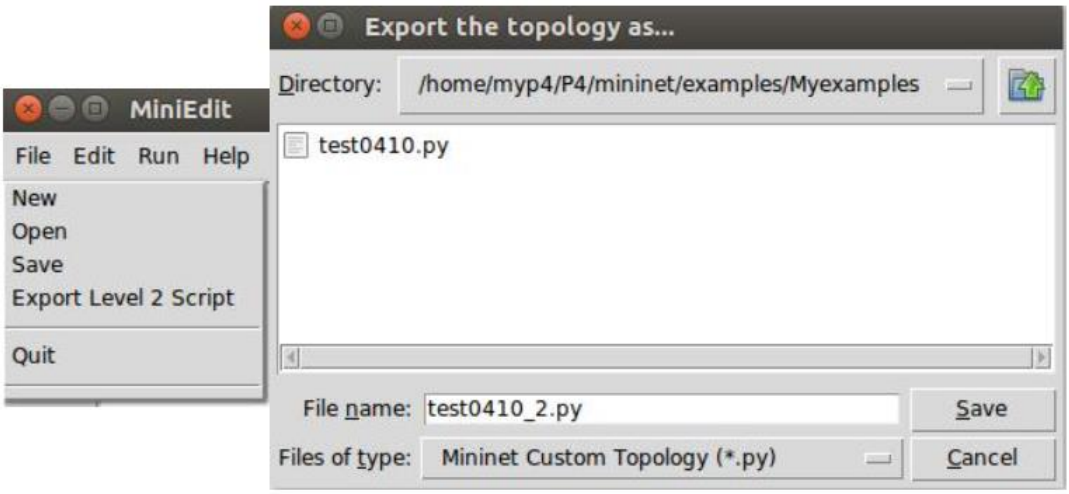


Figura 36. Guardar topología de miniedit.

12.6 CONTROLADORES

Los controladores SDN son el cerebro de la red, son los que gestionan todo el tráfico y las políticas de enrutamiento, en este apartado se seleccionaron 3 controladores que fueron analizados y se seleccionó uno, que fue el que se implementó en el prototipo de red SDN.

12.6.1 OPENDAYLIGHT.

Este controlador como los otros seleccionados son de código abierto opensource, por lo que no es necesario pagar por la licencia para ser usado, es un controlador para SDN, opendaylight (ODL) permite la comunicación con el protocolo openflow antes ya mencionado, el controlador es el que decide por donde enviar el tráfico a los conmutadores conectados en toda la SDN.

En las redes tradicionales cada router o switch tiene políticas de enrutamiento y tablas de flujo y depende de cada fabricante como configure sus equipos, con openflow no existe ese inconveniente dado que el controlador es quien gestiona y los equipos solo

envían los datos por donde el controlador decide, así es como se consigue la centralización de la red.

Una de las mayores ventajas del controlador opendaylight es que tiene una avanzada interfaz gráfica que nos permite la visualización de la red, teniendo el control centralizado de los equipos físicos y virtuales, con las APIs podemos crear nuevas aplicaciones y personalizar la configuración de la red. [14]

12.6.1.1 INSTALACION DE OPENDAYLIGHT

Para su instalación, lo primero que debemos saber es, abrir el terminal de Ubuntu, con el comando Ctrl + Alt + T, y a continuación, seguir el siguiente proceso.

- Se pone el comando “sudo apt install openjdk-8-jre-headless” dado que en esta versión trabaja opendaylight.

Figura 37. Instalación de jdk.

- En este momento se ejecuta la variable global con el comando “JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64” y a continuación el comando “source ~/.bashhrc”

Figura 38. Instalación de la variable global.

- Se procede a poner la ubicación del archivo que se va a descargar en este caso la versión es la lithium 0.3.0 de opendaylight.

Figura 39. Extracción de opendaylight.

- Una vez descargado el opendaylight, se debe desempaquetar para que funcione con normalidad con el comando “tar –xvf distribution-karaf-0.3.0-lithium.tar.gz”.

Figura 40. Desempaquetado de opendaylight.

- Terminado todo el proceso de instalación ahora se debe ejecutar el opendaylight para el funcionamiento, debemos entar a la carpeta de distribution karaf y ejecutar el comando “sudo ./bin/karaf” y debe aparecer una imagen similar a la de la Figura. 40 para saber que este ejecutándose opendaylight.

Figura 41. Opendaylight en funcionamiento.

Ahora que ya está instalado opendaylight ya se puede usar conectándolo con mininet para realizar las pruebas de tráfico, para determinar cuál controlador seleccionar para la implementación del prototipo de red SDN.

12.6.2 CONTROLADOR POX.

El controlador POX es muy sencillo de instalar y de agregar a mininet, no tiene una interfaz gráfica y está escrita en el lenguaje de programación Python, simplemente se debe iniciar el principal componente que es pox.py, que se encargará de reconocer los demás módulos para su funcionamiento. [14]

12.6.2.1 INSTALACION DE POX

- La instalación del controlador se hace mediante el terminal de Ubuntu, ya antes mencionado, se escribe el comando "mininet/útil/install.sh -p" y procederá a la clonación para la instalación.

Figura 42. Instalación de controlador pox.

- Para comprobar su instalación y su funcionamiento se utilizará el siguiente comando "python2 pox.py forwarding.l2 learning", se debe abrir otra terminal, para crear una topología, y verificar que el controlador, se conecte con la topología, de la dirección de la tarjeta de red (127.0.0.1).

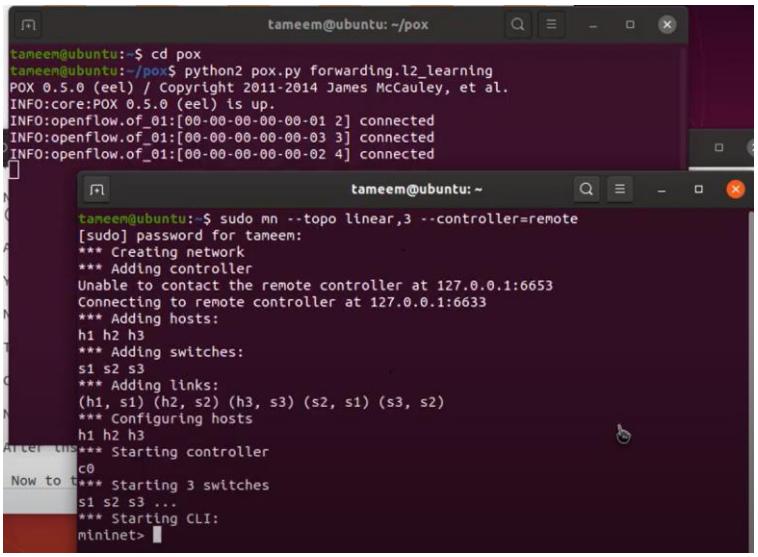


Figura 43. Controlador Pox funcionando.

12.6.3 CONTROLADOR FLOODLIGHT

Es otro controlador de red SDN que se encarga de gestionar los equipos interconectados de la SDN, el controlador es de código abierto por lo que se puede utilizar sin ningún costo por licencia, además Floodlight tiene una interfaz gráfica, lo que permite visualizar a los equipos conectados en la SDN, está a su vez se conectó con mininet donde se realizaron pruebas de tráfico de datos para ser comparados con los otros controladores. [6]

12.6.3.1 INSTALACION DE FLOODLIGHT.

En las instalaciones anteriores de los controladores fue mediante los repositorios que existen en los servidores, en el caso de floodlight se efectuó mediante la descarga de la imagen del controlador, para conocer cómo se realizan las diferentes formas de instalar los controladores.

- Primero se debe ir a la página de floodlight para proceder a la descarga de la imagen.
  https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343542/Getting+Started

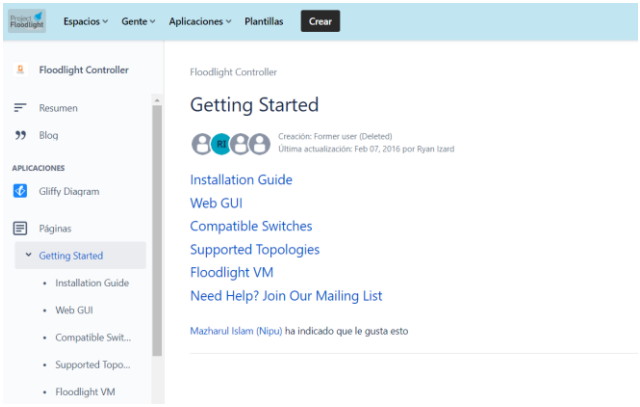


Figura 44. Página de descarga de floodlight.

- A continuación se selecciona floodlight VM y se espera el tiempo de descarga, dependiendo del internet que se tenga contratado, se demorará el archivo en descargarse.

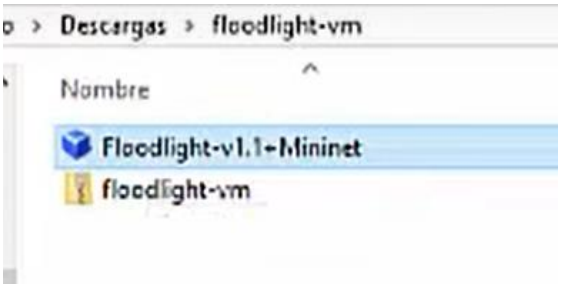


Figura 45. Descarga del archivo

- Se crea una nueva máquina virtual, con la imagen descargada, e iniciamos en un sistema operativo Ubuntu, el usuario = Floodlight y el password = Floodlight

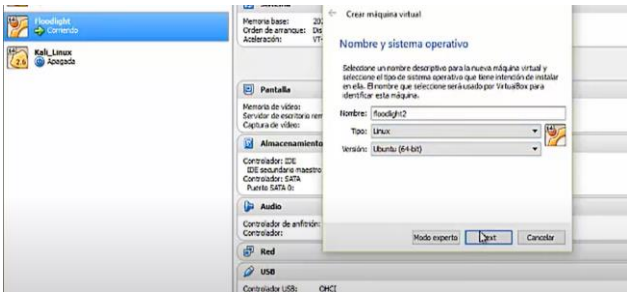


Figura 46. Máquina virtual Floodlight.

- Para iniciar el controlador Floodlight se debe abrir un terminal de Ubuntu, ir a la carpeta de Floodlight con el comando “cd Floodlight”, al ingresar en esa carpeta colocar el comando “java –jar target/floodinig.jar”, en ese momento empezara a correr el controlador Floodlight.

