ANÁLISIS COMPARATIVO DE INTERCAMBIO DE PAQUETES ENTRE REDES DEFINIDAS POR SOFTWARE Y REDES TRADICIONALES, APOYADO EN LA HERRAMIENTA DE SIMULACIÓN MININET

Presentado por:
Julián David Díaz Torres

Universidad Santo Tomás Colombia
División de Ingenierías
Ingeniería de Telecomunicaciones
Bogotá D.C.
2020

ANÁLISIS COMPARATIVO DE INTERCAMBIO DE PAQUETES ENTRE REDES DEFINIDAS POR SOFTWARE Y REDES TRADICIONALES, APOYADO EN LA HERRAMIENTA DE SIMULACIÓN MININET

Presentado por:
Julián David Díaz Torres

Asesor de proyecto:
Ing. Pedro Alejandro Mancera Lagos

Documento de Monografía para optar por el título en Ingeniería de Telecomunicaciones

Universidad Santo Tomás Colombia
División de ingenierías
Ingeniería de Telecomunicaciones
Bogotá D.C.
2020

Nota de aceptación

_____
_____
_____
_____
_____
_____

Ing. Gustavo Alonso Chica Pedraza
Jurado

Bogotá D.C., Agosto 20/2020
Ciudad y fecha de entrega

Agradecimientos

Agradezco a Dios por permitirme alcanzar este objetivo, a mi familia en general, quienes estuvieron apoyándome durante toda esta trayectoria, a los docentes de la academia por acompañarme con paciencia y perseverancia de este sueño y a mí raza llanera, de donde salí un día convencido que podría esta meta lograr.

Resumen

El siguiente documento monográfico se objeta con el fin de brindar una amplia visión el cual representa las Redes Definidas por Software (SDN) y su interacción hacia el futuro de las redes programables, brindando aquellos conceptos frescos que día a día obtiene más fuerza en las áreas de las telecomunicaciones y su arquitectura de datos. También se investiga sobre Openflow su origen y evolución. Del mismo modo, se busca realizar un ejercicio comparativo con un fin educativo y práctico para analizar la temática en el software Mininet: Emulador de redes virtuales donde permite visualizar diversos aspectos teóricos de Openflow y SDN de forma aplicada. Se ejecuta la simulación con el fin de reconocer el valor que tiene el controlador de esta novedosa arquitectura. Se apoya con la interfaz gráfica de mininet denominada Miniedit para lograr constituir una arquitectura gráfica base, donde se realiza las funciones de un controlador de capa 2 y se brinda generar un ecosistema óptimo para este documento. Por otro lado, se realiza un montaje de una red en Packet tracer con el fin de realizar la comparación directa y la viabilidad de migración que podría existir cuando se tiene una red tradicional y se quiere avanzar hacia las redes SDN.

Actualmente las redes de telecomunicaciones requieren innovación debido a la saturación y rigidez de su infraestructura, es por ello que las SDN son consideradas como una propuesta de solución innovadora con un gran campo de investigación hacia una nueva generación en las telecomunicaciones, En este documento se objetó un análisis comparativo entre las simulaciones en redes tradicionales y las definidas por software, donde se evidencia la optimización en el tiempo de respuesta sobre las redes SDN, incluso en las redes de área amplia, se caracterizó el tamaño de los paquetes en cada una de las pruebas de transmisión de datos, resaltando la diferencia superior en el tamaño del paquete de Mininet y su ventaja en velocidad garantizando un aumento en la disponibilidad de la red, frente a la red tradicional. Se analizó el ancho de banda determinando la gran variabilidad que presenta en las redes tradicionales, mientras SDN genera una estandarización sobre el tiempo en la transmisión de los paquetes permitiendo un ahorro de recursos.

Abstract

The following monographic document is objected in order to provide a broad vision which represents Software Defined Networks (SDN) and their interaction towards the future of programmable networks, providing those fresh concepts that day by day obtain more strength in the areas of telecommunications and its data architecture. Openflow is also investigated for its origin and evolution. In the same way, it seeks to carry out a comparative exercise with an educational and practical purpose to analyze the subject in the Mininet software: Virtual network emulator where it allows to visualize various theoretical aspects of Openflow and SDN in an applied way. The simulation is executed in order to recognize the value that the controller has of this novel architecture. It is supported by the mininet graphical interface called Miniedit to achieve a base graphical architecture, where it performs the functions of a layer 2 controller and provides an optimal ecosystem for this document. On the other hand, a network is assembled in Packet tracer in order to make a direct comparison and the feasibility of migration that could exist when you have a traditional network and you want to move towards SDN networks.

Currently, telecommunications networks require innovation due to the saturation and rigidity of their infrastructure, which is why SDN is considered as an innovative solution proposal with a large field of research towards a new generation in telecommunications. This document objected a comparative analysis between simulations in traditional networks and those defined by software, where the optimization in the response time over SDN networks is evidenced, even in wide area networks, the size of the packets in each of the data transmission tests, highlighting the superior difference in the Mininet packet size and its speed advantage, guaranteeing an increase in network availability, compared to the traditional network. The bandwidth was analyzed determining the great variability that it presents in traditional networks, while SDN generates a standardization on the time in the transmission of the packets allowing a saving of resources.

Contenido

Índice de imágenes **Pág.**

Índice de tablas

Capítulo 1: Problemática

1. Introducción

Desde los inicios de internet en los años 60, se ha venido consolidando la importancia de las telecomunicaciones en el mundo, y su crecimiento desde entonces ha sido exponencial, tanto así que el protocolo que actualmente utilizamos (TCP/IP) es reconocido como la base del internet. Época donde comenzaron a surgir diversas tecnologías para la transmisión de datos, aunque es este protocolo el que nos permite comunicar computadores con sistemas operativos diferentes. debemos reconocer que existen distintos centros de investigación tecnológica el cual se encarga de crear diferentes prototipos de protocolos de comunicación, objetando una función específica para cada uno de ellos, en este caso, el protocolo que debemos reconocer es el protocolo Openflow desempeñado en las redes definidas por software.

Openflow se considera como un protocolo emergente y en formato abierto para las comunicaciones sobre las redes definidas por software, donde su función principal es conectar un controlador con los diferentes dispositivos de Switching, definiendo los respectivos caminos de enrutamiento para la gestión del reenvío de paquetes. Este estándar surge gracias a la Open Networking Foundation, por medio de este protocolo una red puede ser administrada en su totalidad, esto se debe a la descentralización de las capas de control y de datos que conocemos normalmente sobre el modelo ISO/OSI, y la principal funcionalidad es brindar la conectividad entre controlador y los diferentes elementos de red.

Comúnmente encontramos las 7 capas del modelo ISO/OSI: (Física, Enlace de datos, Red, Transporte, Sesión y Presentación), modelo donde gran cantidad de fabricantes líderes en el mercado de las telecomunicaciones dominan imponentemente, aplicado como un marco de referenciarían encargado de guiar la estructuración de un sistema de interconexión para las telecomunicaciones a nivel global, cada capa cuenta con diversos protocolos de comunicación, que pueden encontrarse a lo largo del diseño y desarrollo de la infraestructura de red. A partir de este modelo, para se propone tres capas sobre su arquitectura: (infraestructura, control, aplicación) al desarrollo no centramos sobre la capa de infraestructura y control con el fin de entender con más claridad esta nueva tendencia, ya que la capa de aplicación puede ser empleada a comodidad de los diferentes fabricantes y/o desarrolladores.

2. Descripción del problema

El sector empresarial actualmente está considerando la migración hacia soluciones en la nube, y este proceso también se está viendo enfocado en su área de infraestructura y gestión de redes donde la definición de redes definidas por software (SDN) la empiezan a proyectar a un futuro no muy lejano como una solución tecnológica que permite mejorar aspectos como el rendimiento y la optimización de costos al tener la posibilidad de gestionar de forma inteligente los recursos de la red.

De este modo, comienza la caracterización de las diferentes problemáticas que pueden llegar a existir al buscar una migración de redes tradicionales a redes SDN, esto debido a los altos costos de inversión que requieren cada uno de los equipos ya comprados, y que no han solventado su utilidad, como también el posible sobrecosto ò pérdida que puedan afrontar al realizar este cambio de arquitectura tecnológica. Por ello, es importante aportar un punto de vista sobre la posible posición que pueda tener una empresa al encontrarse entre estas dos tecnologías.

La seguridad cibernética es uno de los factores más importantes para una empresa donde su capital se centra en la infraestructura de red y la valorización de la información, es por ello que día a día las compañías tecnológicas le apuestan tanto a desarrollo de aplicativos que le brinden seguridad, la definición de políticas, generación de parámetros de red y garantizar una confiabilidad son los aspectos que se buscan en una red actualmente.

Del mismo modo, En el mundo de la tecnología naces proyectos innovadores así como también hay tecnologías que van quedando totalmente obsoletas, cada proyecto nace con un objetivo en especial, ¿existen la viabilidad de migrar una red tecnológica tradicional a una red definida por software garantizando su topología de red, brindando ventajas operativas funcionales y permita flexibilidad en la innovación para no ser obsoleta a mediano plazo?

3. Objetivos:

3.1 Objetivo general

Desarrollar un análisis comparativo sobre los principales parámetros de desempeño de las Redes Definidas por Software - SDN a través del software de simulación Mini net, y las redes tradicionales a través del software de simulación Packet tracer,

3.2 Objetivos específicos

- Especificar los fundamentos de las redes definidas por software, su aplicabilidad y los principales parámetros de desempeño a nivel de intercambio de paquetes.
- Especificar los principales parámetros de desempeño de las redes tradicionales (No definidas por Software)
- Proponer dos topologías de red que permitan obtener datos del rendimiento en el intercambio de paquetes en los simuladores seleccionados.

- Analizar los resultados obtenidos en la simulación de las tipologías seleccionadas.

- Comparar la funcionalidad y desempeño bajo la información obtenida

4. Justificación

4.1 Justificación técnica

Se requiere esclarecer la viabilidad existente entra la migración de una red tradicional a una red de tecnología avanzada como SDN, con el fin de lograr optimizar recursos de red, garantizar desempeño y seguridad sobre una infraestructura tecnológica innovadora, donde permita generar una conectividad con un centro de control total, de forma completa con el fin de brindar la prestación de diversos servicios, analizando las posibles soluciones a los diferentes tipos de desarrollos tecnológicos y protección de ataques que puedan ser recibido este protocolo. Esta investigación busca brindar garantías para un diseño, desarrollo y migración de red el cual pueda ser implementado en diferentes escenarios académicos donde se requiera conocer las ventajas y desventajas de implementar una red SDN. Destacando la importancia de la innovación y la seguridad de las redes.

4.2 Justificación académica

El protocolo Openflow debe ser instruido de una forma dinámica, clara y concisa con el fin de facilitar a la comunidad académica la captación del funcionamiento y aplicación de este protocolo y el desarrollo del tipo de tecnología que la aplica, debido a que apunta ser uno de los principales protocolos de encaminamiento entre switchs y servidor o controlador aplicado sobre una red, por ende se encontrará con dicho protocolo en las diferentes áreas del desarrollo de infraestructura de red SDN, por su capacidad, alcance y programabilidad, este protocolo es uno de los más robustos y escalables que podremos encontrar, por ello es de suma importancia que los estudiantes conozcamos claramente la funcionalidad, robustez e importancia que genera su aplicabilidad en el mundo , adicionalmente por su capacidad de evolución que presenta para ser la base de nuevas propuestas de comunicación.

5. Estado del arte

Las redes tradicionales en la actualidad se están volviendo obsoletas debido a la gran congestión que provoca el flujo de la información, destacando la cantidad de procesos que deben afrontar el mensaje al trasladarse desde un emisor a un receptor, cumpliendo con una cantidad de complejos protocolos que conforman la red de extremo a extremo, del mismo modo, se puede evidenciar los diferentes procesos que existen en la administración de una red sobre el modelo ISO/OSI ya que el procesamiento de cada paquete está designado por la programación que cuenta cada equipo de la red, caso opuesto a lo que propone las redes SDN con Openflow nace gracias a la fundación de Open Networking Foundation (ONF), con el propósito de generar la estandarización de las tecnologías emergentes con el fin de impulsar este software como modelo tecnológico competitivo sobre la gestión de las redes de centros de datos y demás propuestas de redes[7].

A diario se despliegan redes de telecomunicaciones en el mundo pero han llegado a un punto donde estas redes tienden a volverse obsoletas por lo estáticas y complejas que pueden llegar a ser y esto involucra a cada uno de los equipos que interconecta y cantidad de reglas, políticas de administración y de control que maneja el tráfico de paquetes. La rigidez que poseen las redes de hoy incrementa la complejidad del desarrollo de tecnologías innovadoras, capacidad para el despliegue de nuevos servicios, brindar un dinamismo frente a los nuevos productos y servicios tecnológicos, el EOT con una gran cantidad de dispositivos y sensores que recolectar información en tiempo real y futuras creaciones que pueda girar en torno a la necesidad de comunicación del ser humano [7].

Debe resaltar la cantidad de servicios que se presentan en la nube, la cantidad de dispositivos inteligentes y escritorios remotos que se crean a diario en el mundo, buscando valorizar o trasladar el foco económico que gira en el entorno de la Tecnología, la información y las comunicaciones, donde se reconoce la perdida de costo en producción de hardware y software por el determinado uso de código abierto, y la entrada en el mercado de hardware sin marca ni características adicionales, esto ha permitido aumentar la productividad de los proveedores en el entorno de la prestación de productos y servicios como: instalación, programación, ejecución y migración de funciones y aplicaciones. Esta proyección nos dirigirá a un entorno futuro rodeado de redes virtualizadas y un proceso de “Softwarización” que día a día se va reconociendo donde se valorizará mediante la optimización de procesos [9].

Desde la adopción de datacenters en las compañías, con la adquisición de los diferentes equipos tecnológicos de comunicaciones, se vio la necesidad de crear y optimizar una red que se encontrara bajo su propio dominio de administración. Desde estos mismos centros los firewalls, los PBX, balanceadores de carga y demás hardware que realizaban una función específica sobre la red, ha venido siendo migradas a softwares desarrollados ejecutados sobre servidores, incluyendo la evolución de los routers y los switchs con el fin de simplificar y optimizar sus funciones, brindando ahorro de energía y demás reducción de costo que se pueda adquirir. Todo este proceso que se ha venido generando desde una pequeña empresa hasta una multinacional sobre su propia infraestructura de red, se proyecta a estar bajo el control de un software o un conjunto de software del plano de control. Lo que se considera el origen de las redes definidas por softwares, permitiendo efectividad en la operación de la red [12].

5.1 Evolución

Los constantes avances tecnológicos de los últimos años proyectan hacia el aumento de los diferentes servicios y contenido que se requiere por los diferentes clientes y/o usuarios de las telecomunicaciones. Dada dicha evolución tecnología, logró aumentar la dificultad de los procesos en las redes, desde sus parámetros hasta su administración donde el nivel de entendimiento para el manejo y desarrollo de nuevas soluciones sobre la infraestructura existente se vuelve nula, y es por ello que surge la necesidad de migrar a un tipo de tecnología que permita un dinamismo y una programabilidad más flexible cumpliendo con los parámetros y políticas anteriores, brindando la oportunidad de seguir evolucionando [12].

Se presenta un modelo de red tradicional en la imagen 1, con el fin de realizar un análisis comparativo de la funcionalidad práctica transicional hacia SDN, en la imagen 2 se observan cómo se desacopla físicamente el plano de control, del plano de datos, donde se asigna un controlador el cual se encarga de realizar la tarea de la conmutación, esta arquitectura permite una mayor velocidad, control y nivel de gestión en los equipos de la red, donde permite aumentar la interacción y comodidad sobre los cambios de funcionalidades en la red en general.

