.UBAfiuba
FACULTAD DE INGENIERÍA

Introducción a los sistemas distribuidos
(75.43) Esteban Carisimo y Juan Ignacio Lopez Pecora

Trabajo Práctico 2
Software-Defined Networks

21 de noviembre de 2023

Integrantes:

- Agustina Bocaccio (106393)
- Nicolas Pinto (105064)
- Fernando Balmaceda (105525)
- Valentina Adelsflügel (108201)

Índice

1. Introducción

En este trabajo práctico se propuso el objetivo de adquirir conocimientos acerca de las Software-Defined Networks (SDN) y OpenFlow, haciendo uso de herramientas como Mininet, así como otras tecnologías pertinentes. La finalidad principal fue la construcción de una topología de red y la implementación de un firewall, como parte de la exploración y comprensión de estos conceptos avanzados en el ámbito de las redes de comunicación.

2. Hipótesis y suposiciones realizadas

- El mínimo de switches que puede tener la topología es uno.
- El firewall se ubicará en un sólo switch, por ejemplo si se configura en el del extremo izquierdo, sólo afectará el flujo entre hosts enfrentados (es decir, entre aquellos separados por un switch) o entre los hosts que estén del lado izquierdo.
- El firewall solo considerará protocolos de transporte UDP o TCP como posibles reglas de bloqueo para esa capa.

3. Implementación

3.1. Topología

Se realizó una topología parametrizable en la cual se tiene una cantidad variable de switches formando una cadena. Además en cada extremo se tienen dos hosts.

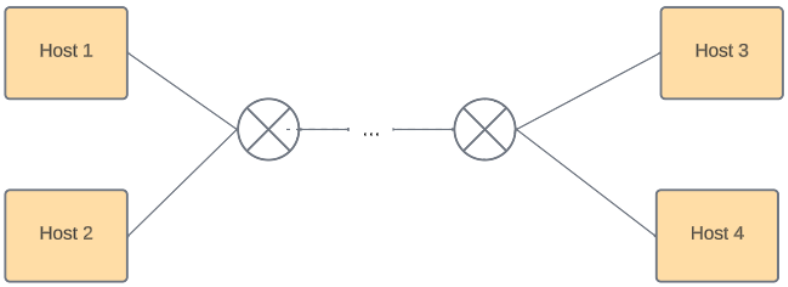

Figura 1: Topología

Para esto se utilizó Mininet, un emulador de red que permite crear y probar redes, y su API de Python: se crea una nueva clase llamada Tp2Topo que hereda de la clase Topo. Esta clase representa la topología específica que se desea crear en Mininet. El constructor de la clase toma un parámetro llamado n_switches, que indica el número de switches que se desean en la topología. Luego, se crean 4 hosts (h0, h1, h2, h3) y se crean switches según el valor de n_switches. h1 y h2 se conectan al primer switch (s1). h3 y h4 se conectan al último switch (s n_switches-1). Finalmente se establecen enlaces entre switches consecutivos (de s1 a s n_switches-1).

3.2. Firewall

Un Firewall es un componente de seguridad de red que monitorea el tráfico de red entrante y saliente y decide si permite el paso o bloquea un tráfico específico en función de un conjunto definido de reglas de seguridad. Los firewalls han sido una primera línea de defensa en seguridad de red durante más de 25 años. Establecen una barrera entre las redes internas seguras y controladas que pueden ser de confianza y las que no son de confianza fuera de las redes, como Internet. Un firewall puede ser hardware, software o ambos. En el caso de este trabajo, implementamos un

firewall utilizando software, más precisamente, a través de un controlador llamado POX. Las reglas de bloqueo definidas son:

1. Se deben descartar todos los mensajes cuyo puerto destino sea 80.

2. Se deben descartar todos los mensajes que provengan del host 1, tengan como puerto destino el 5001, y estén utilizando el protocolo UDP.

3. Se debe elegir dos hosts cualquiera, y los mismos no deben poder comunicarse de ninguna forma.

La API de POX fue utilizada de la siguiente manera:

- Importación de Módulos y Logger:

  Se importan los módulos necesarios de POX y otras librerías, como core de POX. Se inicializa un log para ir registrando errores, warnings e info.