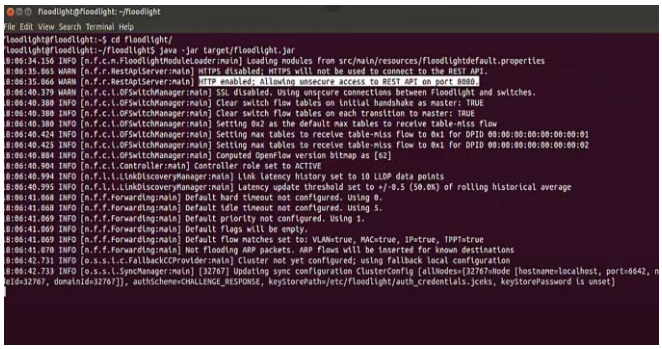


Figura 47. Floodlight en funcionamiento.

12.7 IMPLEMENTACION DEL PROTOTIPO DE RED SDN.

Para la implementación del prototipo se tuvo que realizar la instalación de una máquina virtual a través del software VirtualBox, para ejecutar un sistema operativo Linux, opensource, específicamente se instaló Ubuntu 20, a continuación se instaló el simulador de red SDN, llamado mininet, donde se realizaron las pruebas de tráfico con los diferentes controladores que se instalaron.

12.7.1 ANALSIS Y SELECCIÓN DEL CONTROLADOR SDN

El análisis de los controladores se llevó a cabo mediante el software Wireshark para visualizar el tráfico que se generó, verificar la latencia mínima, latencia máxima de los controladores al gestionar la red SDN, y observar si hubo pérdidas de paquetes al momento de enviar y recibir mensajes.

Para ello se utilizó primero un envío de 50 paquetes y se analizó el tiempo de latencia máximo y mínimo de cada uno de los controladores así como el ancho de banda de la interfaz, con wireshark podemos visualizar la salida de los paquetes de cada uno de las interfaces conectadas.

Para poder ejecutar wireshark simplemente se debe ejecutar el comando “sudo wireshark” y el programa iniciará.

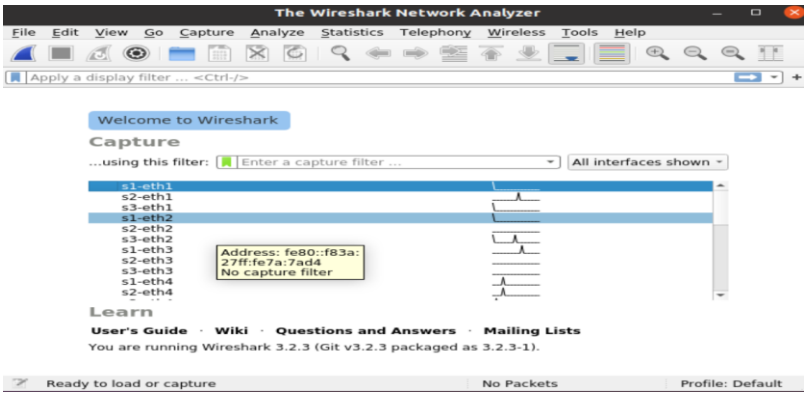


Figura 48. Wireshark

Envió de 50 paquetes en el controlador Opendaylight gestionando la red.

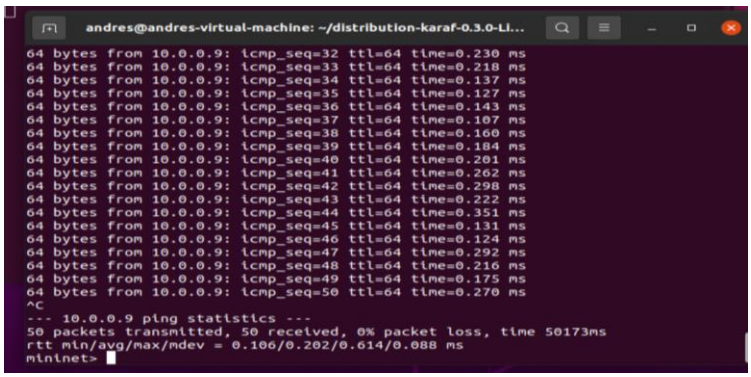


Figura 49. Envió de paquetes

Figura 50. Ancho de banda

Se utilizó el controlador Pox para el siguiente envió de paquetes y verificar si existen perdidas, ver su latencia y su ancho de banda.

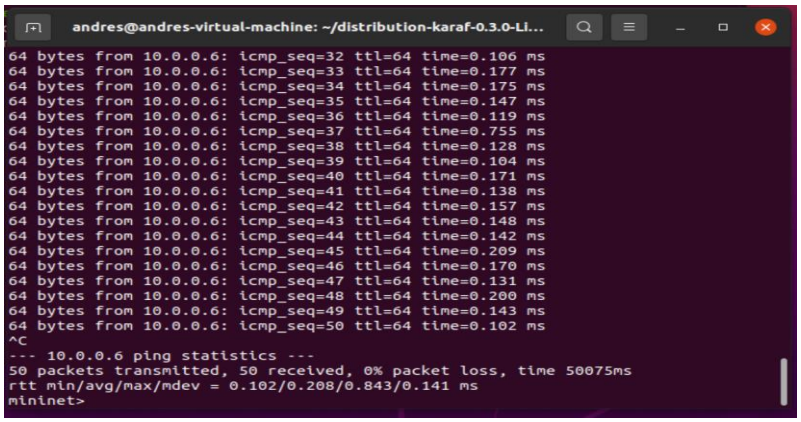


Figura 51. Envió de paquetes Pox.

Figura 52. Ancho de banda en controlador Pox

A continuación se verificará los paquetes enviados en el controlador Floodlight, para visualizar si existen paquetes perdidos y la latencia.

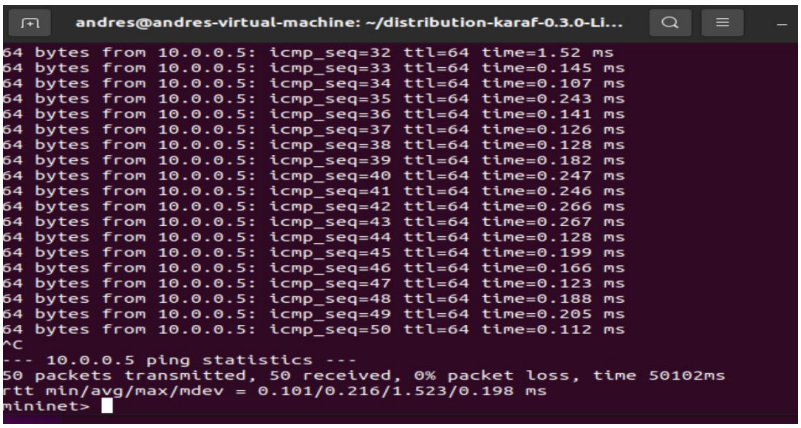


Figura 53. Envió de trafico Floodlight.

Figura 54. Ancho de banda Floodlight.

12.7.2 DISEÑO EIMPLEMENTACION DE UN PROTOTIPO DE RED SDN PARA EL CAMPUS NORTE DE LA UNACH.

El diseño del prototipo se realizó en el laboratorio del campus norte de la Unach, donde el principal objetivo fue realizar las interconexiones de los equipos de la red, y así la universidad pueda tener el primer prototipo SDN en funcionamiento.

El principal problema que surgió al momento de diseñar la red que debía ser implementada fue el equipo de conmutación (switch), que debían soportar el protocolo openflow, que es fundamental para que los equipos se puedan comunicar, dado que SDN todavía está en etapa de desarrollo no existe en el mercado gran

cantidad de equipos que puedan soportar este protocolo, por lo que hubo que investigar equipos que se adapten a las necesidades que se requería, el equipo seleccionado para realizar la conmutación fue el dispositivo de la marca Mikrotik, modelo hAP lite RB941-2nD, que tenía una actualización que permitía ejecutar openflow.

12.7.2.1 DISEÑO DE LA RED SDN.

Para el diseño de la red, lo primero es saber el equipo que se va a utilizar en la implementación, en este caso lo principal es el controlador y los switchs openflow que fue la base fundamental para el prototipo.

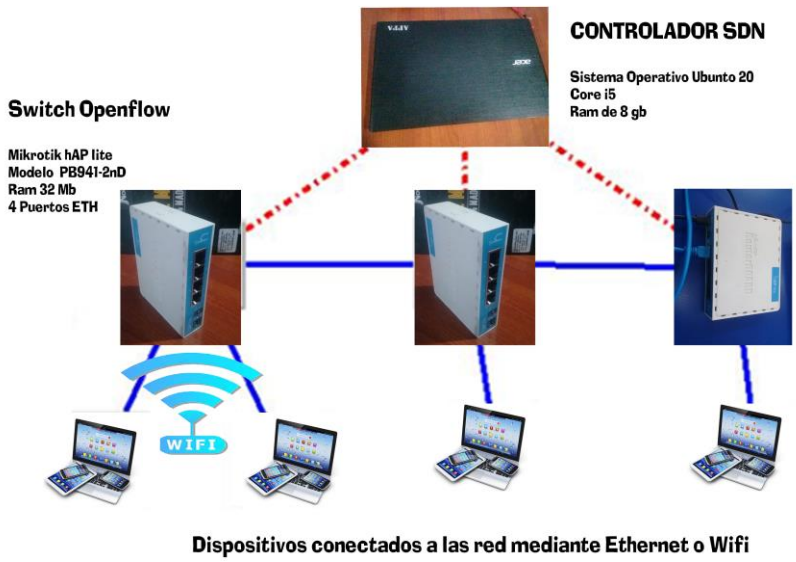


Figura 55. Diseño de prototipo de red SDN

- Controlador SDN, es una computadora donde se encuentra el controlador que fue usado para la implementación de SDD, las características son:

  - Sistema operativo Ubuntu 20
  - Ram de 8 Gb

- Disco duro de 1 Tb
- Procesador Core i5

- Switch Openflow, se utilizó el routerboard de la marca Mikrotik, por la razón que en ellos se puede utilizar el protocolo openflow.

  - Marca Mikrotik
  - Modelo RB941-2nD
  - Ram de 32 Mb
  - Puertos 4 Ethernet
  - Wifi incorporado

12.7.2.2 CONEXIONES Y CONFIGURACIÓN DEL PROTOTIPO SDN.

Para las conexiones y la configuración de los equipos que se va a utilizar se empezó por el controlador, lo primero fue ejecutarle para que este activo en la red. Para su inicialización se debe ingresar en el terminal de Ubuntu 20 con el comando Ctrl + Alt + T, después se debe ingresar a la carpeta donde se fue instalado el controlador con el comando "cd (nombre de la carpeta)" una vez en la carpeta se debe ejecutar el comando "./bin/karaf" para que se inicialice el controlador. [13]

Figura 56. Inicialización de Opendaylight.