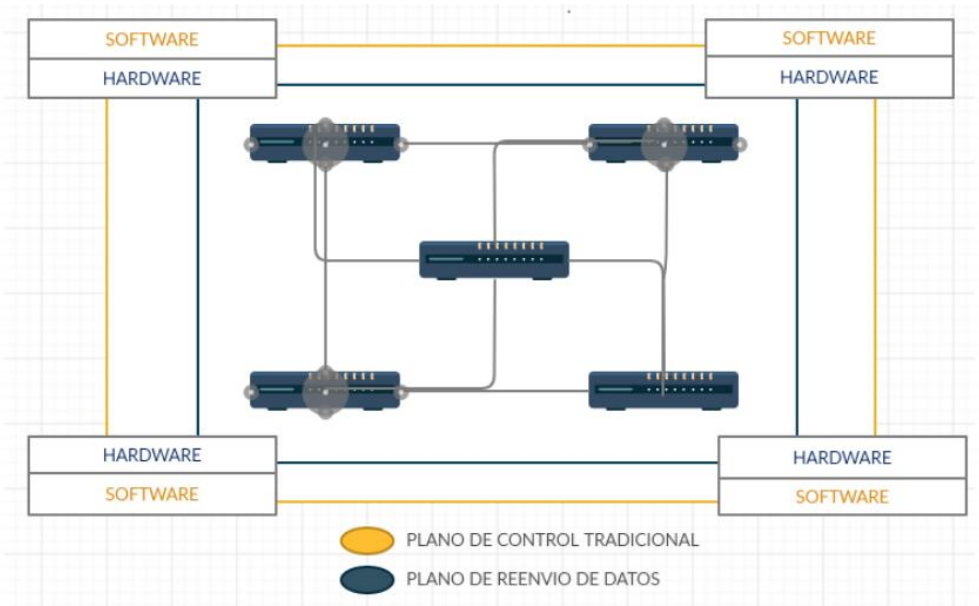


Imagen 1: Red tradicional: Plano de datos y control

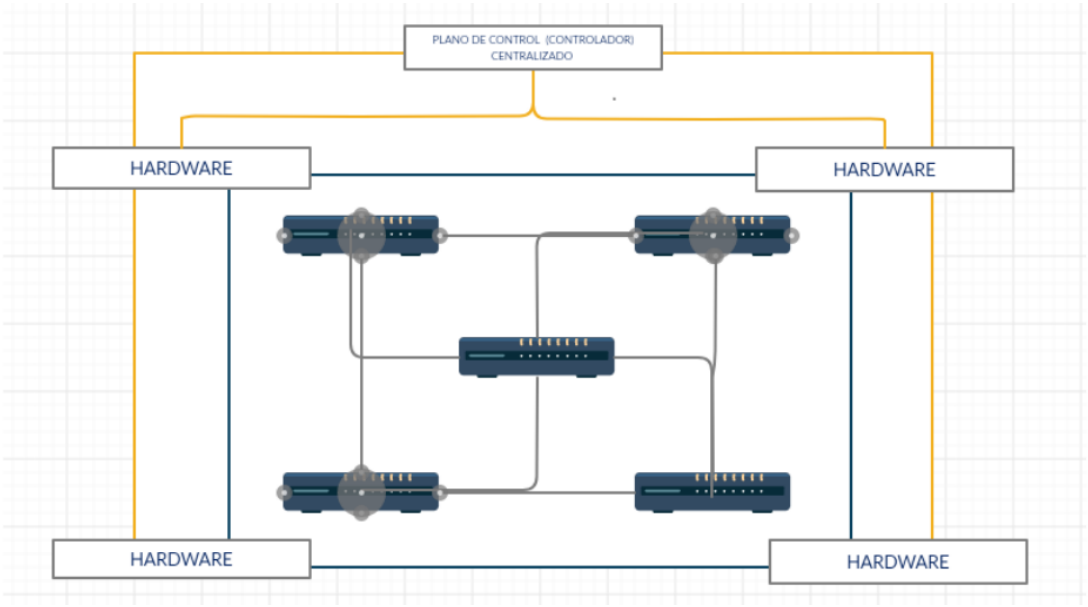


Imagen 2: Red SDN: Plano de datos y control descentralizada

5.2 Fortalezas

La propuesta tecnológica se presenta con diversos software de emulación y ejecución según el estilo de los fabricantes, esto se debe a la creación de la ONF donde gran cantidad de empresas del sector de las telecomunicaciones, programadores y desarrolladores de aplicativos se han unido a esta organización, con el fin de estandarizar un protocolo accesible, y virtualizado donde exista un lenguaje de comunicación en común para cada uno de los desarrolladores de los controladores y aplicativos de la red SDN [13].

5.3 Debilidades

Desafortunadamente el protocolo Openflow se encuentra en investigación con pocas versiones ya en el mercado acadèmico el cual nos genera reglas y condiciones que limitan su flexibilidad a la hora de ser aplicado, esto se debe a que gran cantidad de equipos no pueden ser homologables con el protocolo Openflow y de debe esperar a que cumplan con su periodo de vida útil para ser reemplazados [17].

6. MARCO TEÓRICO

Redes definidas por software: (SDN: Software Defined Network) Es un concepto prácticamente novedoso, que surge bajo la necesidad de dar una solución avanzada a las peticiones requeridas por las redes que poseen una masiva transitividad y que se están quedando cortas a la hora de brindar una ruta rápida de interconectividad, esto se debe adicionalmente a la complejidad que poseen debido a su crecimiento para buscar una mayor capacidad, y esto funciona bajo la ejecución de la red de transporte y la de control en un mismo dispositivo, lo que hace compleja su gestión y causa que el equipo se reporte como obsoleto, por otro lado debemos destacar que cada equipo es programado de forma independiente bajo las especificaciones de cada fabricante lo que aumenta su complejidad a la hora de gestionar fallas en tiempo real [1].

Capa de control: Se objeta la capa de control como controlador de red, donde lo asimilamos al sistema de operación de la red, el cual mantiene una vista centralizada por el plano de control hacia la red donde se encarga de generar las tomas de decisiones más aptas para el reenvío de las tramas de información analizando cada uno de los estados que anuncian los dispositivos de la red que se encuentran conectados, Los dispositivos mantienen su comunicación con el controlador por medio de una interfaz denominada South Bandle openflow, adicionalmente se cuenta con una interfaz de programación aplicada por medio de la interfaz northbound el cual permite el intercambio de datos sobre los servicios de red y las aplicaciones de negocio [2].

Capa de aplicación: En esta área, la red se permite manipular de forma única, a como se desee proveer el servicio a los clientes, para ello es necesario de una interfaz de programación de aplicaciones API, donde los proveedores de servicios de internet o los operadores de red cuenten con su propia API, normalmente estas API pueden ser creadas por desarrolles terceros a los ISP, aunque aún no existe la oficialización de un estándar que defina su estructuración por la ONF (Open Networking Foundation), sin embargo es muy requerida para lograr interoperabilidad garantizada entre los diferentes business applications como también con los controladores de red [2].

Interfaz de programación de aplicaciones: API es un grupo de protocolos y definiciones enfocadas en el desarrollo integral de software de aplicativos, busca simplificar procesos de conectividad generando ahorro en tiempo y dinero. Las API simplifican el diseño, brinda flexibilidad cómodas formas de administración y uso de las aplicaciones para optimizar el servicio de la red. Se reconoce que las API brindan una colaboración entre el equipo comercial y el de TI, esto busca brindar una respuesta rápida a las necesidades comerciales dentro del mercado digital ya que este suele estar en constante cambio [3].

Openflow: Tecnología derivada de una investigación desarrollada sobre redes virtuales natal de la universidad de Stanford, creada en la ONF (Open Networking Foundation) con el fin de regular, promocionar y expandir el uso de Openflow en las diferentes redes empresariales. Así como Openflow en la actualidad existen otros tipos de controladores, donde por medio de la ONF difunden esta y otras tecnologías bien relacionadas con las redes definidas por software. Protocolo centralizador de decisiones, el cual permite conectar un servidor con uno o varios grupos de

conmutadores señalando la ruta para el traslado de paquetes sobre una red convencional, las redes convencionales normalmente cuentan con conmutadores que poseen su propio software que analiza la situación, al ejecutar openflow como protocolo, se genera un centro de decisiones aplicado para la migración de paquetes, donde permite programar de forma independiente los conmutadores y diferentes equipos sobre centros de datos [4].

ISP: se define como empresa encargada de prestar el servicio de acceso a la internet, existe una clasificación sobre los proveedores de dicho servicio; servicio de almacenamiento hosting y/o caché, contenido, red, motores de búsqueda, enlaces peer to peer y enlaces [5].

Tecnologías para Virtualización de las redes

Las redes virtuales es una definición que día a día toma mayor fuerza para aplicarse en las grandes organizaciones que cuentan con un departamento de Ti, e incluso las empresas, la inteligencia de esta tecnología se conserva en el software y no en el hardware individual que conforman la red, esta combinación de software con hardware de red como los enrutadores y switchs permite una configuración y administración avanzada.

VLAN Red Virtual de Área Local

Una Red virtual de Área local se reconoce como un método utilizado para generar redes lógicas independientes sobre una misma infraestructura de red, pueden existir varias redes virtuales sobre un hardware o en una única red física, este método permite deshacernos de varias limitaciones con las que se cuenta en la infraestructura física de direccionamiento o de geografía, porque nos permite segmentar de forma lógica basándose en la agrupación de equipos teniendo en cuenta los criterios que se quieran tener en cuenta, como por ejemplo: protocolo, número de puertos y hasta direcciones Mac. Las vlans se encuentran definidas bajo estándares IEEE 802.1D, 802.1P, 802.1Q, 802.10 [11].

Tipos de vlans

Vlan por puerto (Nivel 1): normalmente es conocida como Port Switching, se selecciona el puerto del dispositivo switchs y se le asigna la vlan que se requiere asociar, de ese modo solo los miembros de dicha Vlan son los que se conectan a ese puerto [11].

Vlan por direcciones Mac (Nivel 2): Desde el dispositivo de conmutación o switchs se configuran todos los host que se requieran conectar a una vlan con su dirección Mac, este tipo de vlan brinda la ventaja de que no se debe aferrar a un puerto específico, y que el usuario puede cambiar su ubicación sin problema [11].

Vlan por tipos de protocolos (Nivel 2): Se configura de tal forma que analiza el contenido de la trama Mac en el campo donde señala el tipo de protocolo

Vlan por dirección de subred (Nivel 3): la subred virtual donde se reconoce por la cabecera, se utiliza para rastrear a que vlan pertenece y son los paquetes quienes pertenecen a las vlan y no las estaciones como los anteriores tipos.

Vlan nivel superior: Sobre este tipo de Vlan se permite configurar con dos o más factores dependientes como número de puerto, dirección Mac, horario de conexión la forma de acceso, y otras más.

VPN Redes Privadas Virtuales

Las redes virtuales privadas es una tecnología de la red que permite extender una red local a través de una red pública o no controlada de forma segura. Esta tecnología permite un intercambio de datos sobre redes compartidas o públicas de una forma privada con varias funcionalidades como seguridad y políticas para la gestión de redes privadas [11]

Tipos de vpn

Punto a punto

El servidor de redes virtuales privadas cuenta con un vínculo fijo a internet y establece conexiones vía internet que provengan de sitios diferentes y genera un túnel VPN. Este procedimiento permite eliminar los diferentes vínculos punto a punto que tradicionalmente se realizaban de forma física entre nodos y ayudan a economizar costos

Acceso remoto

El acceso remoto permite que los usuarios se conecten de forma remota por medio de internet con un vínculo de acceso, una vez se autentique el sistema le otorga un nivel de acceso como si estuviese en su sitio de trabajo (host) Es un método usado de forma común en la actualidad [11].

Tunneling

Esta tecnología consiste en generar un encapsulamiento de un protocolo de red sobre otro protocolo, brindando un túnel dentro de una red de computadoras, al establecer este túnel se implementa una unidad de datos de protocolo dentro de otra unidad igual, esto con el fin de transmitir extremo a extremo la información sin necesidad de desempaquetar la información viajera en interior [11].

VPN over LAN

Esta aplicación no es muy conocida pero es realmente útil, ya que permite aislar zonas y servicios en la misma red interna de la empresa, básicamente consiste en establecer redes privadas virtuales sobre una red local, y esto permite aumentar la seguridad para el acceso inalámbrico, dividiéndolas con el fin de evitar extracción de la información o un acceso no autorizado [11].

AN y API Redes Activas y programables

Las redes activas y programables están enfocadas hacia el control de red, donde por medio de una interfaz de programación de aplicaciones se formula la optimización de los recursos como: procesamiento, colas de paquetes, almacenamiento y demás. De forma individual sobre los nodos que conllevan la construcción de las funciones

personalizadas para ser ejecutadas en un subconjunto de paquetes que pasan por medio del nodo [11].

Redes superpuestas

Las redes overlay o superpuestas es una red virtual con nodos lógicamente enlazados, donde cuenta con una o más redes construidas de forma subyacente, busca implementar servicios de red que no están disponibles en la red subyacente, se puede aplicar la red superpuesta a medida que se proporcione servicios de capa superior [11].

Ancho de banda:

El ancho de banda es la tasa de bits que representa la cantidad de información capaz de transferirse entre dos puntos en una red sobre un determinado tiempo, el ancho de banda es representado en bits por segundo y es catalogado como tasa de bits, también puede ser empleado para denotar el ancho de banda consumido o empleado sobre una transmisión y se encuentra denotado como Throughput: definido como la medida de transferencia de datos entregados de forma exitosa sobre un enlace de comunicación.[9]

TTL

El tiempo de vida de un paquete es un indicador el cual se encarga de señalar la cantidad de nodos por los que pasa, descontando en cada salto una unidad hasta llegar a 0 donde será descartado, es de aclarar que el valor inicial de TTL lo indica el emisor de la red, donde condiciona la distancia a la que puede llegar el paquete sobre la red. [9]

Latencia

Se considera como latencia a la suma de los retardos temporales que se generan sobre una red, nacen a raíz de la demora de en la transmisión de los paquetes dentro de la red, también influyen los tamaños del paquete. [9]

Capítulo 2: Fundamentos, aplicación y parámetros de desempeño sobre el software emulador de redes: Mininet

1. Introducción

La creación de nuevas tecnologías cuentan con un proceso investigativo a lo largo del tiempo, y muchos de ellos van evolucionando poco a poco gracias al desarrollo de la ciencia, con base en lo anterior el uso de diversos entornos de simulación se convierten en instrumentos esenciales para el desarrollo de cualquier proyecto sin importar el grado de dificultad en las áreas de la ciencia y la ingeniería, por medio de las simulaciones podemos realizar un estudio previo de los comportamientos a ocurrir en la implementación real, claramente no arrojará con exactitud los resultados pero es una condición muy similar. Este tipo de prácticas nos permite depurar y detectar errores de nivel superior con el fin de minimizar errores en el despliegue. Al estar en la academia se evidencia la importancia de realizar esta actividad ya que los recursos son muy limitados y se debe manejar una investigación académica a muy bajo costo. Singularmente en este documento monográfico se muestra un proceso de desarrollo investigativo sobre el área de las telecomunicaciones, empleando un hardware y un software específico.

Mininet es un software de emulación de redes donde nos permite crear una red virtual de switchs, host y controladores en la categoría de redes definidas por software de forma rápida, sencilla de entender sus comandos y con la particularidad de ejecutar un emulador.

El simulador es básicamente el dispositivo o aparato que ejecuta la reproducción de un sistema, donde se experimenta lo que muy probablemente puede llegar a suceder en la realidad mientras que el emulador es un software que permite la ejecución de programas o sistemas codificados, en plataformas diferentes a la que fueron escritos en su originalidad. La diferencia con el dispositivo real se encuentra en el hardware sobre el cual se trabaja.

2. simuladores y emuladores SDN

Para llevar a cabo el desarrollo de un proyecto sobre las redes definidas por software, se tiene presente que existen 4 simuladores o emuladores de topologías el cual prueba y evalúa los protocolos y las aplicaciones diseñadas. Mininet: Emulador diseñado para desenvolverse con la tecnología Openflow y está dirigido hacia un entorno de creación de prototipos de red en una forma fácil y rápida [18]. NS-3: Es un simulador de redes el cual cuenta con una nueva funcionalidad y consiste en simular dispositivos Openflow, se centra en ofrecer un sistema más cercano a la realidad con una escalabilidad similar a la de mininet, aunque en la ejecución del prototipo es más lento [19]. Estinet: Simulador y emulador de redes definidas por software con soporte necesario para Openflow y su valor agregado es que a diferencia que Mininet y NS-3 Estinet dispone de un entorno gráfico, su principal funcionalidad es mostrar de forma gráfica el comportamiento de la red [20]. STS: es un simulador de red centralizado en generar entornos de prueba y examinación de estados de red de una forma interactiva, su objetivo principal es hallar y solucionar problemáticas a fin. [21] para llevar a cabo este documento se deben caracterizar las principales habilidades como eficaz forma de absorción de información, Buen banco de gestión de información y

ejecución de pruebas rápidas razones por las cuales, se decide trabajar con mininet en este documento.