Para la configuración de los Routerboard de Mikrotik y funciones como switch openflow se debe tener mucho cuidado, dado que la última versión de RouterOS, que es la versión 7, no tiene incorporado la herramienta openflow por lo que no es compatible con la investigación, la solución fue bajar la versión de RouterOS a la

versión 6.48, y descargar todos los paquetes extras que contiene, además de instalar Winbox, que es la herramienta que permite el ingreso al sistema operativo del Mikrotik para realizar la configuración, Enlace de las versiones de Mikrotik y de Winbox. https://mikrotik.com/download

Se utilizó el Routerboard Mikrotik porque es opensource, no es necesario pagar por la licencia para su utilización, a continuación la configuración para instalar openflow en los equipos routerboard.

- Para empezar a emplear el routerboard se debe resetearlo para quitar cualquier configuración que exista en el Mmikrotik, esto se hace, manteniendo aplastado el botón "reset" durante unos 5 segundos hasta que la luz empiece a parpadear.

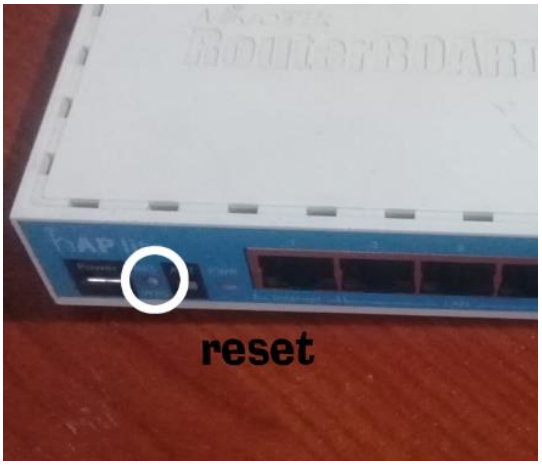


Figura 57. Resetear Mikrotik.

- Se ejecuta Winbox para poder ingresar al sistema operativo del routerboard y empezar las configuraciones.

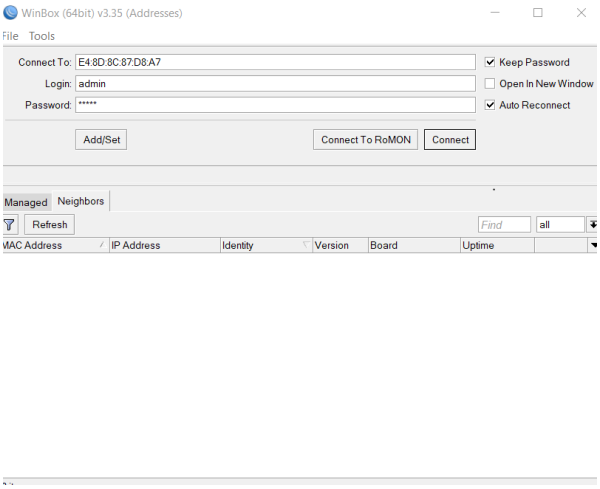


Figura 58. Ejecutar Winbox.

- Iniciamos Winbox donde debemos de colocar el ip 192.168.88.1, el login = admin y en password = Vacio.

- En este momento de proceder a la configuración openflow de Mikrotik, reviso que se encuentre la herramienta de openflow en el routerboard, en caso de no estar se debe descargar los paquetes extras para la version 6.48 que es la que está instalada.

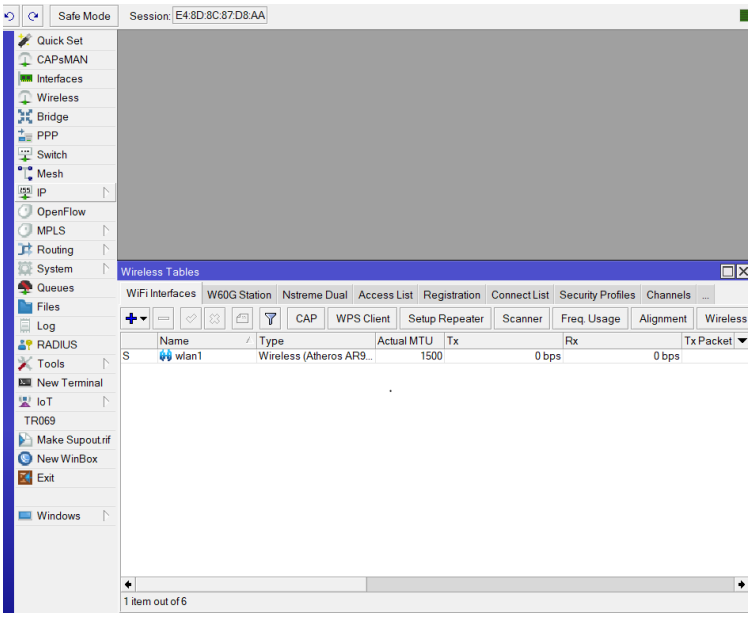


Figura 59. Herramienta Openflow en Mikrotik.

- Para poder realizar la conexión con el controlador y el routerboard se debe asignar un puerto Ethernet, en este caso se conectó con el puerto Ethernet 1 y se realizó la configuración, se debe colocar la misma red del controlador, en este caso es la 192.168.134.0, para conocer la red del controlador se debe ir primero a la máquina virtual, abrir un terminal y colocar el comando "ifconfig" donde se visualizara a que ser pertenece.

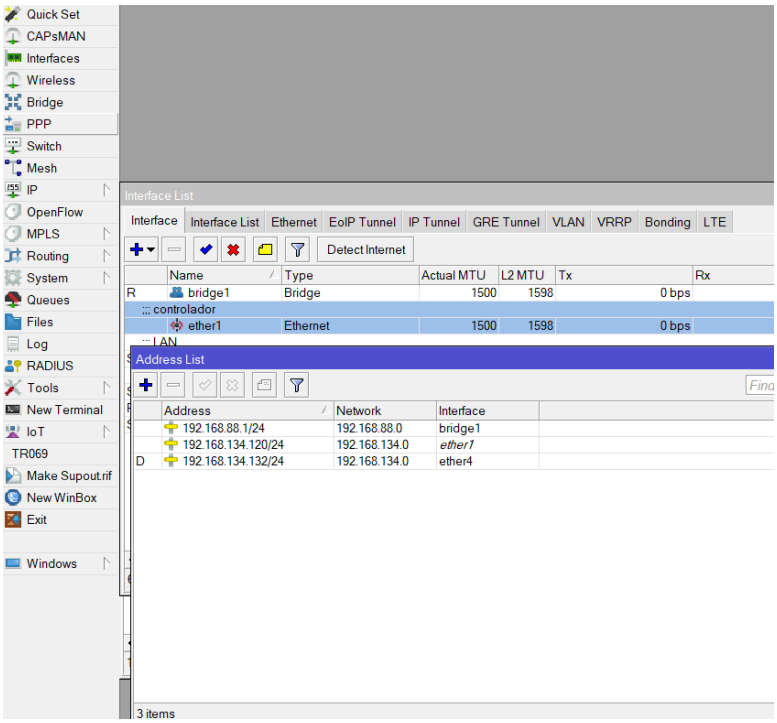


Figura 60. Asignación de puerto para controlador.

- Para la configuración de los puertos openflow, se debe ir a la herramienta de openflow, al seleccionar se despliega una pantalla donde se debe agregar los puertos que se desea añadir, seleccionando el símbolo de "+" se puede ir agregando los puertos.

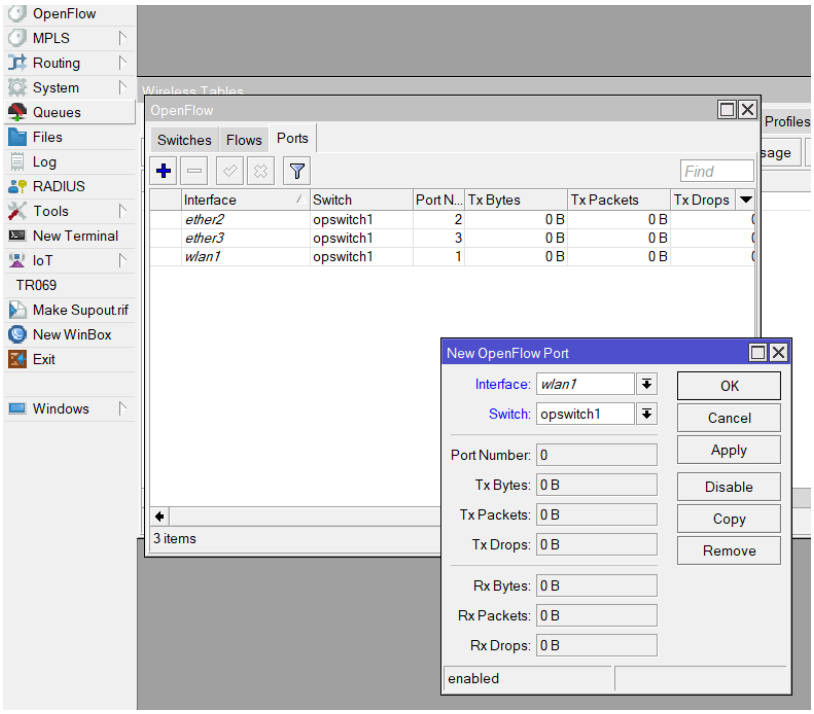


Figura 61. Implementación de puertos openflow.

Se activó la red wifi, que tiene Mikrotik, para que más usuarios se puedan conectar a la red SDN, se debe conectar el routerboard con el cable UTP al controlador para que se realice la comunicación, así como a los demás routerboard, se inicia Opendaylight para la visualización de la red que se creó, para ingresar a opendaylight se debe ingresar mediante un servidor , el cual en este caso es “http:// 192.168.134.128:8181/index.html” el cual pedirá un usuario y una contraseña, en los dos caso se debe poner “admin”, con ello nos visualizara la topología.