2.1 Requisitos e instalación en Ubuntu Linux

2.1.1 Requisitos:

Se debe tener en cuenta que para instalar Mininet, hay que cumplir con su único requisito el cual es contar con SO Linux, ya que sobre este software es que únicamente funciona, adicionalmente se debe corroborar que el sistema operativo cuente con los kerner actualizados o en una versión superior a la 2.6.33, del mismo modo mininet va a permitir una mejor distribución sobre Ubuntu.

Instalación:

Puede seguir estos 3 pasos para descargar e instalar el emulador en su máquina Ubuntu, aunque también existen métodos para instalarla como VM, para ello consulte la página oficial del desarrollador en la bibliografía [18]

1. Descarga de forma nativa desde la fuente con el comando: git clone git://github.com/mininet/mininet
2. Ejecutar mininet/util/install.sh [options: -a, -nfv, -s mydir]
3. Al completar la instalación, compruebe la funcionalidad básica de Mininet con el siguiente comando: sudo mn --test pingall

De este modo, [18]

3.1 Creación de la red

En este emulador existen varias formas para lograr crear una red definida por software o simplemente el diseño de una red, como lo son:

- por medio de la consola mn
- Por medio de aplicativo gráfico
- Por medio de API Python codificación

Se inicializa el entorno de mininet ingresando por la terminal de la máquina utilizando el siguiente comando:

- Sudo mn

Al inicializarse, automáticamente genera una topología por defecto, de este modo al crear una topología puede ser muy fácil y rápida, pero del mismo modo va a ser muy complejo cuando se requiera hacer una topología personalizada.

2.1.2 Creación, interacción y comandos usuales aplicables sobre mininet

Existen diversas formas de crear una topología de red sobre el software, puntualmente se mencionan dos (2), la primera consiste en crear una topología rápida, desde la misma consola o terminal de comandos sobre la máquina que tiene instalado mininet,

Para crear una topología rápida [22] se aplica primeramente sobre la línea de comandos, el lanzador que permitirá crear la instancia de la red (mn), adicionalmente con el parámetro Topo se llama al modelo de Topologías a formar,

Comando sudo mn:

```
root@juliandiaz-ThinkPad-L440:/home/juliandiaz# sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=11915>
<Host h2: h2-eth0:10.0.0.2 pid=11917>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=11922>
<Controller c0: 127.0.0.1:6633 pid=11908>
mininet>
```

Imagen 3: Creación rápida topología por defecto

Comando sudo mn --topo single, N: en este caso 'N' será 8 y representa la cantidad de host que necesitamos.

```
root@juliandiaz-ThinkPad-L440:/home/juliandiaz# sudo mn --topo single,8
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
h5 h5-eth0:s1-eth5
h6 h6-eth0:s1-eth6
h7 h7-eth0:s1-eth7
h8 h8-eth0:s1-eth8
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0 s1-eth5:
h5-eth0 s1-eth6:h6-eth0 s1-eth7:h7-eth0 s1-eth8:h8-eth0
c0
mininet>
```

Imagen 4: Creación rápida topología simple con un switchs y 8 hosts

Comando sudo mn --topo linear, N: donde n es la cantidad de host y switchs que se requieren.

```
root@juliandiaz-ThinkPad-L440:/home/juliandiaz# sudo mn --topo linear,3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
c0
mininet>
```

Imagen 5: Creación rápida topología lineal con tres switchs interconectados y 3 hosts

Comandos de Testing

Entiéndase equipos como dispositivo switchs o host dentro de las siguientes interpretaciones

>link [equipo1] [equipo2] [Estado a ejecutar: UP o Down]

Este comando permite interconectar dos equipos en caliente

```
mininet> link s2 s3 down
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
c0
mininet>
```

Imagen 6: Eliminación de conexión entre switchs s2 y s3

```
mininet> link s2 s3 up
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3
c0
mininet>
```

Imagen 7: generación de conexión entre switchs s2 y s3

> [Equipo 1] ping [equipo 2]
Este comando permite testear la interconexión entre equipo de origen y uno de destino

```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=64.5 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.465 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.063 ms
^C
--- 10.0.0.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2021ms
rtt min/avg/max/mdev = 0.063/21.707/64.593/30.325 ms
mininet> h1 ping s3
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.026 ms
^C
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.026/0.029/0.034/0.003 ms
```

Imagen 8: prueba de conectividad entre diferentes equipos de la res

> [equipo1] ping -c N [equipo2]

Adicionalmente se puede paramétrica el máximo de paquetes icmp a emitir para realizar la prueba de conexión agregando el comando -c y la cantidad de paquetes a definir

```
mininet> h1 ping -c 5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=20.0 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=13.1 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.515 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.130 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.067 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4034ms
rtt min/avg/max/mdev = 0.067/6.786/20.083/8.317 ms
mininet>
```

Imagen 9: prueba de conectividad desde el equipo h1 hasta el equipo h3 con un máximo de paquetes igual a 5

>pingpair
Este comando permite realizar un test de interconectividad entre una pajera de host de la red, normalmente toma la primera pareja de hosts creadas

```
mininet> h1 ping -c 5 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=20.0 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=13.1 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.515 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.130 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.067 ms

--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4034ms
rtt min/avg/max/mdev = 0.067/6.786/20.083/8.317 ms
mininet> pingpair
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

Imagen 10: prueba de conectividad entre pareja de equipos de la red.

>iperf

Este comando permite conocer la caracterización del ancho de banda dentro de la red entre dos equipos.

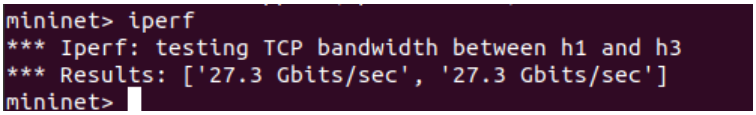


Imagen 11: prueba sobre la capacidad del ancho de banda desde h1 hasta h3.

>nodes
Este comando nos permite conocer todos los nodos existentes aplicados en la topología de la red

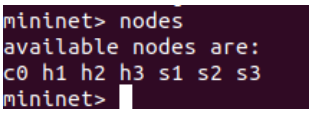


Imagen 12: Visualización de los nodos y/o equipos

>dump
Este comando permite conocer de forma detallada la composición de los nodos en la topología de la red

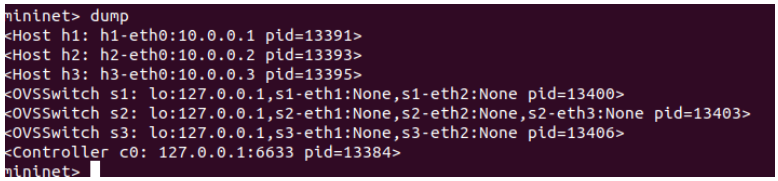


Imagen 13: Visualización de los nodos y/o equipos interconectados descrita con la interfaz

> Net
Este comando permite conocer cómo se encuentran conectados todos los equipos y switchs de la topología de red

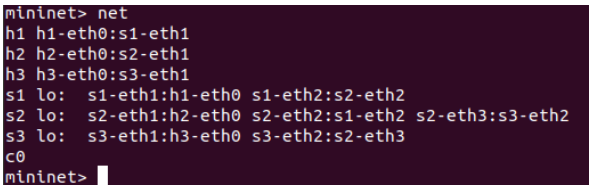


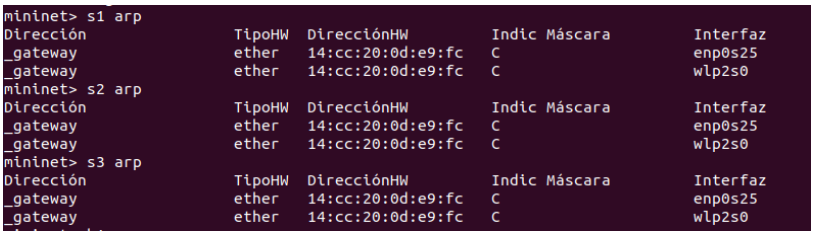
Imagen 14: Visualización de todos dispositivos de red

>EOF
Este comando permite detener todos los equipos de la red y cerrar la emulación

Imagen 15: Detención de los equipos y cierre de la emulación

>s# arp

Este comando permite ejecutar el protocolo arp (protocolo de resolución de direcciones)

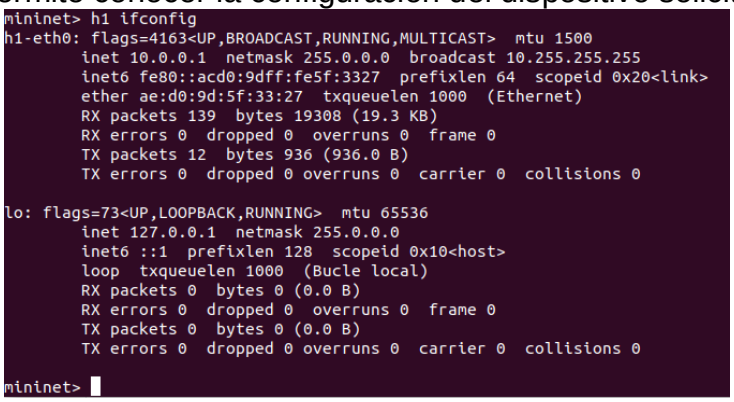


```
mininet> s1 arp
Dirección                TipoHW  DirecciónHW         Indic Máscara         Interfaz
_gateway                 ether   14:cc:20:0d:e9:fc   C                     enp0s25
_gateway                 ether   14:cc:20:0d:e9:fc   C                     wlp2s0
mininet> s2 arp
Dirección                TipoHW  DirecciónHW         Indic Máscara         Interfaz
_gateway                 ether   14:cc:20:0d:e9:fc   C                     enp0s25
_gateway                 ether   14:cc:20:0d:e9:fc   C                     wlp2s0
mininet> s3 arp
Dirección                TipoHW  DirecciónHW         Indic Máscara         Interfaz
_gateway                 ether   14:cc:20:0d:e9:fc   C                     enp0s25
_gateway                 ether   14:cc:20:0d:e9:fc   C                     wlp2s0
```

Imagen 16: arp con los 3 switchs de la topología de red

> (Host o switchs) ifconfig

Este comando permite conocer la configuración del dispositivo solicitado

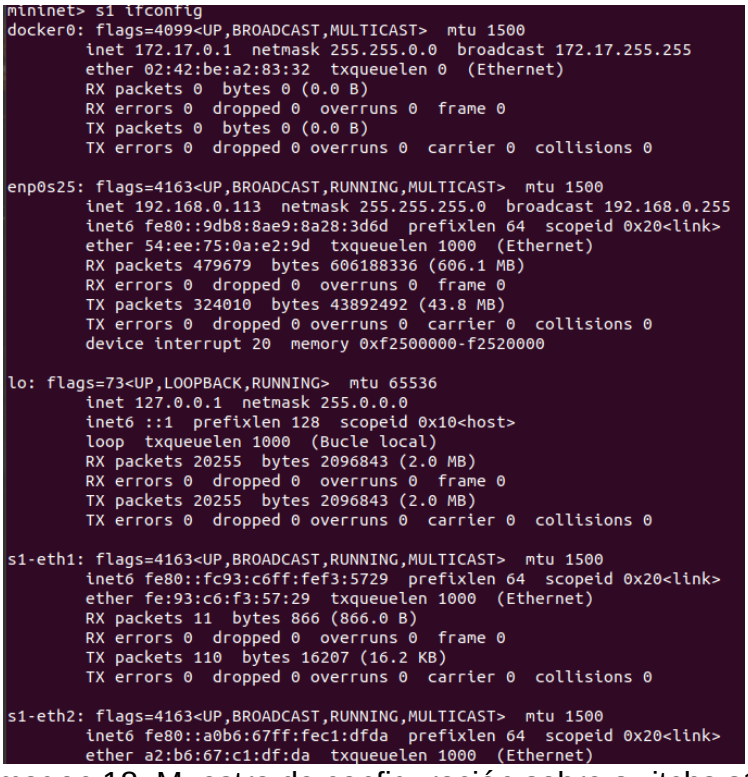


```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::acd0:9dff:fe5f:3327  prefixlen 64  scopeid 0x20<link>
        ether ae:d0:9d:5f:33:27  txqueuelen 1000  (Ethernet)
        RX packets 139  bytes 19308 (19.3 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 936 (936.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Bucle local)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet>
```

Imagen 17: Configuración en el host h1



```
mininet> s1 ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:be:a2:83:32  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s25: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.113  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::9db8:8ae9:8a28:3d6d  prefixlen 64  scopeid 0x20<link>
        ether 54:ee:75:0a:e2:9d  txqueuelen 1000  (Ethernet)
        RX packets 479679  bytes 606188336 (606.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 324010  bytes 43892492 (43.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 20  memory 0xf2500000-f2520000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Bucle local)
        RX packets 20255  bytes 2096843 (2.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20255  bytes 2096843 (2.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::fc93:c6ff:fef3:5729  prefixlen 64  scopeid 0x20<link>
        ether fe:93:c6:f3:57:29  txqueuelen 1000  (Ethernet)
        RX packets 11  bytes 866 (866.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 110  bytes 16207 (16.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::a0b6:67ff:fec1:dfda  prefixlen 64  scopeid 0x20<link>
        ether a2:b6:67:c1:df:da  txqueuelen 1000  (Ethernet)
```

Imagen 18: Muestra de configuración sobre switchs s1

> (Host o switchs) ps

Este comando permite conocer la lista de procesos sobre el dispositivo citado

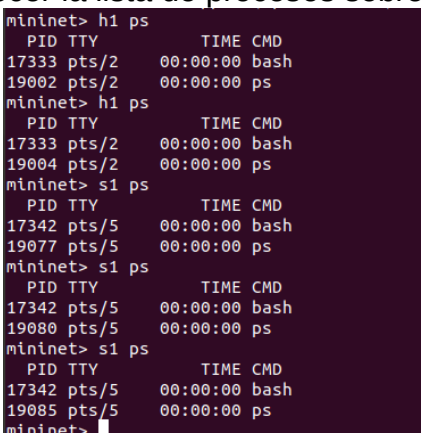


Imagen 19: Muestra la lista de procesos sobre el host y switchs () h1 y s1

>help
Este comando muestra todos los comandos aplicables en mininet

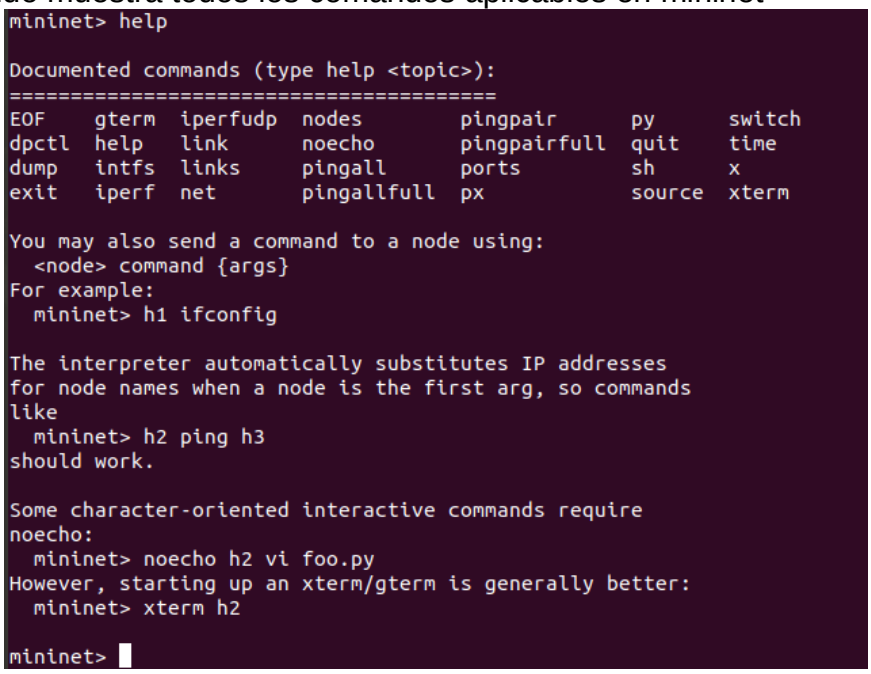


Imagen 20: Muestra de comandos de mininet

>exit
Comando de salida de mininet

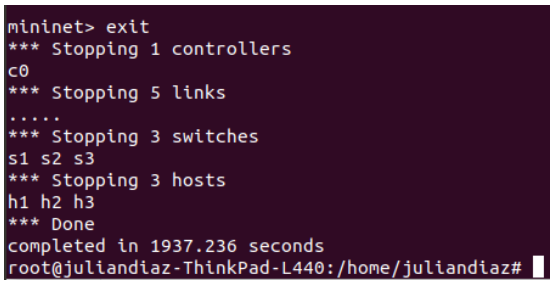


Imagen 21: salida de mininet

#mn -c
Comando para limpiar los desechos de la emulación sobre la terminal del equipo principal

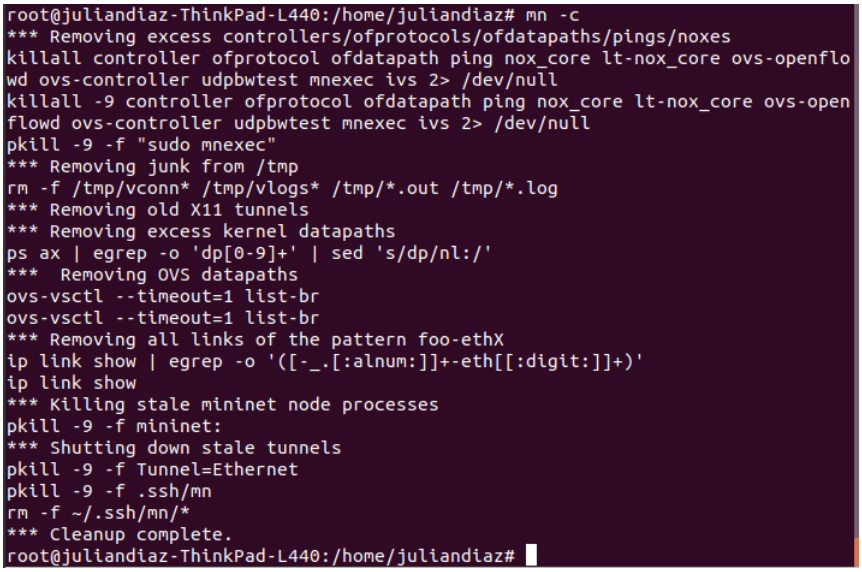


Imagen 22: limpieza de excesos retenidos de la emulación

$sudo su
Este comando nos permite ingresar a nuestra consola en modo privilegiado

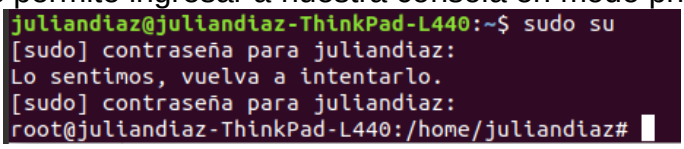


Imagen 23: aplicación comando sudo su

2.1.3 Lenguaje de programación para crear redes en mininet:

El siguiente método utilizado para crear redes definidas por software se basa en el la codificación mediante el lenguaje de programación Python, el cual se enseñara a emplear en el primer ejercicio, a continuación:

Se desarrollan los siguientes ejercicios con el fin de dar a conocer la aplicación de las redes SDN con Mininet, OpenDayLight. Es de destacar que los siguientes ejercicios son ejecutados sobre una máquina virtual dentro del mismo equipo que se registra como el controlador de cada una de las redes.

Antes de comenzar: se debe tener presente antes de inicializar el emulador de redes, que se debe inicializar la herramienta de administración de red OpenDayLight

Desde la terminal de usuario, ubicado sobre la carpeta del software/distribution-karaf- se ejecuta el comando para la inicialización de la herramienta:

$ sudo ./bin/karaf

Al inicializar se observará la siguiente imagen:

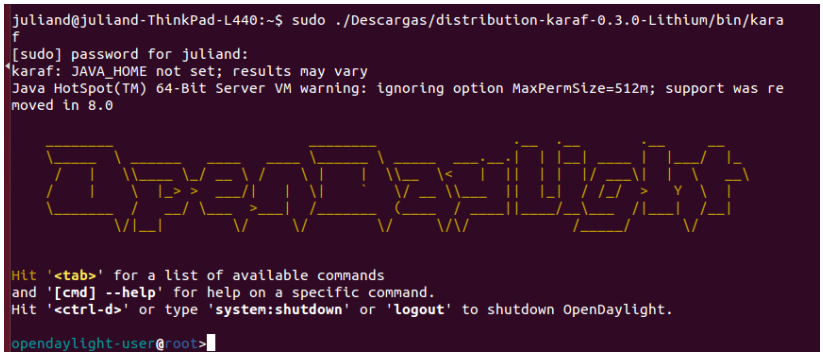


Imagen 24: Captura inicialización OpenDayLigth

Primer ejercicio,

En este primer ejercicio, se plantea una topología lineal, donde se establecen dos switchs y 4 host, la dirección ip para este controlador es: 192.168.0.113

Por medio del comando nano, podemos generar un archivo de texto el cual nos permite crear, editar y guardar en cualquier tipo de formato, en este caso debemos tener presente que los archivos deben ser guardados con el sufijo **.py** y se ubican en la carpeta custom sobre el repositorio de mininet. A continuación se ejecuta crea el archivo:

Imagen 25: apertura de archivo para visualización de código en lenguaje Python

Se debe importar diferentes tipos de funciones y clases con el fin de proporcionar limitaciones y aislamiento para optimizar el rendimiento de sus funcionalidades.

seguido, se debe definir la red, donde se establezca un tipo de red predeterminado o si es de topología libre, señalar **None** y negar su construcción, esto nos permitirá continuar con la creación de los dispositivos que usaremos para la emulación de la red definida por software que se desee representar,

Configuración del controlador:

Por medio de **net.addController** permite crear el dispositivo, para llevar a cabo su configuración, seguido de esto se debe agregar los parámetro deseados dentro de un paréntesis, de este modo: net.addController **(name='c0, controller=RemoteController, ip='192.168.0.113', protocol='tcp', port=6633)**. Por medio de name= nos permite asignarle el nombre de distinción a lo largo de la simulación destacando que debe ir entre apóstrofes, continuo a esto, se debe definir el tipo de controlador, en este caso se emplea un controlador remoto debido a la implementación de la herramienta OpenDayLight con el fin de visualizar la gestión, se debe asignar la dirección ip del dispositivo que se proyecta como controlador, en este caso es: 192.168.0.113 y su comunicación va a ser por medio del protocolo de control de transmisión a través del puerto 6633.

Se emplea el comando **info (' ___ ')** con el fin de imprimir un mensaje en el paso a paso de la ejecución del código.

Como en la configuración del controlador, es similar la configuración de los switchs y hosts, como se muestra a continuación:

(Nombre del dispositivo) = net.addSwitch o net.addHost ('(nombre del dispositivo)', **en este espacio se puede agregar alguna otra configuración que se desee para el dispositivo, separándolo por medio de comas**)

```
""" funcionando """
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.link import TCLink, Intf
from mininet.cli import CLI
from mininet.link import Intf
from mininet.log import setLogLevel, info
from subprocess import call

def myNetwork():
        net = Mininet( topo=None, build=False)

        info( '**Agregar controlador**\n')
        net.addController(name='c0',controller=RemoteController,
                ip='192.168.0.113', protocol='tcp', port=6633)

        info( '** Agregar conmutador **\n')
        s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
        s2 = net.addSwitch('s2', cls=OVSKernelSwitch)

        info( '** Agregar hosts **\n')
        h1 = net.addHost('h1', ip='10.0.0.1/24')
```

Imagen 26: visualización código Python ejercicio 1 parte 1

Por medio del comando net.addLink(X#, Y#) se permite crear los diferentes enlaces entre dispositivos, (No aplica para el o los controladores), seguido de esto para finalizar se agrega los inicializadores de red con la sentencia condicional de la inicialización de la red.

```
        net = Mininet( topo=None, build=False)

        info( '**Agregar controlador**\n')
        net.addController(name='c0',controller=RemoteController,
                ip='192.168.0.113', protocol='tcp', port=6633)

        info( '** Agregar conmutador **\n')
        s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
        s2 = net.addSwitch('s2', cls=OVSKernelSwitch)

        info( '** Agregar hosts **\n')
        h1 = net.addHost('h1', ip='10.0.0.1/24')
        h2 = net.addHost('h2', ip='10.0.0.2/24')
        h3 = net.addHost('h3', ip='10.0.0.3/24')
        h4 = net.addHost('h4', ip='10.0.0.4/24')
        info( '** Agregar links **\n')
        net.addLink(h3, s1)
        net.addLink(h4, s1)
        net.addLink(h1, s2)
        net.addLink(h2, s2)
        net.addLink(s1, s2)
        info('** Iniciando red **\n')
        net.start ()
        CLI (net)
        net.stop ()
```

Imagen 27: visualización código Python ejercicio 1 parte 2

```
        h3 = net.addHost('h3', ip='10.0.0.3/24')
        h4 = net.addHost('h4', ip='10.0.0.4/24')
        info( '** Agregar links **\n')
        net.addLink(h3, s1)
        net.addLink(h4, s1)
        net.addLink(h1, s2)
        net.addLink(h2, s2)
        net.addLink(s1, s2)
        info('** Iniciando red **\n')
        net.start ()
        CLI (net)
        net.stop ()
if __name__ == '__main__':
        setLogLevel( 'info' )
        myNetwork()
```

Imagen 28: visualización código Python ejercicio 1 parte 3

Para guardar el archivo con éxito ejecutamos Ctrl - x, aceptamos guardar el cambio y confirmamos el nombre del archivo recordando que debe finalizar en **.py**

Se ejecuta el archivo con el comando Python y el nombre del archivo con el sufijo .py, seguido de eso se observa la inicialización de la red, donde se establecen los parámetros codificados. Al aparecer mininet> ya podemos incursionar sobre la red establecida con los comandos brindados anteriormente.

```
root@mininet-vm:/home/mininet/mininet/custom# python prueba01.py
**Agregar controlador**
** Agregar conmutador **
** Agregar hosts **
** Agregar links **
** Iniciando red **
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 s1 s2
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s1-eth1
h4 h4-eth0:s1-eth2
s1 lo:  s1-eth1:h3-eth0 s1-eth2:h4-eth0 s1-eth3:s2-eth3
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth3
c0
mininet> _
```

Imagen 29: ejecución de la red generada por codificación en Python

Desde la terminal del dispositivo controlador inicializamos la herramienta OpenDayLight como se informó anteriormente, seguido de esto se abre el navegador web, y se consulta sobre el localhost por medio del puerto 8181 como se muestra a continuación: http://localhost:8181/index.html

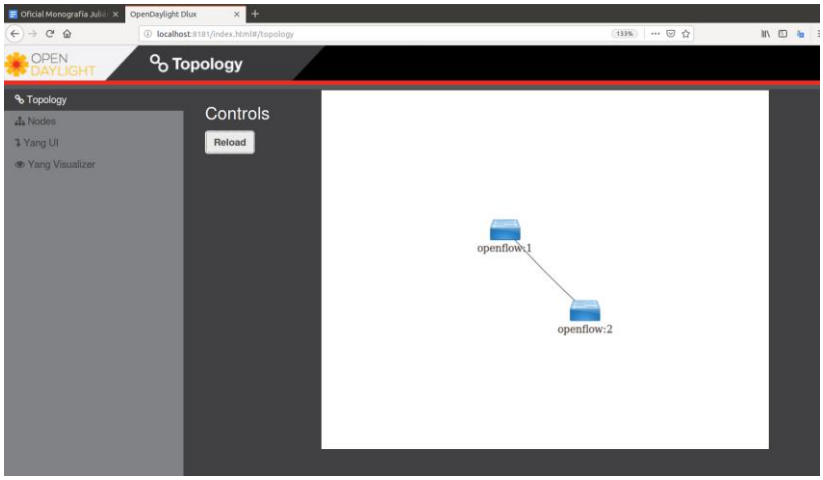


Imagen 30: Primera vista sobre navegador web en la herramienta OpenDayLight

Realizamos un Testing o pingall sobre toda la red, con el fin de reportar los estados de los dispositivos con el controlador para efectuar la visualización de la red

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1327>
<Host h2: h2-eth0:10.0.0.2 pid=1329>
<Host h3: h3-eth0:10.0.0.3 pid=1331>
<Host h4: h4-eth0:10.0.0.4 pid=1333>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=1319>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=1322>
<RemoteController c0: 192.168.0.113:6633 pid=1312>
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
.*** Results: ['176 Mbits/sec', '178 Mbits/sec']
mininet>
```

Imagen 31: Ejecución de comandos de prueba y/o Testing sobre la red

Se debe refrescar el controlador, y aparecerán los dispositivos creados y activos dentro de la red, cada switchs se identifica como openflow ya que es el protocolo que utilizan para compartir sus tablas de flujo con el controlador, y adicionalmente cada uno toma un número no repetible dentro de la red con el fin de caracterizar la tabla de su propiedad como se muestra más adelante.

Sobre Topology se visualiza de forma interactiva el controlador entiende lo que ocurre sobre la red.

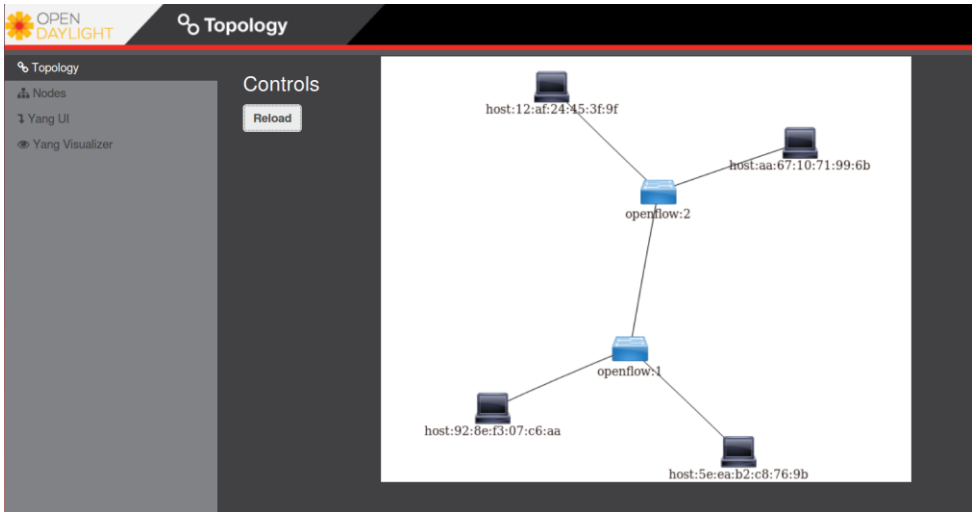


Imagen 32: vista de la topología sobre el controlador con OpenDayLight

En las pestaña Nodes se observan estadísticas de red, cantidad de conexiones y los switchs sometidos al protocolo Openflow detectados por opendaylight y el controlador asignado.