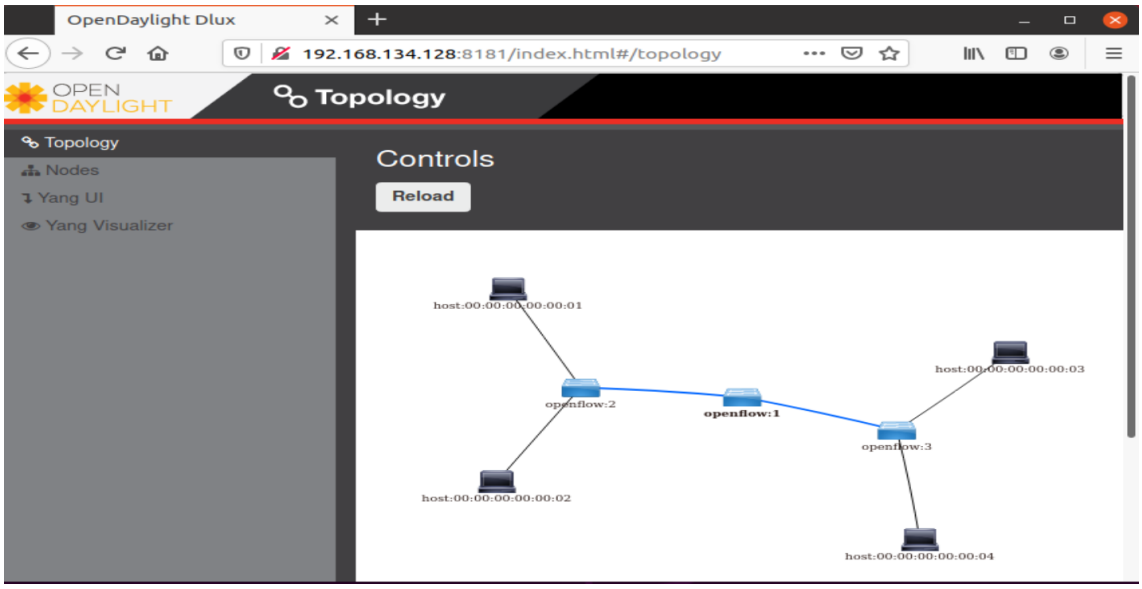


Figura 62. Implementación Visualizada en el controlador ODL

13. CAPÍTULO III. METODOLOGIA.

Tipo de estudio

En el presente proyecto de investigación se aplicó el método de investigación experimental dado que se recolectó datos como latencia, envió de paquetes, en las simulaciones que se llevaron a cabo de una red SDN. Los diferentes controladores SDN fueron comparados y con los resultados obtenidos se seleccionará el más adecuado, según los parámetros que requiera nuestra red.

Método de investigación

El proyecto de investigación que se presenta pertenece al tipo de investigación aplicada, puesto que con los conocimientos obtenidos previamente, fueron aplicados al beneficio de la sociedad, para ello se necesitó el análisis de los controladores SDN que se realizó mediante los softwares de simulación y la implementación de un prototipo de red SDN.

Además, pertenece al método descriptivo, dado que se basa en la observación, al momento de analizar las tablas y gráficas que se obtuvo de las variables del proyecto de investigación.

Muestra y población

Población

- En este proyecto se contemplan 5 poblaciones de 100 datos de latencia correspondientes a, 25, 50, 100, 125 y 150 paquetes enviados.

Muestra

- No hay muestra se trabaja con toda la población en el proyecto.

Técnica de recolección de datos

- Observación: es el proceso en el que se percibió los resultados del análisis mediante software de los controladores como de la red en sí mismo. Con las herramientas existentes en el propio emulador de red y del software Wireshark.

Variables

Variable Independiente

- Números de paquetes enviados.

Variable Dependiente

- Latencia

14. CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

14.1 RESULTADOS DE WIRESHARK

Los controladores SDN fueron analizados por el programa opensource, denominado Wireshark, el cual nos genera una gráfica de tiempo versus paquetes enviados, se realizó de cada uno de los controladores para seleccionar el más adecuado, todos los controladores fueron analizados con la misma topología simulada que se implementó de la red SDN

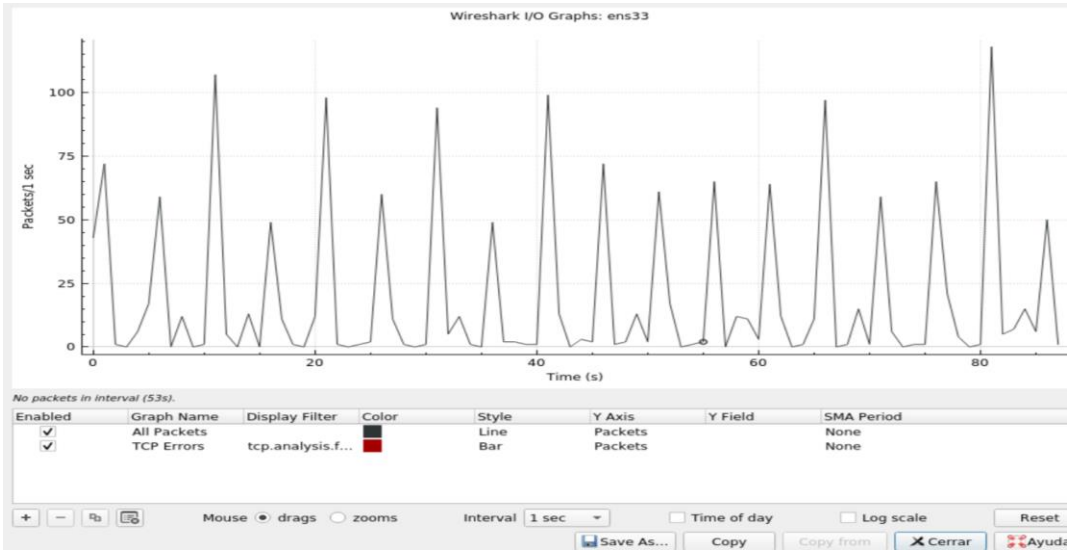


Figura 63. Grafica del controlador ODL.

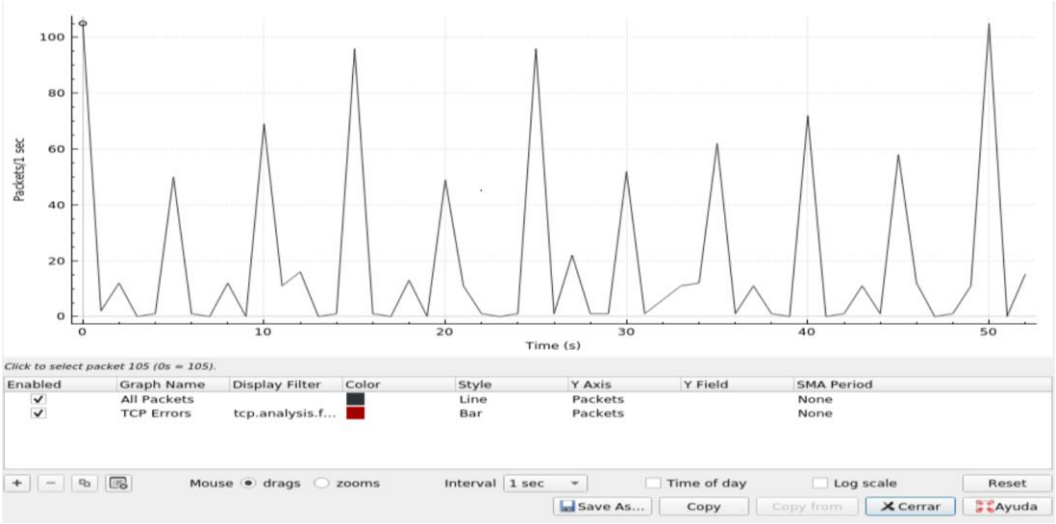


Figura 64. Grafica del controlador Floodlight.

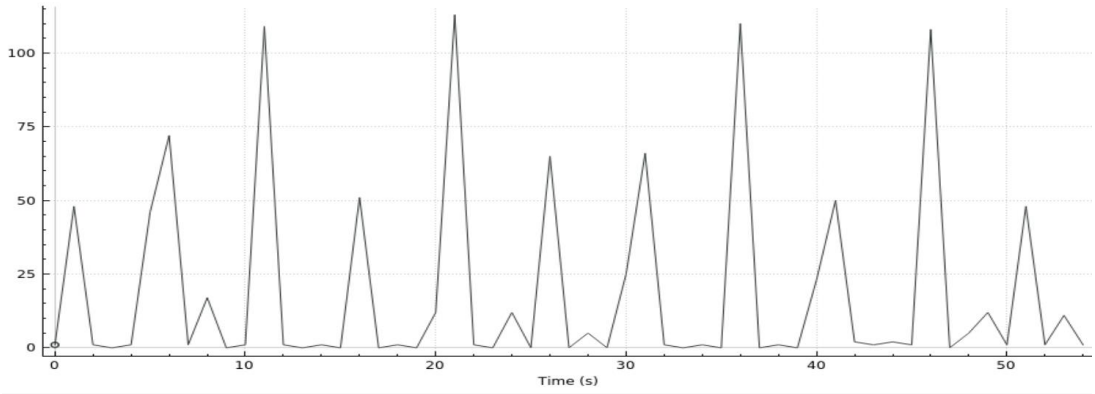


Figura 65. Grafica del controlador Pox.

14.2 PAQUETES ENVIADOS Y LATENCIA

A continuación se observa una comparación de paquetes enviados, recibidos, latencia mínima y latencia máxima, recopilada de los controladores SDN, que se analizaron para seleccionar el más adecuado para la implementación. Para este análisis se tomó toda la población, la variable independiente fue modificada de 25, 50, 100, 125 y 150 paquetes enviados, para obtener la latencia dependiendo del número de paquetes enviados.