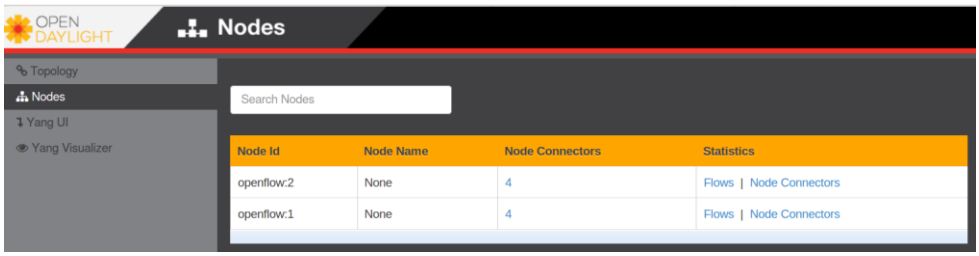
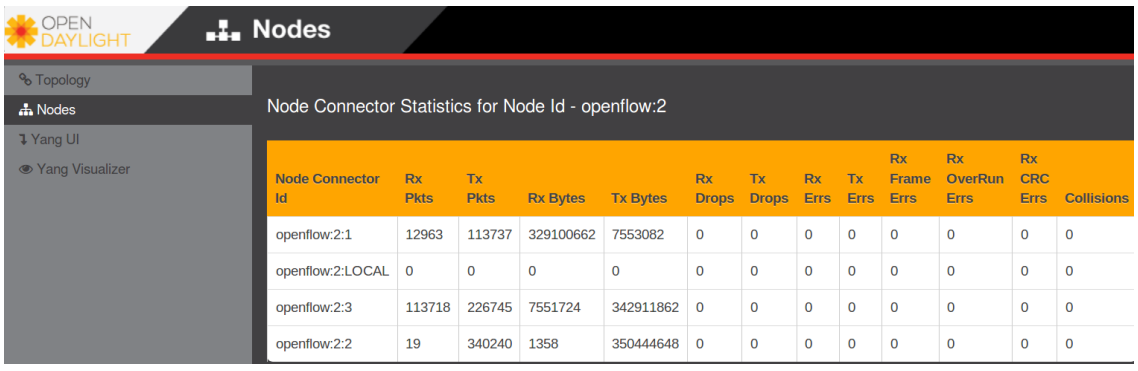


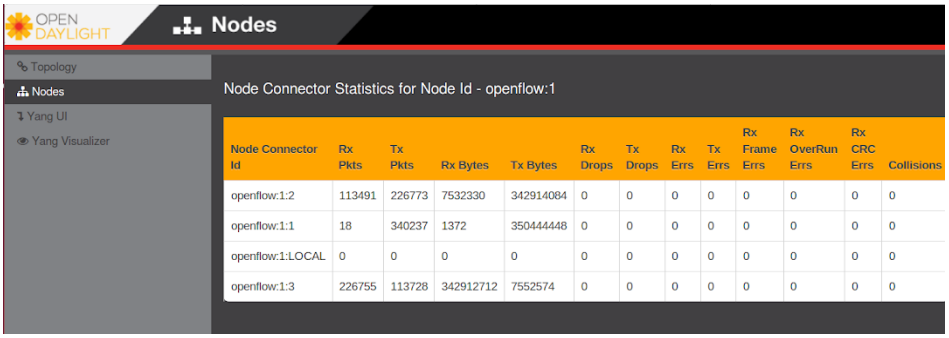
Imagen 33: Nodos openflow con cantidad de conexiones en OpenDayLight

Ingresando sobre cada una de las conexiones de los nodos podemos observar las diferentes estadísticas o datos de interés como: bytes recibidos, bytes transmitidos, paquetes enviados y recibidos, paquetes perdidos entre la conexión y entre otros más que se dan a conocer.



Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:2:1	12963	113737	329100662	7553082	0	0	0	0	0	0	0	0
openflow:2:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:2:3	113718	226745	7551724	342911862	0	0	0	0	0	0	0	0
openflow:2:2	19	340240	1358	350444648	0	0	0	0	0	0	0	0

Imagen 34: nodos conectados sobre el switchs openflow2



Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs	Rx Frame Errs	Rx OverRun Errs	Rx CRC Errs	Collisions
openflow:1:2	113491	226773	7532330	342914084	0	0	0	0	0	0	0	0
openflow:1:1	18	340237	1372	350444448	0	0	0	0	0	0	0	0
openflow:1:LOCAL	0	0	0	0	0	0	0	0	0	0	0	0
openflow:1:3	226755	113728	342912712	7552574	0	0	0	0	0	0	0	0

Imagen 35: nodos conectados sobre el switchs openflow1

Se finaliza la ejecución del software con exit o EOF, y sobre la terminal se ejecuta sudo mn -c con el fin de depurar el restante de la configuración ejecutada sobre mininet.

```
root@mininet-vm:/home/mininet/mininet/custom# mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflowd
 ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-openflo
owd ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
***  Removing OVS datapaths
ovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
ip link show
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
root@mininet-vm:/home/mininet/mininet/custom#
```

Imagen 36: depuración de la configuración sobre mininet

Capítulo 3: DESARROLLO

1. REVISIÓN DE PARÁMETROS DE REDES TRADICIONALES Y SDN

En las redes tradicionales o convencionales de Ethernet, se puede llevar a cabo la a cabo la asignación de valores al ancho de banda con el fin de predeterminar su valor, o emplear el ancho de banda como en este caso, sobre las redes tradicionales el valor del ancho de banda no genera una mayor afectación sobre la velocidad o la capacidad de la conexión de enlaces, en este caso, utilizamos interfaces seriales entre router zonales, lo que implica un valor predeterminado de 128 kb/s. Para las redes SDN el ancho de banda se verá afectado por la cantidad de dispositivos que se encuentren conectados; mininet brinda un ancho de banda total, el cual es compartido por cada una de las máquinas virtuales que se encuentren sobre el hardware instanciadas,

La latencia de la red se puede analizar por medio de un ping extendido o ejecutando varias pruebas de conexión evidenciando la distancia que exista entre el emisor y receptor, este método es comúnmente utilizado para resolver los diferentes problemas de accesibilidad con los dispositivos, seguido se tiene presente el Throughput o tasa de transferencia efectiva ya que este parámetros nos brinda un nivel de aplicabilidad del ancho de banda total disponible. Para las SDN se proveerá una latencia baja, que varía según la cantidad de nodos, la disponibilidad de procesamiento y capacidad de memoria que existan sobre la emulación,

El tiempo de vida de paquetes es el valor sobre el protocolo de internet (IP), donde se prevé conocer diferentes valores en cada una de las emulaciones, ya que cada software cuenta con un valor predeterminado con el fin de indicar el alcance que puede tener un paquete sobre la red, se tiene presente los siguientes valores de TTL:

- 255: sin restricciones de alcance sobre la red,
- 128: limitaciones sobre un mismo continente
- 64: limitaciones sobre una misma región
- 32: se restringe sobre el mismo sitio

Estos valores los vemos aplicados sobre el Testing que se ve representado más adelante.

2. SELECCIÓN DE LA TOPOLOGÍA

En la actualidad, se puede llevar a cabo emulación gráfica, implementando la topología que se desee. En el siguiente ejercicio se presenta una misma topología en dos software diferentes, el primero: permite emular la red en un entorno tradicional, sobre Packet tracer, mientras que el segundo: permite emular la red en un entorno SDN sobre Mininet. Evaluaremos características relevantes que giran en torno al tiempo de vida de los paquetes y el ancho de banda que manejan.

Topología Packet tracer

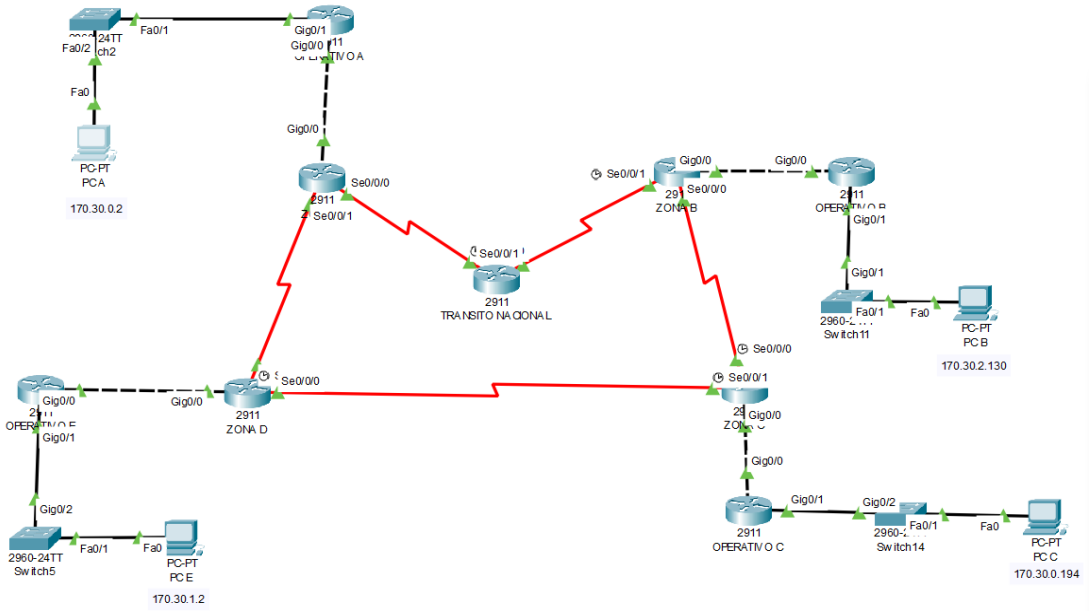


Imagen 37: Topología Packet Tracer

La topología en Packet tracer está representada por 9 Routers, 4 switchs y 4 equipos. Es importante destacar la autonomía que tiene cada router para redireccionar el envío de paquetes a través de la red, El cable empleado para generar la conexión es: serial entre los routers zonales (A, B, C, D, Transito Nacional), entre routers al interior de la zona, cable cruzado. De router a switch cable directo al igual que de switch a equipo.

Topología Mininet

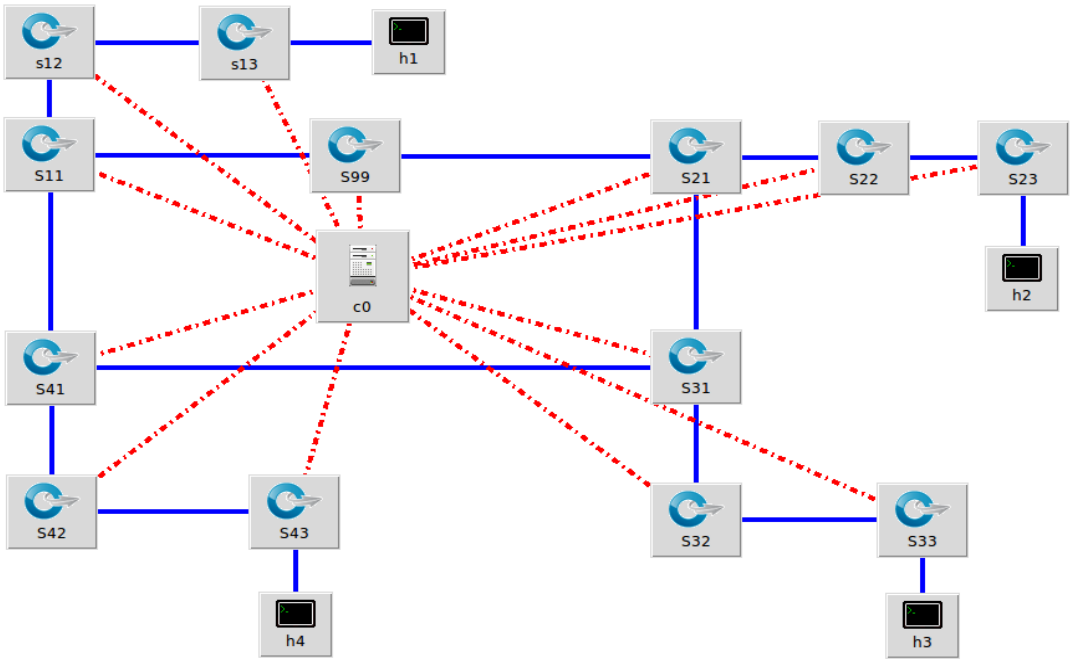


Imagen 38: Topología mininet

La topología en Mininet se encuentra representada por 13 routers, 4 equipos y un controlador, basándonos en el objetivo principal de los switch el cual es el reenvío de

información, se presenta su conectividad directa entre ellos, mientras mantienen una conexión directa con un único controlador.

Al finalizar un proceso de aprendizaje en redes como el CCNP en cisco, se selecciona esta topología ya que se busca emular de forma simplificada una red de alcance global, comúnmente halladas en grandes compañías nacionales, e internacionales que buscan interconectar sus sedes, de forma económica y accesible, es por ello que se hace con el objetivo de adquirir datos claramente comparables, al ejecutar la conexión, es por ello, que se brinda una topología central en forma de anillo, con un nodo adicional representando un salto transitorio, teniendo en cuenta el desprendimiento de cada nodo principal con una topología línea, con el fin de garantizar un distanciamiento entre los demás hosts, ello con el fin de lograr visualizar y exponer cada parámetro definido en el proyecto

Se ejecuta la emulación sobre el software, recopilando información de conectividad entre los equipos, exportando tablas y gráficas de análisis como se muestra a continuación:

3. SIMULACIÓN DE LAS TOPOLOGÍAS Y RECOPILACIÓN DE DATOS DE DESEMPEÑO OBTENIDOS (PARÁMETROS DE CONFIGURACIÓN DE LAS TOPOLOGÍAS PARA LAS SIMULACIONES)

Inicializamos la emulación de la red inicializando la herramienta de gestión de servicio de red OpenDayLight y seguido de ello inicializamos el programa .py de la red mininet como se muestra en la siguiente imagen:

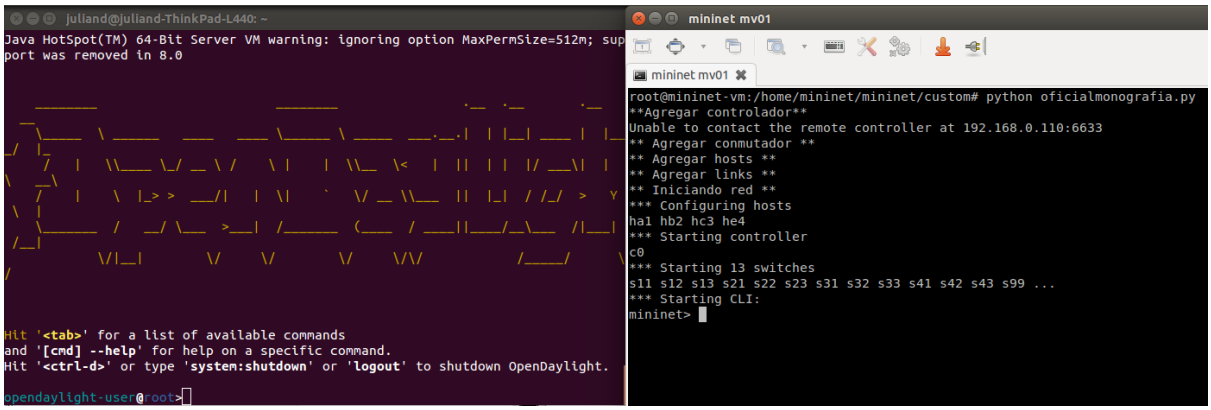


Imagen 39: Inicialización red SDN

En la imagen se observan dos interfaces: al lado izquierdo la inicialización de la herramienta OpenDayLight y al lado derecho se observa la inicialización de la red definida por software,

4. RESULTADOS OBTENIDOS EN LAS SIMULACIONES

Al realizar la inicialización nos dirigimos a nuestra ventana en el navegador donde tendremos acceso a la vista de la red como controlador, al inicializar la red solo observaremos cada switch como un proveedor de tablas de flujo Openflow, donde se enumeran para mantener una organización, al iniciar la comunicación entre los diferentes host, refrescara en pantalla cada uno de los dispositivos conectados:

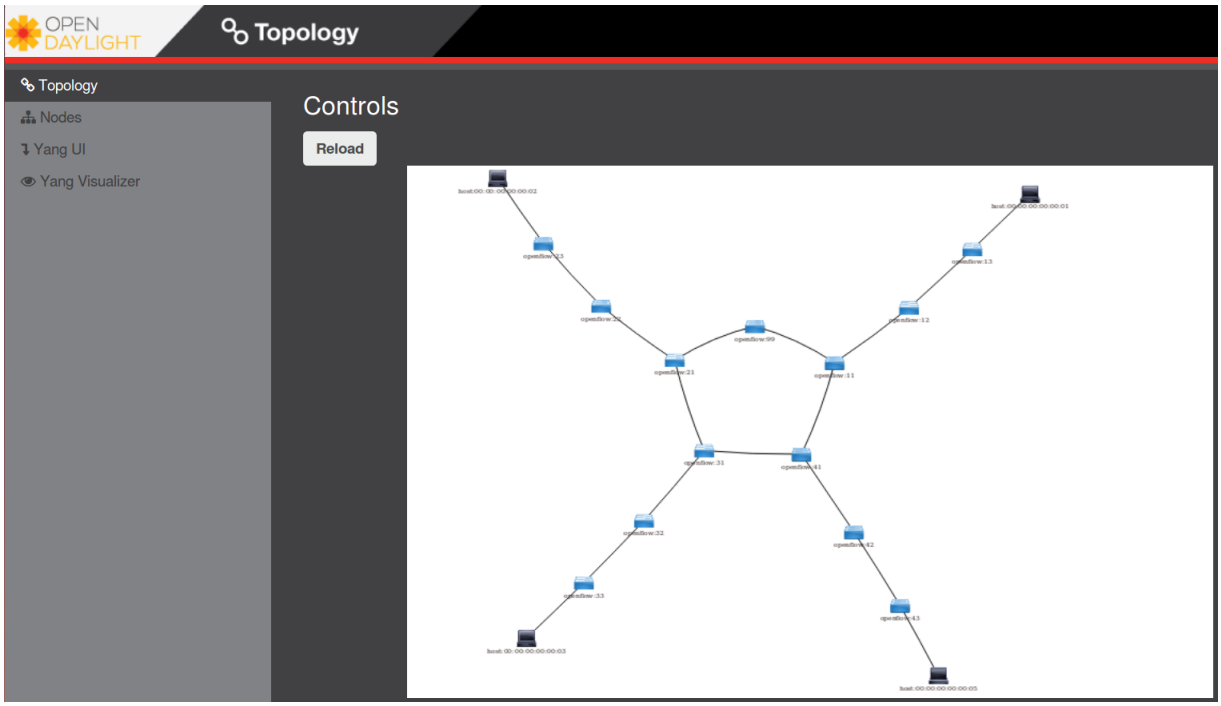


Imagen 40: Red SDN sobre el controlador

Para llevar a cabo la ejecución de las pruebas, se realizarán diferentes pings entre cada uno de los hosts, teniendo en cuenta los diferentes saltos que debe realizar, lo cual es fundamental para tener presente en la variación del tiempo sobre la red, cada evidencia está acompañada de una captura en cada emulación y su tabla de datos, se debe tener presente el tamaño de los paquetes ya que son diferentes en cada emulador, para Packet tracer su tamaño es de 32 bytes, mientras que para mininet su tamaño es de 64 bytes, adicionalmente el tiempo de vida sobre Packet tracer es de 126 donde se limita en un mismo continente, y mininet tiene un tiempo de vida de 64 donde se limita a una misma región,

4.1 PRUEBA ENTRE PC A Y PC B: PACKET TRACER

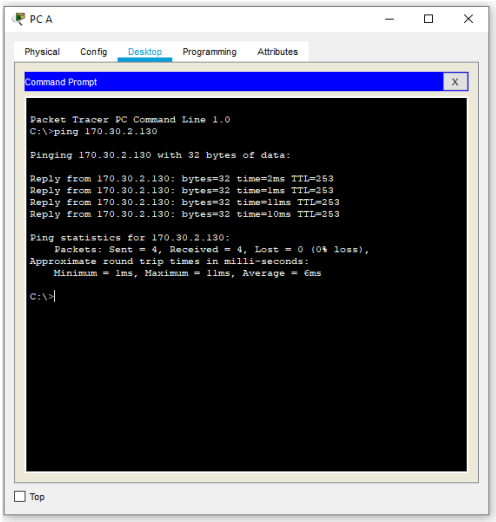


Imagen 41: ping entre pca y pcb en Packet tracer

MININET

```
mininet> ha1 ping -c 4 hb2
PING 170.30.2.130 (170.30.2.130) 56(84) bytes of data.
64 bytes from 170.30.2.130: icmp_seq=1 ttl=64 time=0.461 ms
64 bytes from 170.30.2.130: icmp_seq=2 ttl=64 time=0.223 ms
64 bytes from 170.30.2.130: icmp_seq=3 ttl=64 time=0.265 ms
64 bytes from 170.30.2.130: icmp_seq=4 ttl=64 time=0.219 ms

--- 170.30.2.130 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 0.219/0.292/0.461/0.099 ms
mininet>
```

Imagen 42: ping entre pca y pcb en mininet

PAQUETE	TIEMPO ENTRE PC A - PC B		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,002000	0,000461	253	64
2	0,001000	0,000223	253	64
3	0,011000	0,000265	253	64
4	0,010000	0,000219	253	64

Tabla 1: información prueba entre pca y pcb

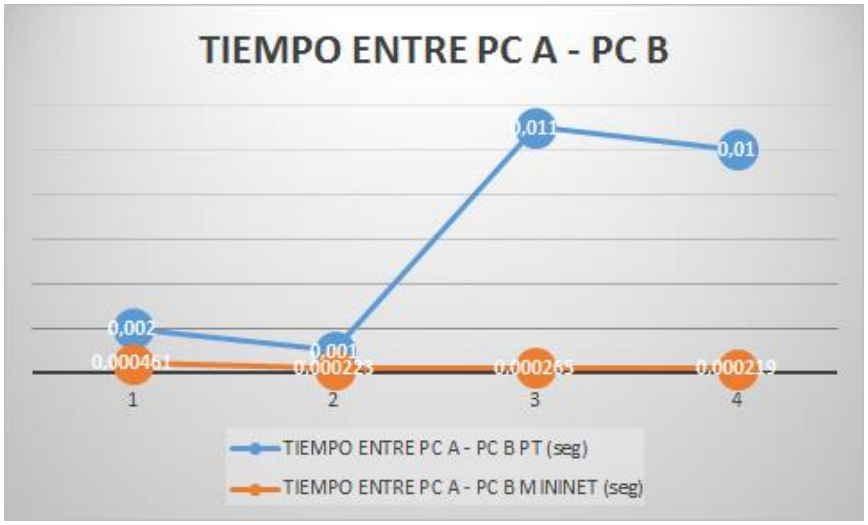


Imagen 43: gráfica comparativa entre pca y pcb

Al emplear la unidad de diagnóstico, se logra observar la variación en los tiempos, donde se destaca la aceleración que presenta el tiempo de envió de paquetes en las redes definidas por software, y esto se debe a que cada switch consulta cada una de sus tablas de flujo antes de realizar el reenvío de tráfico, donde el controlador es el encargado de centralizar y optimizar el trazado de cada direccionamiento de tráfico. Mientras que para el caso de la red convencional, se consulta en las tablas MAC donde si no se reconoce la ruta, emite un mensaje tipo broadcast (ARP) a la espera del receptor, por consiguiente allí inicia la latencia en el envío de las tramas. Del mismo modo se puede analizar el tiempo de vida de los paquetes de la red, donde

mininet brinda el valor por defecto de 64 donde lo limita a una misma región, para el caso de la red convencional no brinda mayor restricción de alcance con el valor 253,

4.2 PRUEBA ENTRE PC A Y PC C

PACKET TRACER

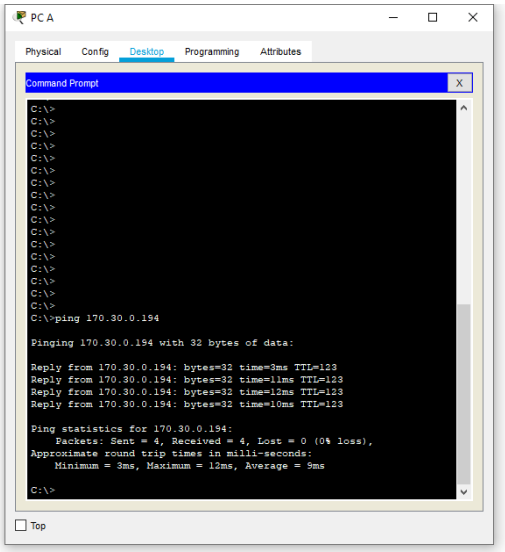


Imagen 44: ping entre pca y pcc en Packet tracer

MININET

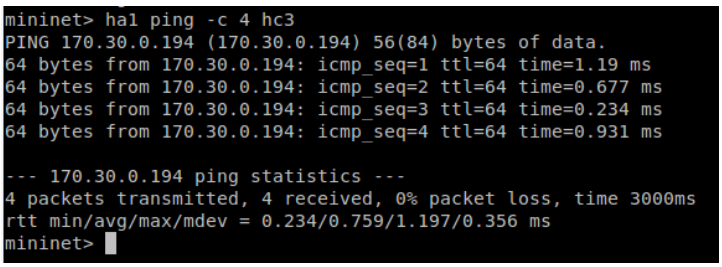


Imagen 45: ping entre pca y pcc en mininet

PAQUETE	TIEMPO ENTRE PC A - PC C		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,003000	0,001190	123	64
2	0,011000	0,000677	123	64
3	0,012000	0,000234	123	64
4	0,010000	0,000931	123	64

Tabla 2: información prueba entre pca y pcc

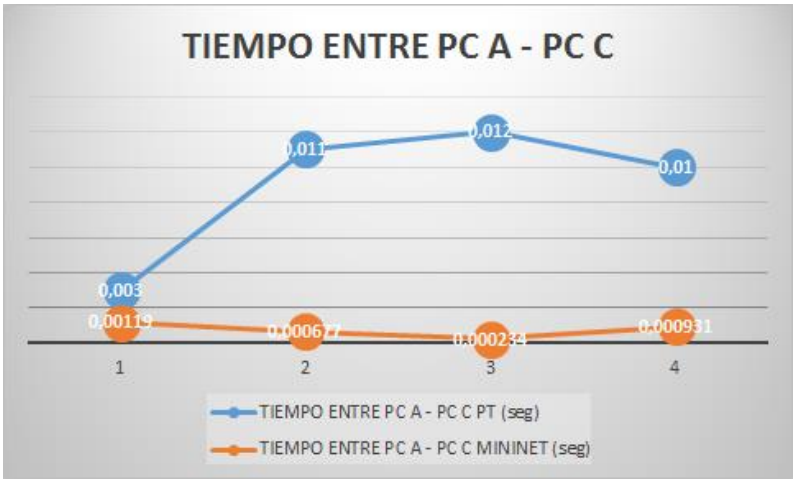


Imagen 46: gráfica comparativa entre pca y pcc

Al realizar cada prueba, se puede observar la variación del tiempo que existe entre cada conexión, en este caso, la red SDN mantiene su acelerado índice de comunicación, mientras sobre las redes tradicionales se evidencia un claro retardo entre los diferentes equipos, esto se puede justificar también a la trayectoria, debido a que la ruta entre el PC A y el PC C es la más larga al igual que la ruta entre el PC E Y EL PC B

4.3 PRUEBA ENTRE PC A Y PC E

PACKET TRACER

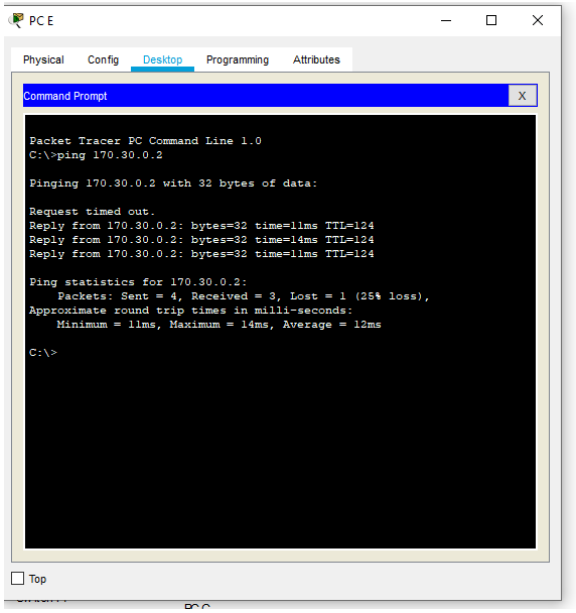


Imagen 47: ping entre pca y pce en Packet tracer

MININET

```
mininet> ha1 ping -c 4 he4
PING 170.30.1.2 (170.30.1.2) 56(84) bytes of data.
64 bytes from 170.30.1.2: icmp_seq=1 ttl=64 time=0.413 ms
64 bytes from 170.30.1.2: icmp_seq=2 ttl=64 time=0.213 ms
64 bytes from 170.30.1.2: icmp_seq=3 ttl=64 time=0.269 ms
64 bytes from 170.30.1.2: icmp_seq=4 ttl=64 time=0.172 ms

--- 170.30.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 0.172/0.266/0.413/0.093 ms
mininet>
```

Imagen 48: ping entre pca y pce en mininet

PAQUETE	TIEMPO ENTRE PC A - PC E		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,000000	0,000413	124	64
2	0,011000	0,000213	124	64
3	0,014000	0,000269	124	64
4	0,011000	0,000172	124	64

Tabla 3: información prueba entre pca y pce

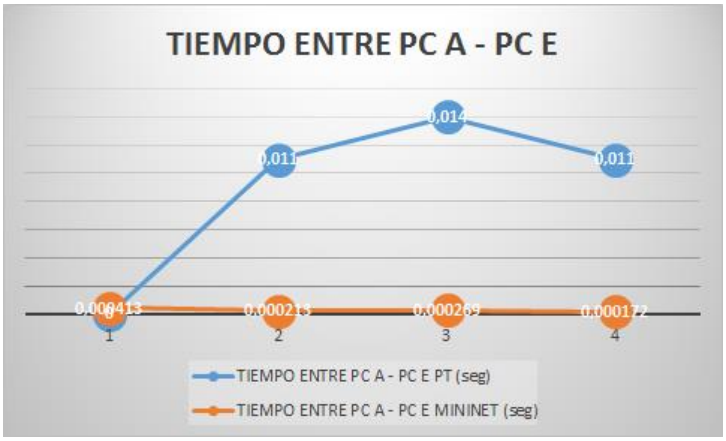


Imagen 49: grafica comparativa entre pca y pce

En esta prueba se puede evidenciar la pérdida del primer paquete, y como existe un retardo en la comunicación, en el segundo paquete se evidencia el retardo para iniciar el envío de paquetes, se identifica que no existe gran variación con las redes SDN y se pronuncia casi como una línea recta, se evidencia el ttl de mininet se mantiene sobre 64, mientras que el de Packet tracer se encuentra sobre 124.

4.4 PRUEBA ENTRE PC B Y PC C

PACKET TRACER

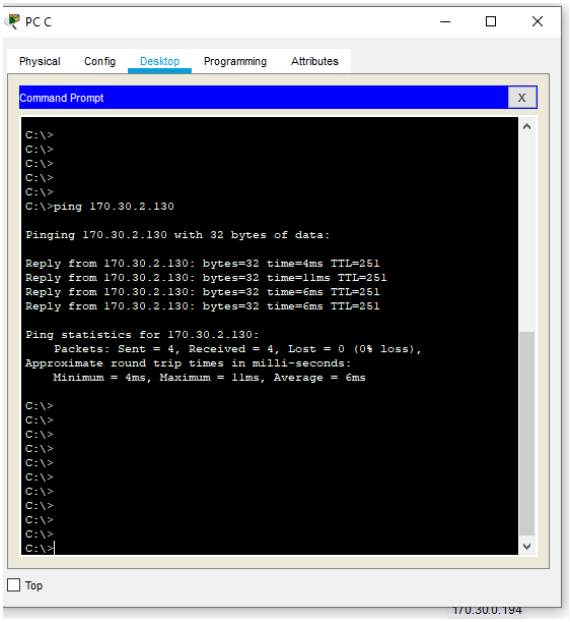


Imagen 50: ping entre pcb y pcc en Packet tracer

MININET

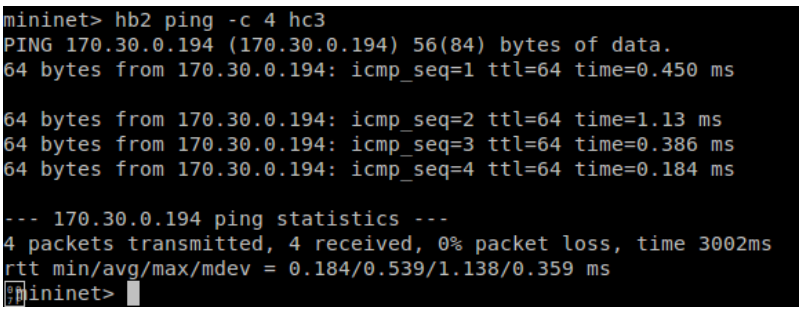


Imagen 51: ping entre pcb y pcc en mininet

PAQUETE	TIEMPO ENTRE PC B - PC C		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,004000	0,000450	251	64
2	0,011000	0,001130	251	64
3	0,006000	0,000386	251	64
4	0,006000	0,000184	251	64

Tabla 4: información prueba entre pcb y pcc

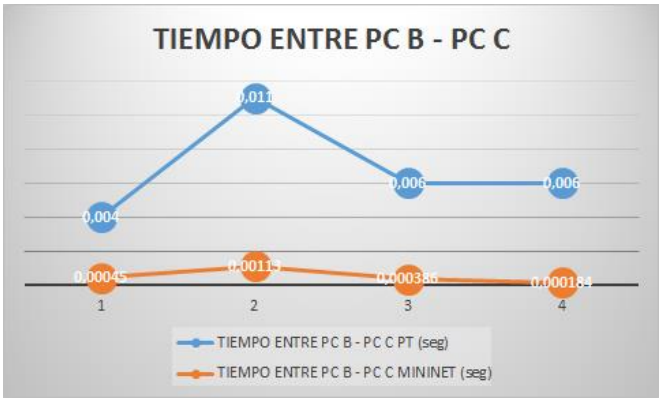


Imagen 52: grafica comparativa entre pcb y pcc

La distancia entre el PC B y el PC C es similar a la que podría tener el PC C con el PC B, se identifica un “sincronismo” entre las dos redes, sobre él ya que en el segundo paquete se evidencia un pico de retardo durante la prueba, claramente cada red busca su armonía en la transmisión al lanzar el tercer paquete

4.5 PRUEBA ENTRE PC B Y PC E

PACKET TRACER

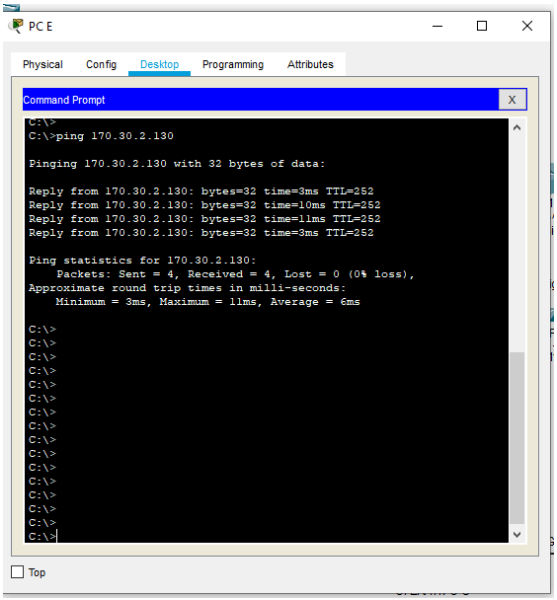


Imagen 53: ping entre pcb y pce en Packet tracer

MININET

```
mininet> hb2 ping -c 4 he4
PING 170.30.1.2 (170.30.1.2) 56(84) bytes of data.
64 bytes from 170.30.1.2: icmp_seq=1 ttl=64 time=0.745 ms
64 bytes from 170.30.1.2: icmp_seq=2 ttl=64 time=0.222 ms
64 bytes from 170.30.1.2: icmp_seq=3 ttl=64 time=0.199 ms
64 bytes from 170.30.1.2: icmp_seq=4 ttl=64 time=0.218 ms

--- 170.30.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.199/0.346/0.745/0.230 ms
```

Imagen 54: ping entre pcb y pce en mininet

PAQUETE	TIEMPO ENTRE PC B - PC E		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,002000	0,000745	252	64
2	0,010000	0,000222	252	64
3	0,011000	0,000199	252	64
4	0,003000	0,000218	252	64

Tabla 5: información prueba entre pcb y pce

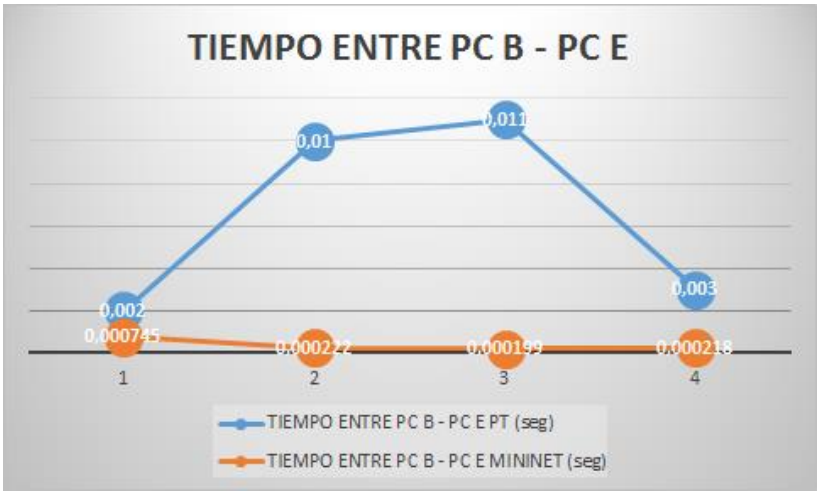


Imagen 55: grafica comparativa entre pcb y pce mininet

Se observa en la gráfica lo siguiente: el primer y el último paquete cuenta con una respuesta acertada en la comunicación sobre las redes tradicionales, mientras que el paquete 2 y 3 presentan un retraso caracterizado en medio de la conexión. Mientras que sobre la red SDN se observa un retraso, solo sobre el primer paquete, mientras que la demás trama se mantiene constante en la aceleración del envió de la información, por otro lado se observa en este caso que el ttl de la emulación sobre Packet tracer es de 252, y mininet cuenta con 64 como en las anteriores emulaciones.

4.6 PRUEBA ENTRE PC C Y PC E

PACKET TRACER

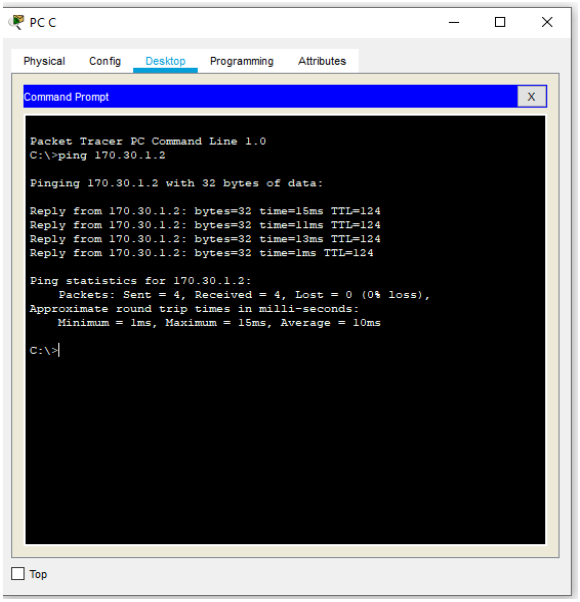


Imagen 56: ping entre pcc y pce en Packet tracer

MININET

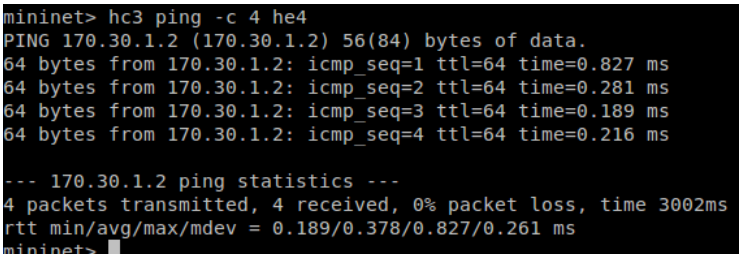


Imagen 57: ping entre pcc y pce en mininet

PAQUETE	TIEMPO ENTRE PC C - PC E		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,015000	0,000827	128	64
2	0,011000	0,000281	128	64
3	0,013000	0,000189	128	64
4	0,001000	0,002160	128	64

Tabla 6: información prueba entre pcc y pce

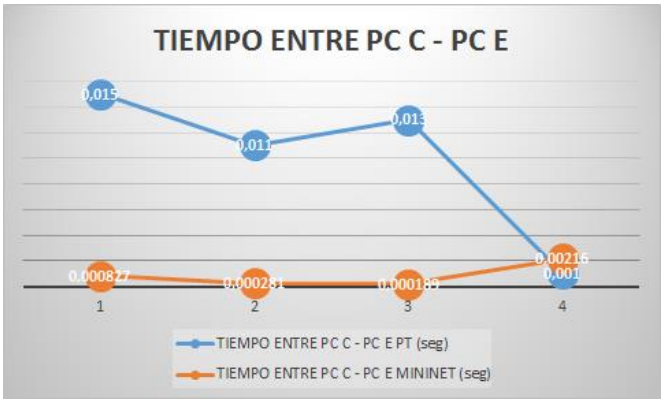


Imagen 58: gráfica comparativa entre pcc y pce

A continuación se observa como la red tradicional inicia con un evidente retraso, mejorando efectivamente al avanzar en el envío de los siguientes paquetes, mientras se observa que la red SND cuenta con una inicialización normalizada, pero finaliza con un pico de retraso, esto puede ocurrir al encontrarse la solicitud del controlador para con los switchs donde soliciten las diferentes tablas de reenvío.

Capítulo 3: Análisis y resultados

Se realiza una prueba con una réplica de 30 paquetes de datos con el fin de analizar de forma más detallada la comparación entre las dos redes, en esta ocasión se escogen dos equipos que cuenten con una distancia considerable para ajustar el análisis.

PRUEBAS DE ANCHO DE BANDA

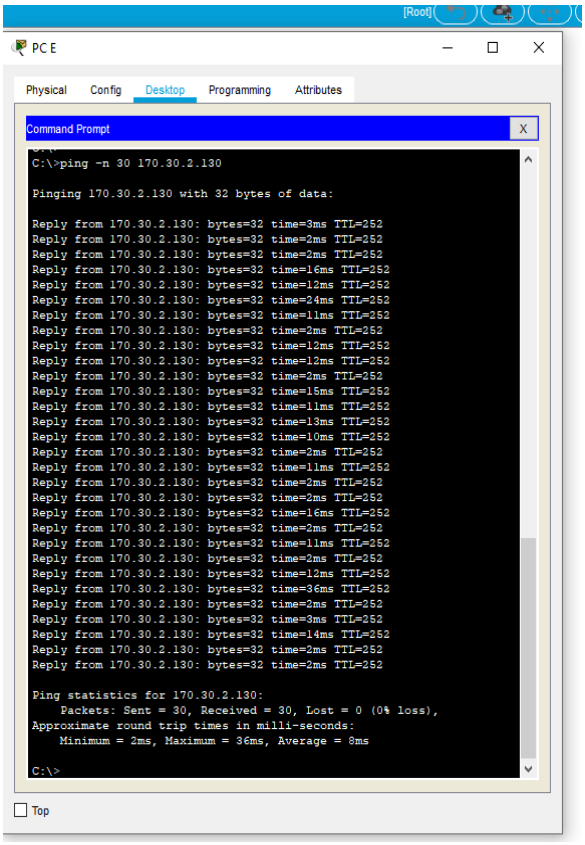


Imagen 59: ping entre pcc y pce en Packet tracer

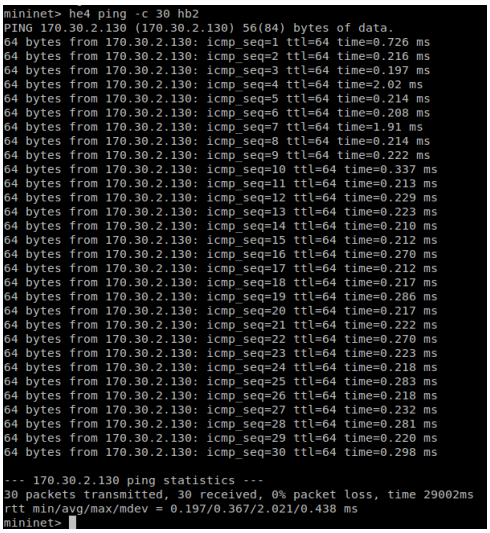


Imagen 60: ping entre pce y pcb en mininet

PAQUETE	TIEMPO ENTRE PC E - PC B		TTL	
	PT (seg)	MININET (seg)	TTL (PT)	TTL (MN)
1	0,003000	0,000726	252	64

2	0,002000	0,000216	252	64
3	0,002000	0,000197	252	64
4	0,016000	0,002020	252	64
5	0,012000	0,000214	252	64
6	0,024000	0,000208	252	64
7	0,011000	0,001910	252	64
8	0,002000	0,000214	252	64
9	0,012000	0,000222	252	64
10	0,012000	0,003370	252	64
11	0,002000	0,000213	252	64
12	0,015000	0,000229	252	64
13	0,011000	0,000223	252	64
14	0,013000	0,000210	252	64
15	0,010000	0,000212	252	64
16	0,002000	0,000270	252	64
17	0,011000	0,000212	252	64
18	0,002000	0,000217	252	64
19	0,002000	0,000286	252	64
20	0,016000	0,000217	252	64
21	0,002000	0,000222	252	64
22	0,011000	0,000270	252	64
23	0,002000	0,000223	252	64
24	0,012000	0,000218	252	64
25	0,036000	0,000283	252	64
26	0,002000	0,000218	252	64
27	0,003000	0,000232	252	64
28	0,014000	0,000281	252	64
29	0,002000	0,000220	252	64
30	0,002000	0,000298	252	64

Tabla 7: información prueba entre pce y pcb 30 paquetes

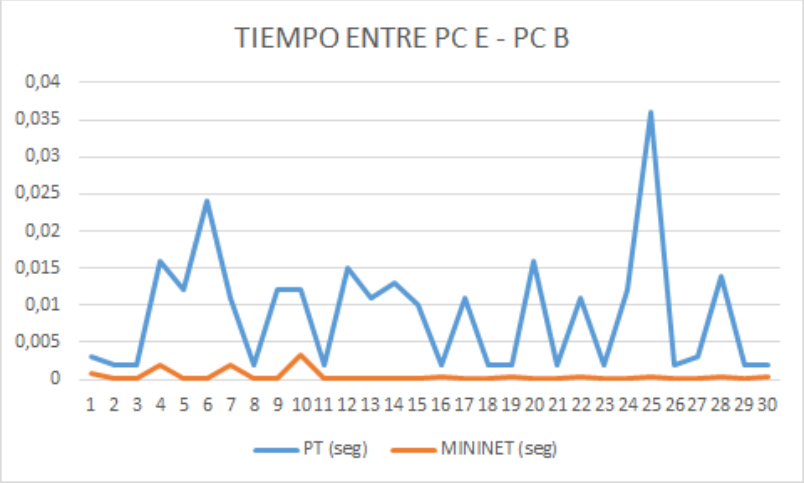


Imagen 61: grafica comparativa entre pce y pcb mininet

Durante la transmisión de los 30 paquetes de Testing, se puede evidenciar de forma puntual lo siguiente:

1. La red SDN cuenta con un tiempo de envío de mayor aceleración, esto se debe a la consulta que realiza cada switch con su tabla de flujo sincronizada por el controlador antes de iniciar el envío de paquetes, mientras la red convencional mantiene su consulta sobre tablas Mac y de no reconocer el la ruta de destino se generan los retrasos evidentes hasta conocer su destino.
2. Los picos sobre la gráfica nos dan a entender el cómo varían las redes con respecto al tiempo, sobre las redes SDN podemos entender que realiza el trazado de la ruta, y lo guarda para mantener un envío acelerado constante a lo largo de la transmisión, para la red tradicional, se puede evidenciar el retardo mientras realiza la consulta del estado de cada dispositivo, y del mismo modo la consulta de la ruta indicada, con el fin de garantizar el envío de paquetes de datos. es de resaltar el tamaño de diferencia de los paquetes, y aun así la transmisión sobre SDN se evidencia que es bastante acelerada a comparación de la variabilidad que presenta la red convencional.
3. La latencia se hace sumamente evidente dentro de la red convencional, y esta es una de las características del porqué el mundo de las telecomunicaciones se está dirigiendo hacia las redes programables, pues evidenciamos una baja latencia en la variación de la transmisión de los datos sobre las redes SDN donde garantiza la optimización de recursos de red.