NOMBRE CONTROLADOR	PAQUETES ENVIADOS	PAQUETES RECIBIDOS	LATENCIA MINIMA (MS)	LATENCIA MAXIMA (MS)	LATENCIA PROMEDIO (MS)

OPENDAYLIGHT	**25**	**25**	**0.091**	**0.725**	**0.166**
FLOODLIGHT	**25**	**24**	**0.089**	**0.502**	**0.188**
POX	**25**	**25**	**0.090**	**0.726**	**0.169**

Tabla 1. *Comparación de controladores.*

NOMBRE CONTROLADOR	PAQUETES ENVIADOS	PAQUETES RECIBIDOS	LATENCIA MINIMA (MS)	LATENCIA MAXIMA (MS)	LATENCIA PROMEDIO (MS)
OPENDAYLIGHT	**50**	**50**	**0.106**	**0.614**	**0.202**
FLOODLIGHT	**50**	**47**	**0.102**	**0.843**	**0.208**
POX	**50**	**50**	**0.101**	**1.523**	**0.216**

Tabla 1. *Comparación de controladores.*

NOMBRE CONTROLADOR	PAQUETES ENVIADOS	PAQUETES RECIBIDOS	LATENCIA MINIMA (MS)	LATENCIA MAXIMA (MS)	LATENCIA PROMEDIO (MS)
OPENDAYLIGHT	**100**	**100**	**0.086**	**0.599**	**0.145**
FLOODLIGHT	**100**	**98**	**0.083**	**1.195**	**0.179**
POX	**100**	**100**	**0.081**	**0.640**	**0.166**

Tabla 1. *Comparación de controladores.*

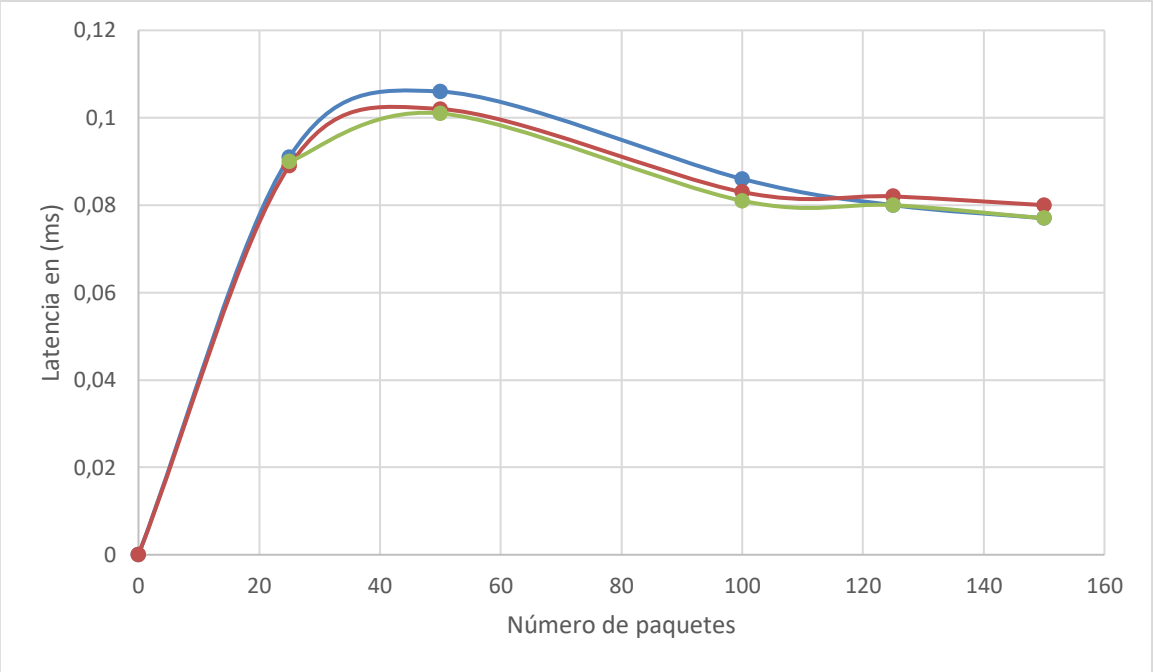
NOMBRE CONTROLADOR	PAQUETES ENVIADOS	PAQUETES RECIBIDOS	LATENCIA MINIMA (MS)	LATENCIA MAXIMA (MS)	LATENCIA PROMEDIO (MS)
OPENDAYLIGHT	**125**	**125**	**0.080**	**0.555**	**0.161**
FLOODLIGHT	**125**	**124**	**0.082**	**0.860**	**0.173**
POX	**125**	**125**	**0.080**	**0.597**	**0.165**

Tabla 1. *Comparación de controladores.*

NOMBRE CONTROLADOR	PAQUETES ENVIADOS	PAQUETES RECIBIDOS	LATENCIA MINIMA (MS)	LATENCIA MAXIMA (MS)	LATENCIA PROMEDIO (MS)
OPENDAYLIGHT	**150**	**150**	**0.077**	**0.599**	**0.161**
FLOODLIGHT	**150**	**149**	**0.080**	**1.026**	**0.164**
POX	**150**	**150**	**0.85**	**0.575**	**0.164**

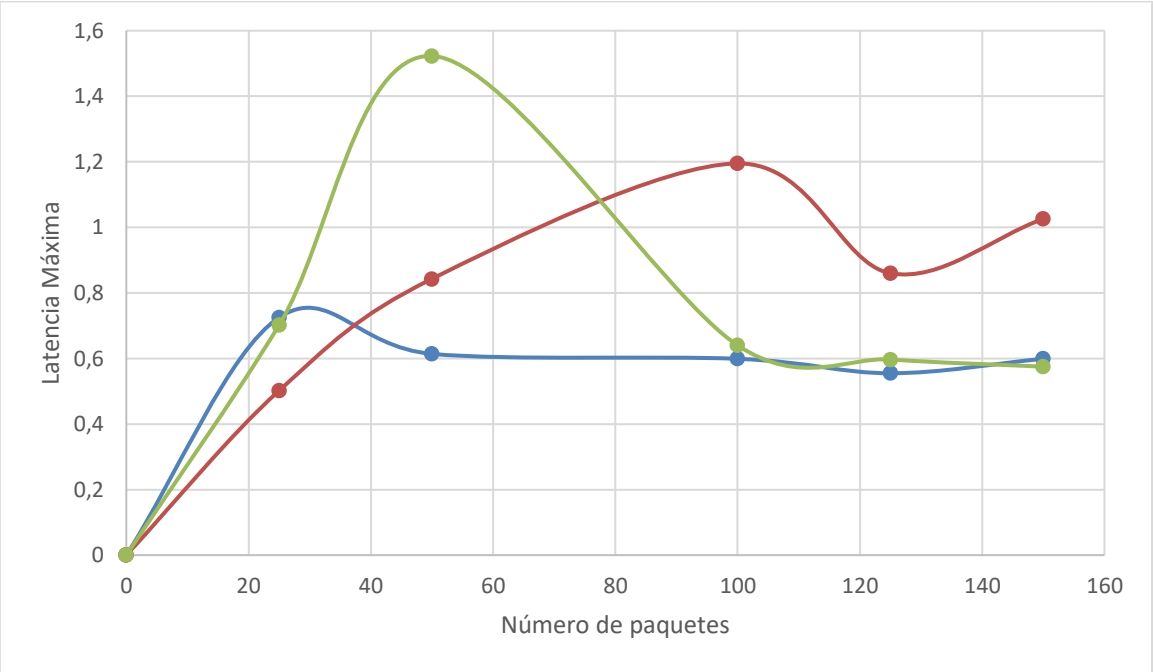
Tabla 1. *Comparación de controladores.*

En la gráfica se muestra el valor de latencia mínima en los paquetes enviados de cada controlador, donde:
Verde = Opendaylight
Rojo = Floodlight
Azul = Pox



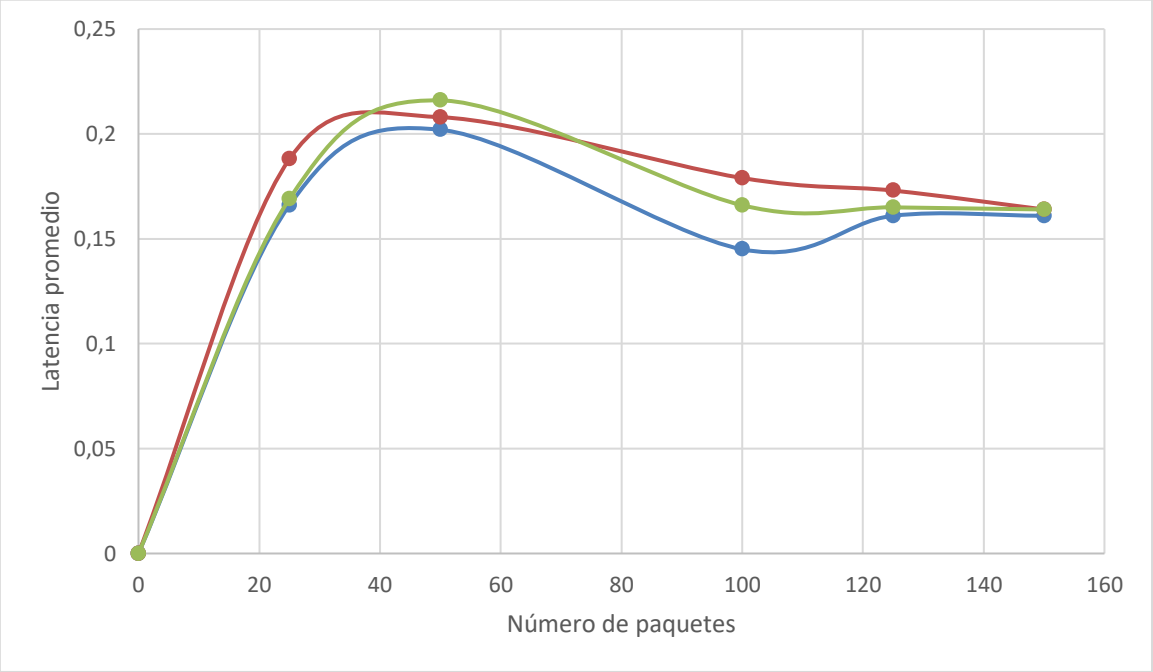
Grafica 1. *Latencia mínima de controladores.*

En la gráfica se muestra el valor de latencia máxima en los paquetes enviados de cada controlador, donde:
Azul = Opendaylight
Rojo = Floodlight
Verde = Pox



Grafica 2. *Latencia máxima de controladores.*

En la gráfica se muestra el valor de latencia maxima en los paquetes enviados de cada controlador, donde:
Azul = Opendaylight
Rojo = Floodlight
Verde = Pox



Grafica 3. *Latencia promedio de controladores.*

14.3 DISCUSIÓN.

- Se compara las gráficas que Wireshark visualizo y se observa como la gráfica de la Figura 62. Tiene la tasa más constante, entre la cantidad de paquetes y el tiempo, esta gráfica corresponde al controlador Opendaylight, la gráfica de la figura 63., perteneciente al controlador Floodlight, tiene bastantes picos bajos, lo que proporciona un envío de paquetes inferior en el mismo tiempo, y por último la gráfica 3 de la figura 64., perteneciente al controlador Pox, tiene su gráfica muy parecida al controlador Floodlight, tiene picos bajos de paquetes en el mismo tiempo que los otros dos controladores. En este aspecto del análisis, el mejor rendimiento es del controlador SDN Opendaylight.

- En el siguiente apartado se analizó, la latencia mínima, latencia máxima, latencia promedio y el número de paquetes recibidos, por lo que se seleccionó el óptimo controlador para ser implementado en la red.

- Para el envío y recepción de paquetes se observó los paquetes que fueron enviados y los paquetes que fueron recibidos, en este apartado el controlador Opendaylight y Pox, obtuvieron un porcentaje del 100% de recibidos y un 0% de perdidos, por el contrario, el controlador Floodlight tuvo una pérdida de paquetes al momento de recibirlos, en todas las simulaciones.

- En el apartado de latencias mínimas medidas en milisegundos, como se muestra en la Gráfica 1. Se puede observar como el controlador Opendaylight tiene la latencia mínima más baja que los otros controladores

- La latencia máxima de la Gráfica 2. Se puede observar que los controladores Pox y Floodlight son muy variables en este aspecto, mientras que se observa que el controlador Opendaylight tiene una latencia máxima más constante.

- En la Grafica 3. Que muestra la latencia promedio de los 3 controladores, se observa como Opendaylight tiene un óptimo rendimiento en esta variable.

- Con todos los datos recolectados y analizados, se seleccionó el controlador SDN OPENDAYLIGHT para ser implementado en el prototipo de red SDN en el laboratorio del campus norte de la Unach, al dar un óptimo rendimiento entre los otros controladores que fueron analizados, además de tener una interfaz gráfica con la que se puede visualizar los componentes que están conectados a la red.

15. CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

15.1 CONCLUSIONES.

- Se analizó 3 controladores opensource para al momento de utilizarlos no se deba pagar ningún tipo de licencia y el uso sea libre, mediante el análisis realizado durante el proyecto de investigación, se llegó a la conclusión que el controlador con mejor rendimiento y el más adecuado para el prototipo de red SDN que se implementó en el laboratorio de campus norte de la Unach.

- En el análisis del rendimiento se observó varios aspectos que fueron importantes para la selección del controlador más adecuado, entre ellos destaca, la latencia promedio, latencia mínima, latencia máxima, el envío y recepción de paquetes, con todas esas variables se concluye que el controlador OPENDAYLIGHT es el más adecuado.

- Se diseñó una red SDN que primero fue simulada en el emulador Mininet, para llevar a cabo las pruebas y análisis del rendimiento de la red, y posteriormente se implementó, con este análisis se concluye que se debe realizar un análisis extenso de los controladores SDN para el diseño de la red.

- Una de las partes más complicadas del proyecto de investigación fue la implementación del prototipo, dado que sigue en desarrollo y solo los grandes

fabricantes disponen de equipos específicos para esta arquitectura, se tuvo que adaptar equipos para que funcionen con los protocolos que son necesarios, con todo el análisis se concluye que se necesita seguir desarrollando SDN porque en un futuro serán las redes que sean implementadas en la internet, y una buena opción para los estudiantes e ingenieros es que sigan conociendo el avance de las SDN.

15.2 RECOMENDACIONES.

- Para las personas que deseen tener un conocimiento más avanzado sobre las redes SDN se recomienda la lectura de árticos y tesis, dado que todavía es una arquitectura en desarrollo y diferente a la arquitectura tradicional que hoy rige en la internet.

- Si se desea hacer implementaciones de red SDN se debe tener en cuenta el equipo que actué como swtich openflow, ya que es parte importante, y la información es limitada en este aspecto, se espera que con el paso del tiempo más personas se interesen por la SDN, y se recomienda una ardua investigación para los equipos que se desee implementar.

- Se recomienda, en un futuro, un avance de este mismo proyecto, pero con controladores de pago o propietarios, ya que en este proyecto se utilizó solo opensource, y sería interesante analizar si los controladores de código abierto ofrecen las mismas funcionalidades que el propietario, y seleccionando el mejor, escalar la red a una red con más cantidad de switchs y hots conectados.

16. BIBLIOGRAFIA

[1] L. Marrone, «Fundamentos de SDN: Software-Defined Networking,» pp. 5-7, 2019.

[2] L. Marrone, «Tema 4 Redes definidas por Software,» *Fundamentos de SDN,* pp. 3-10, 2019.

[3] J. L. Henao Ramirez, «Guia de implementacion y uso del emulador de redes mininet,» pp. 10-20, 2015.

[4] L. Marrone, «Fundamentos de SDN,» *Tema 2 : Open Networking Vxlan,* pp. 12-20, 2019.

[5] L. Marrone, «Fundamentos de SDN,» *Tema 3: NetDevOps a través de Ansible API,* pp. 7-12, 2019.

[6] E. Haro, «Análisis de criterios para la selección de un controlador,» pp. 1-8, 2019.

[7] G. Silva, «Modelos OSI y TCP/IP,» *issuu,* pp. 1-23, 2021.

[8] J. E. Martinez Copete, «Estudio del fundamento de la herramienta mininet,» pp. 35-40, 2015.

[9] M. D. Criollo Bustamante, «Politicas de Qos en redes empresariales para el análisis de rendimiento en entornos convenvionales y SDN,» pp. 48-50, 2020.

[10] S. Cordova Lopez, «Estudio de redes SDN mediante Mininet y Miniedit,» pp. 40-50, 2019.

[11] D. A. Serrano Carrera, «Redes Definidas po Software: Openflow,» pp. 25-35, 2015.

[12] M. M. Fernandez Mora, «Despliegue de una red SDN aplicando el protocolo MPLS y generando politicas de Qos para servicios de telefonia IP,» pp. 32-49, 2016.

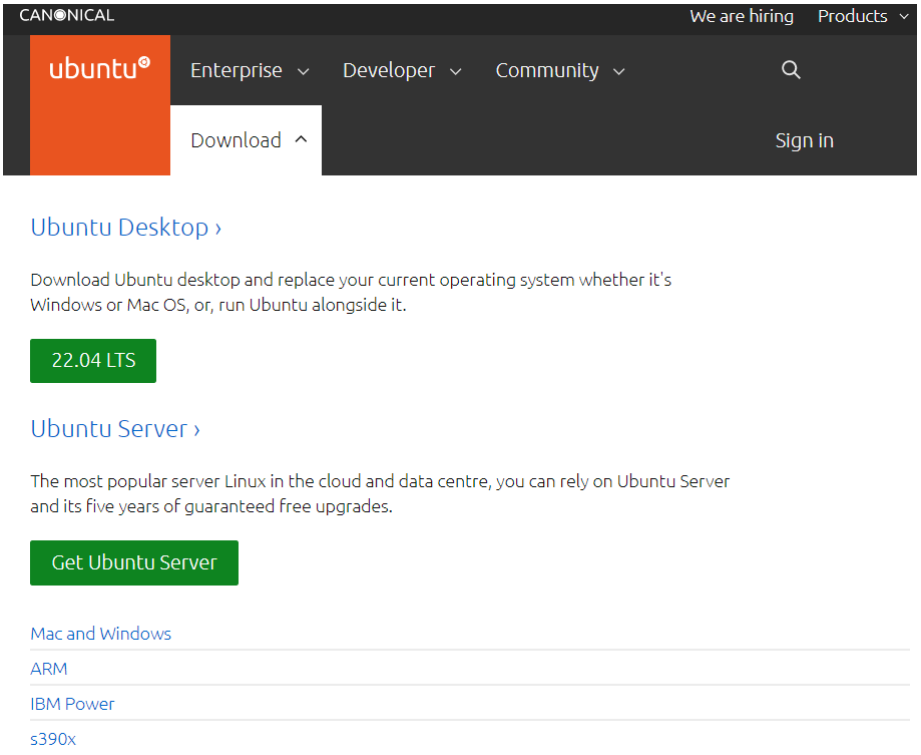
[13] D. L. Obando Jarrin, «Implementación de un testdeb para una red inalámbrica utilizando SDN,» pp. 9-46, 2018.

[14] D. X. Alban Ruiz, «Diseño e Implementación del prototipo de una red definida por software en la universida de las fuerzas armadas "espe",» pp. 77-110, 2015.

17. ANEXOS

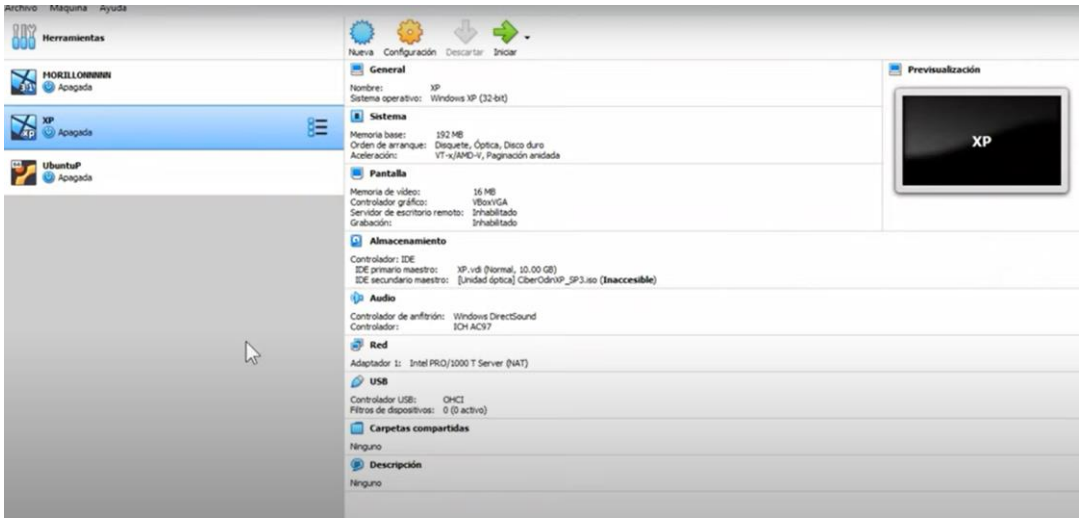
Instalación de Ubuntu para máquina virtual.