Pruebas Realizadas sobre Packet tracer

PAQUETE	TIEMPO ENTRE PC A - PC B	TIEMPO ENTRE PC A - PC C	TIEMPO ENTRE PC A - PC E	TIEMPO ENTRE PC B - PC C	TIEMPO ENTRE PC B - PC E	TIEMPO ENTRE PC C - PC E
	PT (seg)	PT (seg)	PT (seg)	PT (seg)	PT (seg)	PT (seg)
1	0,002000	0,003000	0,000000	0,004000	0,002000	0,015000
2	0,001000	0,011000	0,011000	0,011000	0,010000	0,011000
3	0,011000	0,012000	0,014000	0,006000	0,011000	0,013000

4	0,010000	0,010000	0,011000	0,006000	0,003000	0,001000

Tabla 8: información sobre todas las pruebas realizadas en Packet tracer

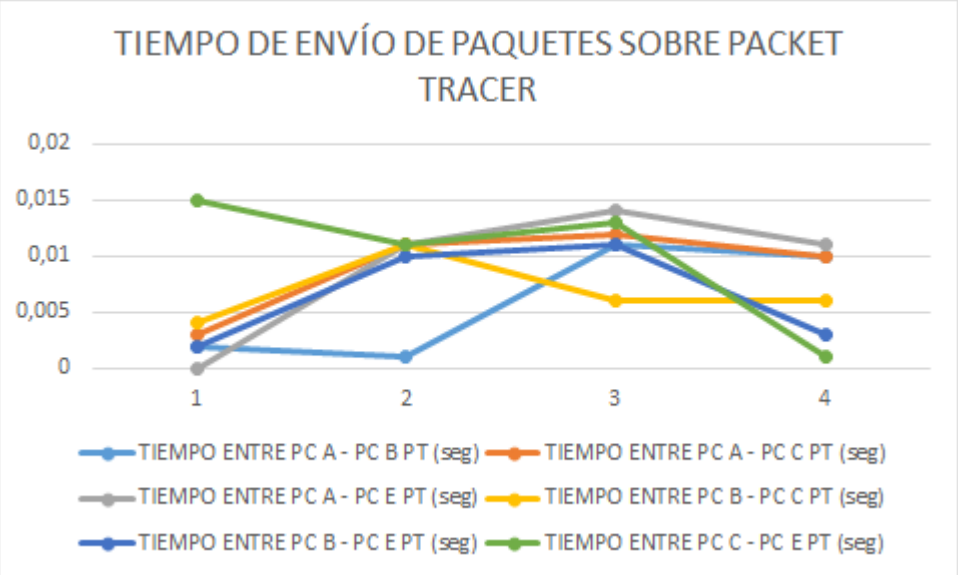


Imagen 62: gráfica comparativa entre todas las pruebas realizadas sobre Packet tracer

Se realiza un análisis comparativo directo entre cada una de las pruebas realizadas sobre Packet tracer y se evidencia el promedio estándar en la duración

Pruebas realizadas sobre Mininet

PAQUETE	TIEMPO ENTRE PC A - PC B	TIEMPO ENTRE PC A - PC C	TIEMPO ENTRE PC A - PC E	TIEMPO ENTRE PC B - PC C	TIEMPO ENTRE PC B - PC E	TIEMPO ENTRE PC C - PC E
	MININET (seg)	MININET (seg)	MININET (seg)	MININET (seg)	MININET (seg)	MININET (seg)
1	0,000461	0,00119	0,000413	0,00045	0,000745	0,000827
2	0,000223	0,000677	0,000213	0,00113	0,000222	0,000281
3	0,000265	0,000234	0,000269	0,000386	0,000199	0,000189
4	0,000219	0,000931	0,000172	0,000184	0,000218	0,00216

Tabla 9: información sobre todas las pruebas realizadas en Packet tracer

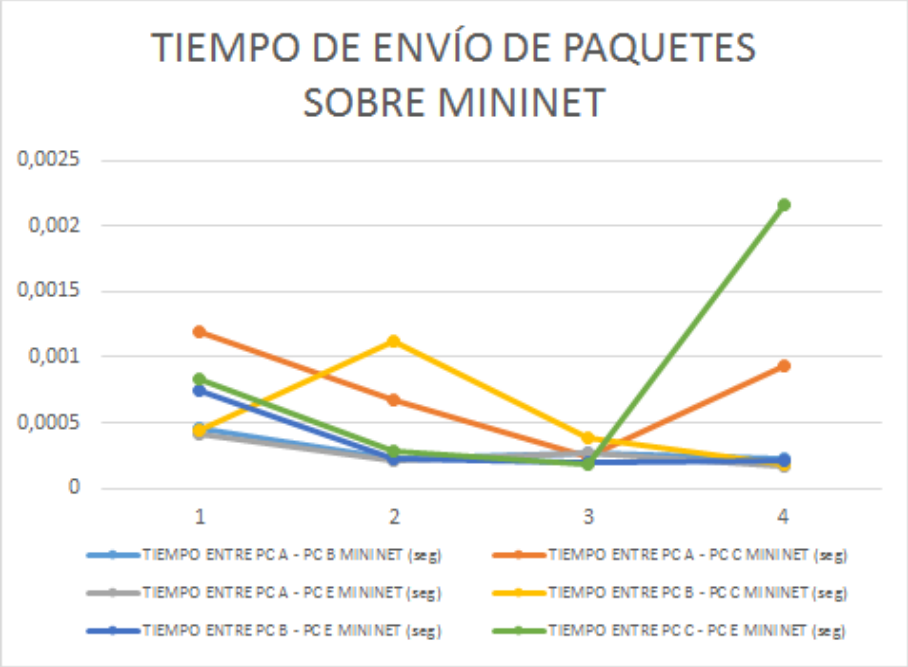


Imagen 63: gráfica comparativa entre todas las pruebas realizadas sobre mininet

Al proyectar cada uno de los resultados obtenidos de las diferentes pruebas sobre la misma gráfica, se logra evidenciar que no existe una gran variación ni saltos con respecto al tiempo, donde prima el ahorro del tiempo durante la acelerada entrega de paquetes, del mismo modo se comprende el bajo nivel de latencia que posee y la nulidad en pérdida de datos sobre esta simulación, y se destaca evidentemente el control sobre el ancho de banda, ya que al regular el tiempo de la transmisión de paquetes, se administra eficientemente su asignación dentro de la red.

Se finaliza con la red ejecutada sobre mininet aplicando el comando iperf, con el fin de conocer el ancho de banda disponible, y su variación:

```
mininet> iperf
*** Iperf: testing TCP bandwidth between ha1 and he4
.*** Results: ['227 Mbits/sec', '232 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between ha1 and he4
*** Results: ['237 Mbits/sec', '242 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between ha1 and he4
*** Results: ['221 Mbits/sec', '226 Mbits/sec']
mininet> iperf
*** Iperf: testing TCP bandwidth between ha1 and he4
*** Results: ['240 Mbits/sec', '244 Mbits/sec']
mininet>
```

Imagen 64: Captura de datos ancho de banda disponible

Análisis principales de la emulación: ventajas y desventajas

	REDES TRADICIONALES	**REDES SDN**
VENTAJAS	Cada dispositivo cuenta con la autonomía de decidir la ruta óptima para el envío de datos. La configuración de sus dispositivos se realiza en caliente con normalidad, no afecta en mayores instancias estos procedimientos. Experiencia en el mercado educativo y académico para el desarrollo de cualquier tipo de infraestructura de red. Seguridad imponente disminuyendo el riesgo al intentar violar la seguridad de la red.	Al tener un único dispositivo que realiza la toma de decisiones, optimiza el tiempo y evita los retardos en las transmisiones de paquetes. Mayor disponibilidad de ancho de banda sobre la red. Menor latencia y retraso al comenzar y durante la transmisión de paquetes. Mayor viabilidad en la innovación para la optimización de los recursos de red. Adopción a tiempo de nuevas tecnologías.
DESVENTAJAS	Complejidad en el desarrollo de conexiones innovadoras, Mayor consumo de parámetros en la red. Mayo dificultad al desplegar y controlar una red. Mayor costo de operación. Complejidad en la red. Retraso en el despliegue de nuevos servicios.	La programabilidad de la red afectaría la disponibilidad. Obligatoriedad en el desarrollo de técnicas de seguridad avanzadas e innovadoras. Definición muy joven, y aún cuenta con bajo. No todos los equipos pueden llegar a ser migrados a esta tecnología, según fabricantes.

Tabla 10: Análisis rápido sobre las ventajas y desventajas de las redes tradicionales y las redes definidas por software

Capítulo 5: Conclusiones

CONCLUSIONES

Las Redes definidas por software permiten el control del flujo de datos, definiendo parámetros sobre los dispositivos de la red asegurando mayo flexibilidad y garantizando rendimiento sobre la infraestructura de red.

Al interactuar con las redes SDN y las redes tradicionales, se logra evidenciar la acelerada velocidad de respuesta que presentan las redes SDN sobre la velocidad que logran mantener las redes tradicionales, aun así sobre las redes de área amplia.

La definición de redes definidas por software es muy joven donde con el protocolo Openflow están a la espera de aumentar su desarrollo práctico, con el fin de seguir descubriendo el límite de su alcance como propuesta tecnológica innovadora

Durante la emulación con los dos software, se resaltó la diferencia en el tamaño de los paquetes para la red sobre Packet tracer el paquete tiene un tamaño de 32 bytes, mientras que para las redes definidas por software, el paquete cuenta con un tamaño de 64 bytes. Según lo anterior, permite afirmar una vez más la ventaja que cuenta SDN con respecto a la aceleración en la entrega de paquetes, donde esto permite aumentar la disponibilidad de la red.

Al analizar el ancho de banda se puede determinar que cuenta con grandes variaciones en las redes tradicionales al encontrarse en medio de una transmisión de paquetes, mientras que sobre las redes SDN tiende a estandarizar el envío de paquetes a través del tiempo optimizando los recursos del ancho de banda.

A nivel de costo beneficio se puede concluir que las redes SDN brindan un costo relativamente económico a comparación de las redes tradicionales cuando se piense en desplegar una infraestructura de red, ya que debe contar con un controlador el cual realizaría todo el trabajo que para en el caso de las redes tradicionales tendrían que hacer cada uno de los dispositivos /routers a adquirir

El controlador sobre las redes SDN debe ser protegido de forma avanzada ya que es el cerebro de toda la red, por consiguiente un mal manejo de esta red, podría causar pérdidas desastrosas dentro de una compañía y hasta en la misma red, se debe tener presente que una red SDN puede contar con más de un controlador, y administradores de red sobre una red definida por software.

BIBLIOGRAFÍA

[1] U. Cauca, “Método de control cognitivo aplicado al enrutamiento de una red bajo arquitectura sdn/nfv,” 2018.

[2] A. Ballesteros, “Seguridad en redes definidas por software,” p. 62, 2016, [Online]. Available: https://rdu.iua.edu.ar/bitstream/123456789/1142/1/TF.pdf.

[3] E. H. Laos, “DEFINICIÓN DE UN API DE PROGRAMACIÓN PARALELA Y CONCURRENTE DE ALTO NIVEL SOBRE CLUSTERS DE ALTO DESEMPEÑO,” J. Chem. Inf. Model., vol. 12 Suppl 1, no. 9, pp. 1–29, 2005, doi: 10.1007/978-1-4614-7990-1.

[4] Universidad técnica de Ambato facultad de ingeniería en sistemas electrónica e industrial “red definida por software (SDN) en base a una infraestructura de software de libre distribución” 2015.

[5] D. Álvarez Amézquita, J. Padilla Herrera, A. Garzón Zuluaga, and L. Muñoz Hernández, “Proveedores de Servicios de Internet y de contenidos, responsabilidad civil y derechos de autor,” vol. 4, no. 3, pp. 51–64, 2009.

[6] S. Rodriguez Santamarìa, “Mecanismos de control de las comunicaciones en la internet del futuro a travès de Openflow” Universidad de Cantabria., 2012.

[7] A. L. Planas, “Configuración De Un Entorno De Emulación Que Permita El Diseño, Desarrollo Y Evaluación De Software- Defined Networks Con Calidad De Servicio,” Ing. en Informática, no. 2003, 2003.

[8] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, “A survey of active network research,” Communications Magazine, IEEE, vol. 35, no. 1, pp. 80–86, 1997

[9] M. Barrera, P. Neider, S. Losada, E. Rojas, and G. Mancilla, ESTADO DEL ARTE EN REDES DEFINIDAS POR SOFTWARE (SDN) STATE OF THE ART IN SOFTWARE-DEFINED NETWORKING. 1836.

[10] M. Ramírez and A. López, “Redes de datos definidas por software - SDN, arquitectura, componentes y funcionamiento,” J. Cienc. E Ing., vol. 10, no. 2145–2628, pp. 55–61, 2018, [Online]. Available: https://bit.ly/2DEBYVP.

[11] T. Cisco and Y. M. Tik, “ESTUDIO, EVALUACIÓN Y COMPARACIÓN DE PROTOCOLOS DE ENRUTAMIENTO EN TECNOLOGÍAS CISCO Y MIKRO-TIK,” 2016.

[12] I. E. Carisimo and I. J. I. Alvarez-hamelin, “Razones para usar SDNs Quienes somos CoNexDat es el grupo de Redes Complejas y Comunicación,” 2017.

[13] D. Bolatti et al., “Estudio de Herramientas de Simulación en Redes Definidas por Software,” pp. 176–180.

[14] U. P. D. E. Cartagena, “Programación de redes SDN mediante el controlador POX,” pp. 1–85, 2015.

[15] A. F. Ruiz, “Estado del Arte Redes Definidas Por Software,” Universidad Católica de Pereira, Pereira, 2014

[16] R. Thomas, L. DaSilva, and A. MacKenzie, “Cognitive networks,” in New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. First IEEE International Symposium on, Nov 2005, pp. 352–360.

[17] R.W. Thomas, D.H. Friend, L.A. DaSilva, and A.B. MacKenzie, “Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives,” IEEE Commun. Mag., vol. 44, n. 12, Dec. 2006

[18] mininet, "Mininet, An Instant Virtual Network on your Laptop (or other PC)," [Online]. Available: http://mininet.org/.

[19] ucb-sts, «STS - SDN troubleshooting simulator,» [En línea]. Available: http://ucbsts.github.io/sts/.

[20] V. project, "VirtualBox," ORACLE, [Online]. Available: https://www.virtualbox.org.

[21] V. project, «VMWare,» VMware, Inc, 2015. [En línea]. Available: https://www.vmware.com.

[22] mininet, “Mininet, Emulador para la creación rápida de prototipos de redes definidas por software” [Online]. Avaliable: https://github.com/mininet.

[23] J. Kempf, S. Whyte, J. Ellithorpe et al., “OpenFlow MPLS and the Open Source Label Switched Router,” in 23rd International Teletraffic Congress, 2011

[24] A. Blenk, A. Basta, and W. Kellerer, “HyperFlex: An SDN Virtual- ization Architecture with Flexible Hypervisor Function Allocation,” in IFIP/IEEE IM, 2015.

[25] “OpenFlow Table Type Patterns,” Open Networking Foundation, Speci- fication, Aug. 2014.

[26] N. Feamster, J. Rexford and E. Zegura, "The Road to SDN: An Intellectual History of Programmable Networks", SIGCOMM Comput. Commun. Rev., vol. 44, no. 2, pp. 87-98, 2014

[27] A. Doria, J. Hadi Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810

[28] D. Kreutz et al., ‘‘Software-defined networking: A comprehensive survey,’’ Proc. IEEE, vol. 103, no. 1, pp. 14–76, Jan. 2015

[29] Bruno Nunes Astuto, Marc Mendonça, Xuan Nam Nguyen, Katia Obraczka, Thierry Turletti, "A Survey of Software-Defined Networking: Past ,Present, and Future of Programmable Networks," IEEE Communications Surveys & Tutorials, 2013

[30] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat,

«PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric,» ACM SIGCOMM Computer Communication Review, vol. 39, 2009.