Primero se accede a la página oficial de Ubuntu para realizar la descarga de la imagen y poder ser instalada en la máquina virtual de VirtualBox. Enlace https://ubuntu.com/



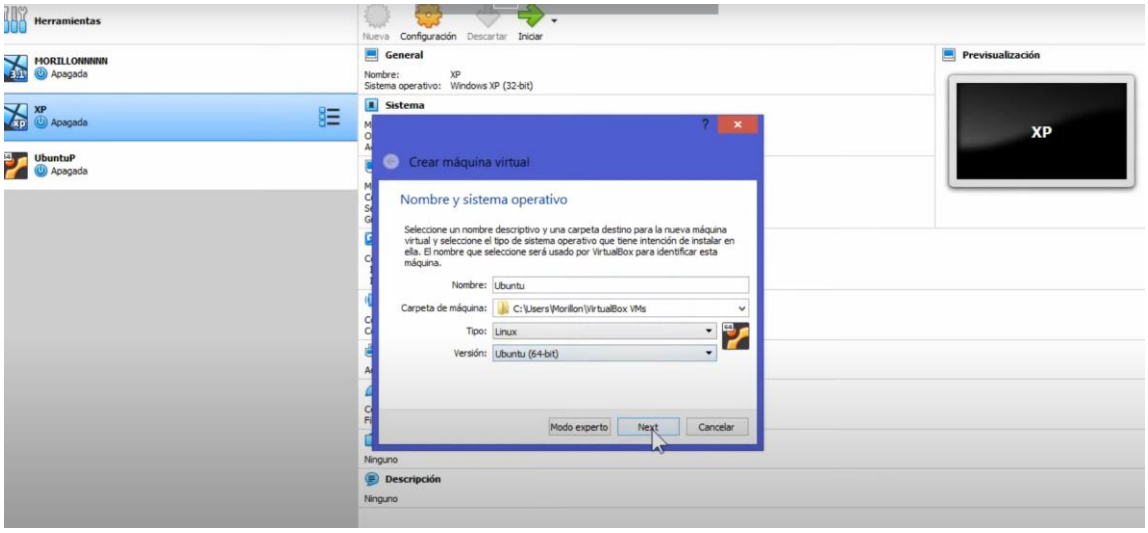
Anexo 1. Página de Ubuntu

A continuación se debe abrir el software de VirtualBox para proceder a la instalación de Ubuntu.



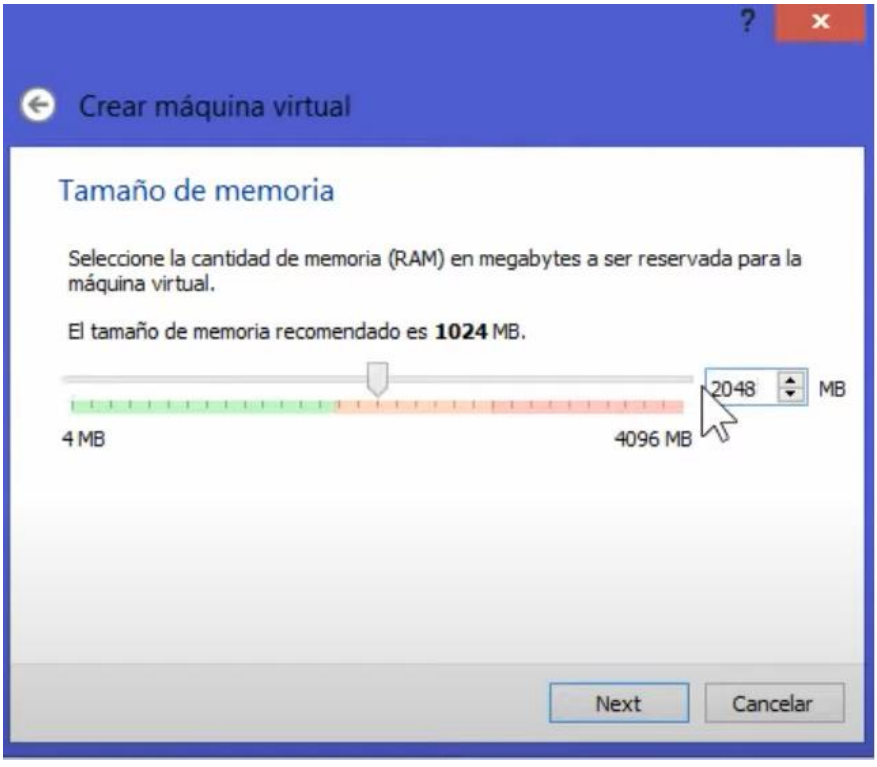
Anexo 2. Ejecutar VirtualBox.

Ahora se procede a realizar las configuraciones de una nueva máquina virtual, en la que se procede a dar click en la parte superior central con el nombre “nueva” y a continuación se colocara el nombre de la máquina virtual y la versión de Ubuntu.



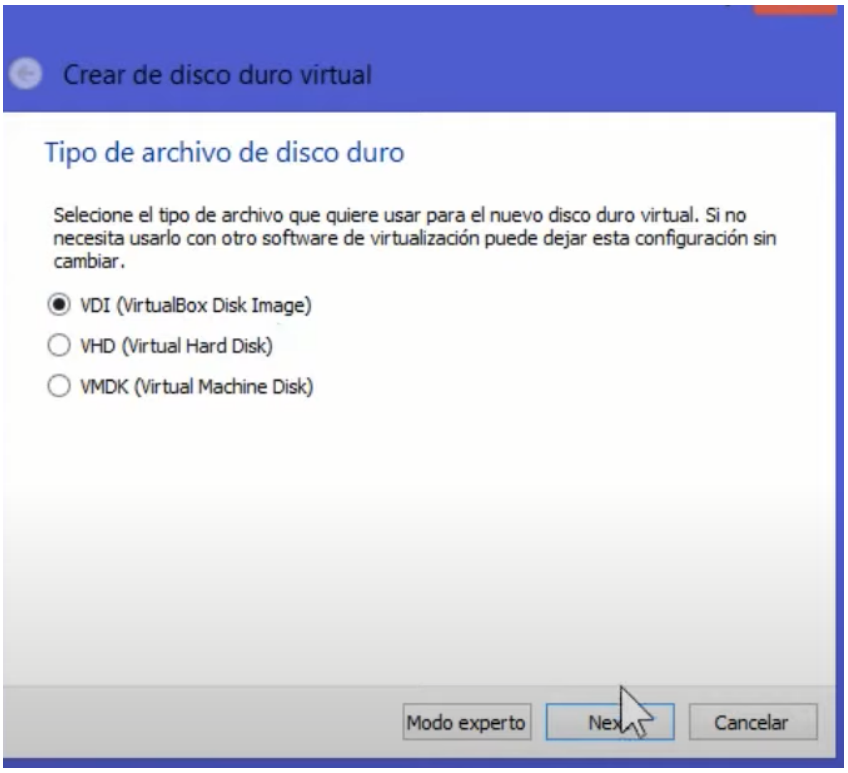
Anexo 3. Configuración Ubuntu.

Se le asigna una cantidad de memoria Ram, se compartida con nuestra memoria Ram de la computadora física.



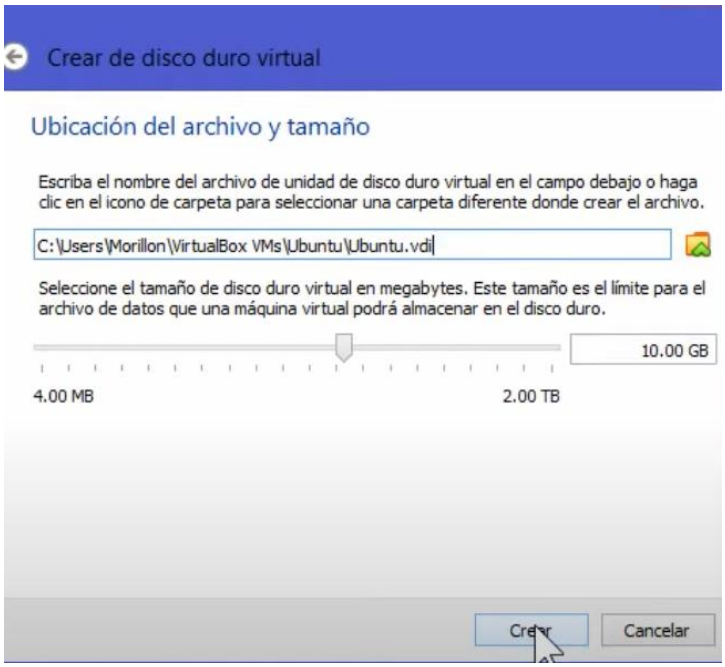
Anexo 4. Configuración de Ram.

Se selecciona una imagen virtual.



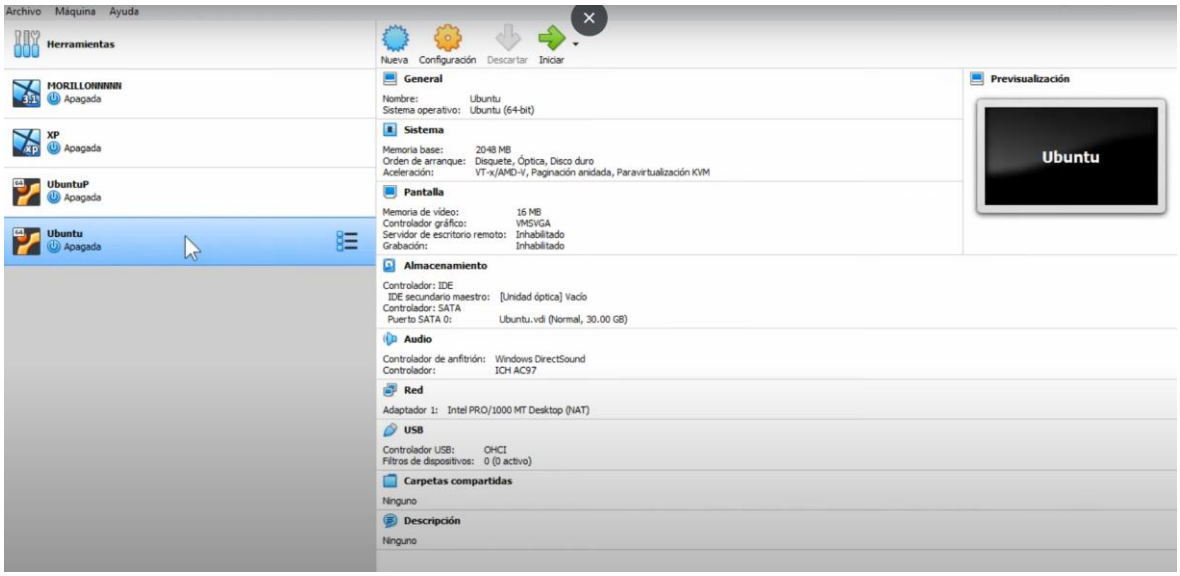
Anexo 5. Imagen virtual.

Se selecciona la ubicación del archivo y el disco duro de la máquina virtual que igual que la memoria Ram, será compartida por la maquina fisca.



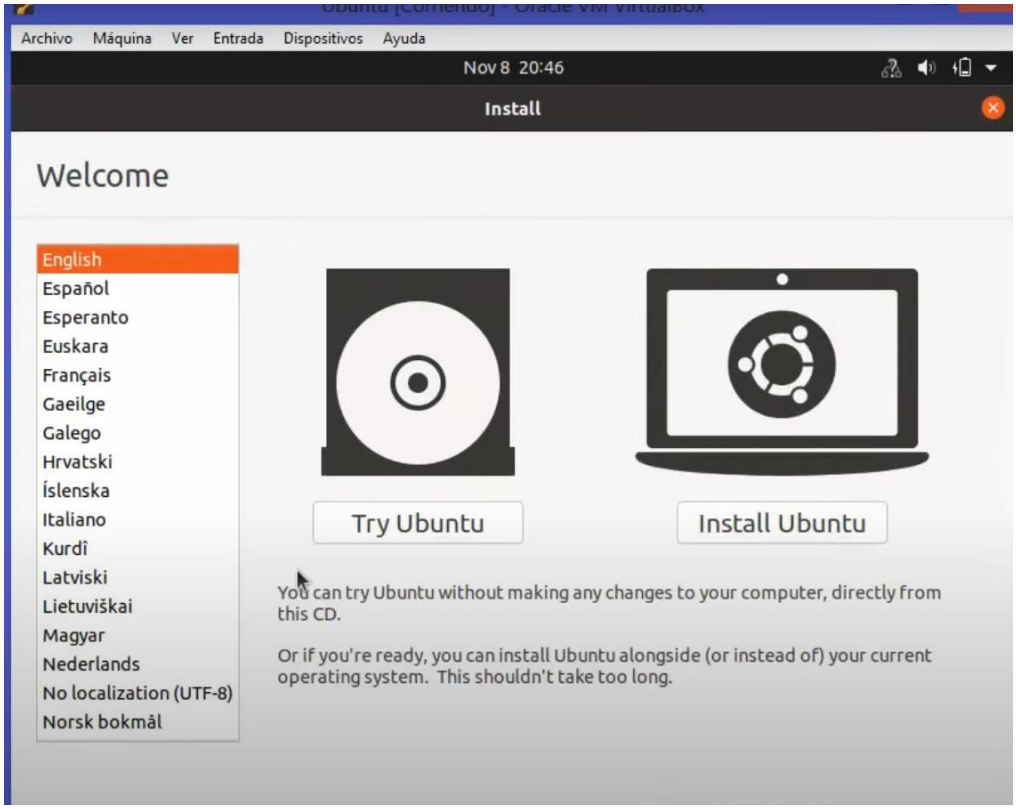
Anexo 6. Agregar Imagen virtual

Una vez configurada la maquina virtual se debe iniciar dando doble click en el nombre de la maquina creada.

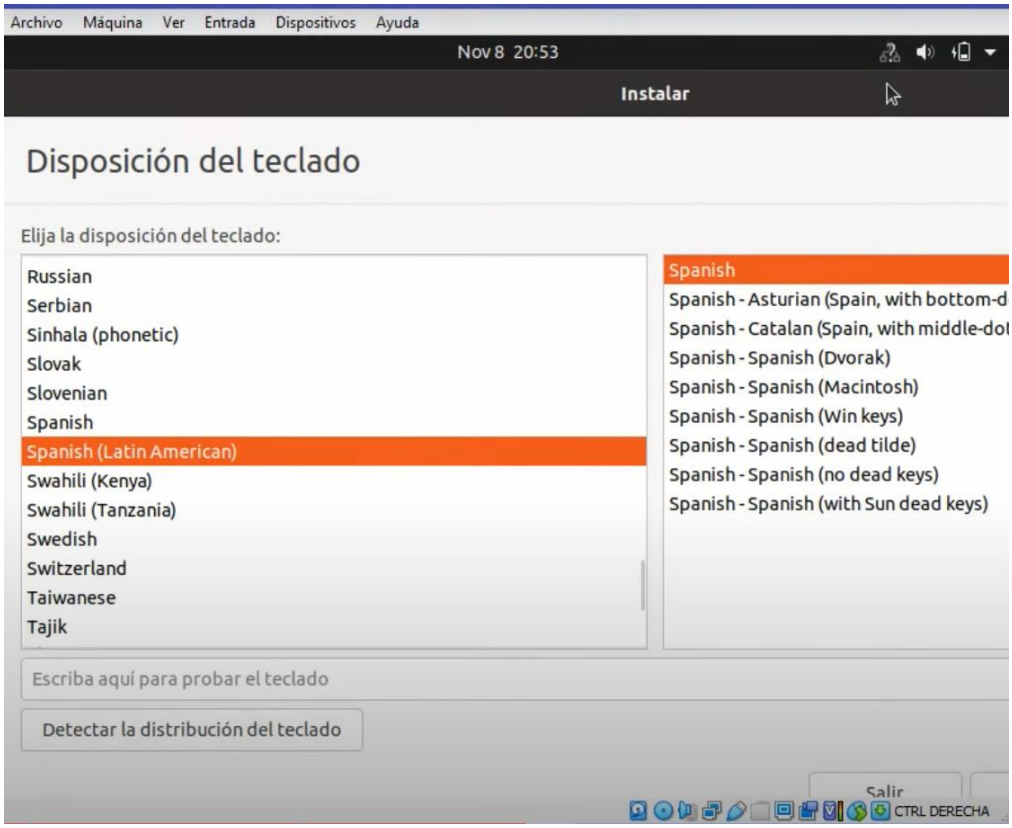


Anexo 7. Iniciar máquina virtual.

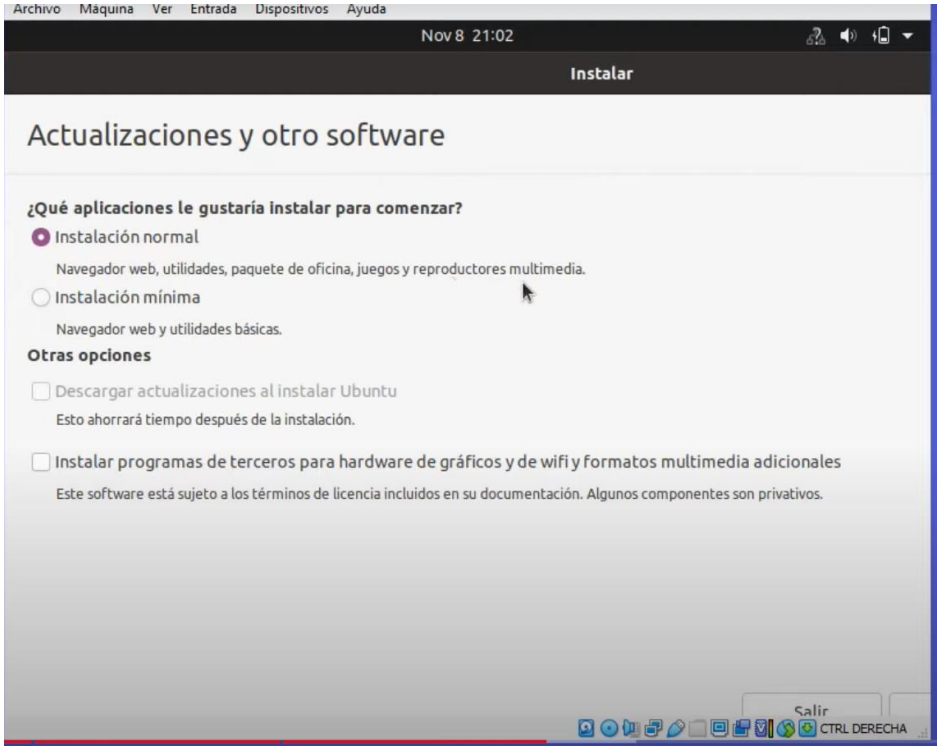
Al momento de inicializar, aparecerá una pantalla donde se pedirá que se elija el idioma y se instale Ubuntu.



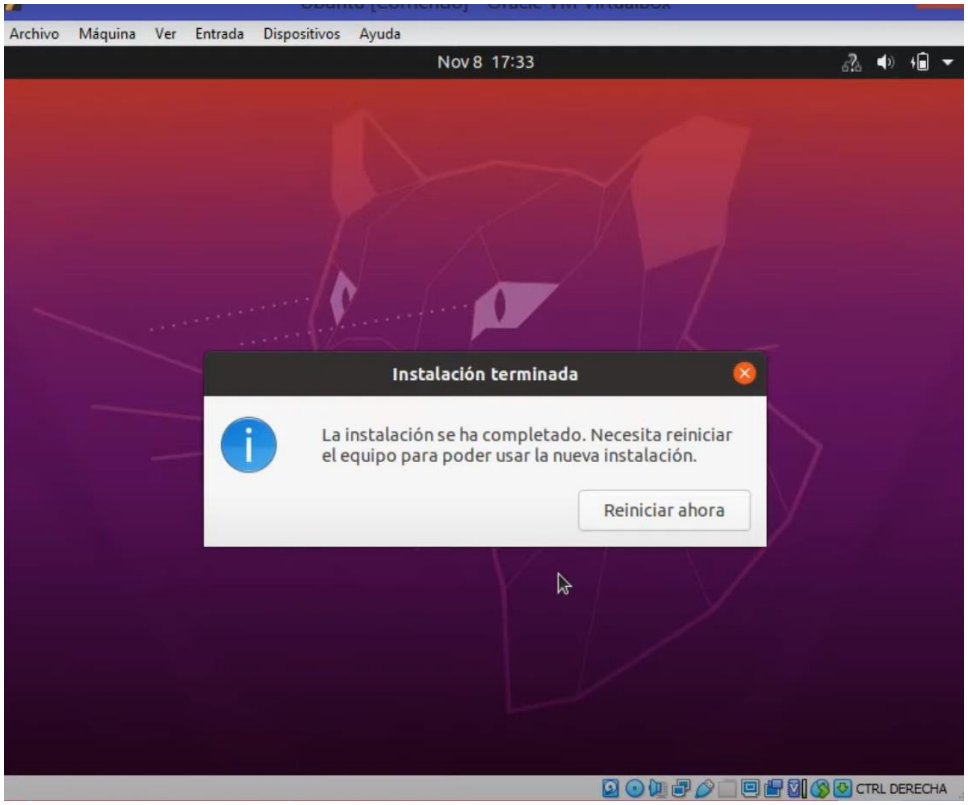
Anexo 8. Instalación de Ubuntu.



Anexo 9 Selección del idioma del teclado.



Anexo 10. Instalación normal



Anexo 11. Instalación completa.