- Clase Firewall y Métodos Asociados:

  Se define una clase llamada Firewall que hereda de EventMixin, lo que permite manejar eventos en POX. En el método __init__, se registra la clase para manejar eventos de conexión OpenFlow. Cuando los mensajes provienen del switch, aparecen en POX como eventos para los cuales puede escribir controladores de eventos. En el método _handle_ConnectionUp, se maneja el evento de conexión establecida. Se instalan reglas de firewall en el switch específico identificado por su DPID. Si un switch es “s3”, entonces su DPID será 3. Esto puede ser problemático cuando se usa con la opción –mac. La opción –mac asigna direcciones MAC a los hosts de la misma manera: si un host es ”h3”, entonces su MAC será 00:00:00:00:00:03. Esto significa que la parte del DPID que según la especificación OpenFlow está destinada a ser una dirección MAC es la misma que la dirección MAC de uno de los hosts. Esto puede ser confuso ya que generalmente se supone que las MAC son únicas.

- Configuración de Reglas:

  Se establece la configuración del firewall leyendo un archivo JSON llamado config.json. Se extrae el identificador del switch y las reglas desde el archivo de configuración.

- Instalación de Reglas en el Switch:

  Se instalan reglas en el switch, específicamente en el switch con dpid igual al indicado por configuración. Por ejemplo, se instala una regla que bloquea mensajes con destino al puerto 80 (tp_dst = 80). Se utilizó ofp_flow_mod para especificar las reglas que determinan cómo se deben procesar los paquetes en el switch. Algunos de los parámetros importantes que se pueden incluir en un mensaje ofp_flow_mod son:

  * match: Especifica las condiciones que deben cumplir los paquetes para que la regla se aplique. Puede incluir criterios como direcciones MAC, direcciones IP, puertos, etc.

  * actions: Indica las acciones que deben realizarse cuando un paquete coincide con las condiciones especificadas en el campo match. Puede incluir acciones como reenviar a un puerto específico, modificar campos del encabezado del paquete, descartar el paquete, etc. Al no indicarle action, el paquete se droppea.

- Método launch:

  Este método se utiliza para registrar la clase Firewall como controlador en POX.

  Los switches aprenden automáticamente la topología mediante l2_learning.

4. Ejecución

- Correr POX en una terminal con Python 2:

```
python2.7 pox.py log.level --DEBUG openflow.of_01 forwarding.l2_learning
Firewall
```

  Usar el código de la rama fangtooth.

- Levantar la topología en otra terminal, especificando cantidad de switches deseada (mayor a 0):

```
1        sudo mn --custom topology.py --topo tp2,n_switches=[n] --controller
     remote
2
```

- Una vez lograda la topología, se debe verificar el correcto funcionamiento de la red mediante el comando pingall.
- Correr iperf en host: Primero con xterm [nombre_host] (dentro de mininet) abrimos una terminal para dos hosts. Luego ahí dentro, ejecutamos iperf:

```
1        SERVIDOR: iperf -u -s -p [PUERTO]
2        CLIENTE: iperf -u -c [IP SERVIDOR] -p [PUERTO SERVIDOR]
3
```

  Donde -u es para UDP, -c es cliente, -p es para especificar el puerto y -s es para servidor.

  De ser necesario correr iperf con TCP, se deben agregar limitaciones:

```
1         iperf -c [IP SERVIDOR]  -p [PUERTO SERVIDOR] -b [x] -n [x] -l [x] -t
     [x]
2
```

  b - limita ancho de banda (ej 1Mb)

  n - limita cantidad de paquetes (ej 20)

  l - limita tamaño paquetes (ej 0 por ancho de banda)

  t - limita tiempo de conexion (ej 10 seg)

5. Pruebas

Para testear nuestro trabajo utilizamos Wireshark e Iperf: Iperf es una herramienta que se utiliza para hacer pruebas en redes informáticas. El funcionamiento habitual es crear flujos de datos TCP y UDP y medir el rendimiento de la red. De esta manera, logramos enviar datos de host un cliente a un host servidor y medir el rendimiento entre los dos extremos de la comunicación. Con Wireshark podemos visualizar los paquetes que se envían nuestros host, por lo tanto, cuando el firewall debe bloquear un envío, vemos que en la interfaz del host emisor están los paquetes enviados pero del lado del host receptor, no.

Mostramos en las siguientes secciones imágenes de evidencia para las 3 reglas configuradas:

5.1. Regla 1

En esta regla se deben descartar todos los mensajes cuyo puerto destino sea 80. Por lo tanto para probarlo, levantamos un servidor en un host, en este caso el host 1, que escuche en el puerto 80. Luego levantamos un cliente en el host 3 para que le envíe mensajes. Como el firewall filtra los paquetes que van al puerto 80, el host 1 nunca los recibe.

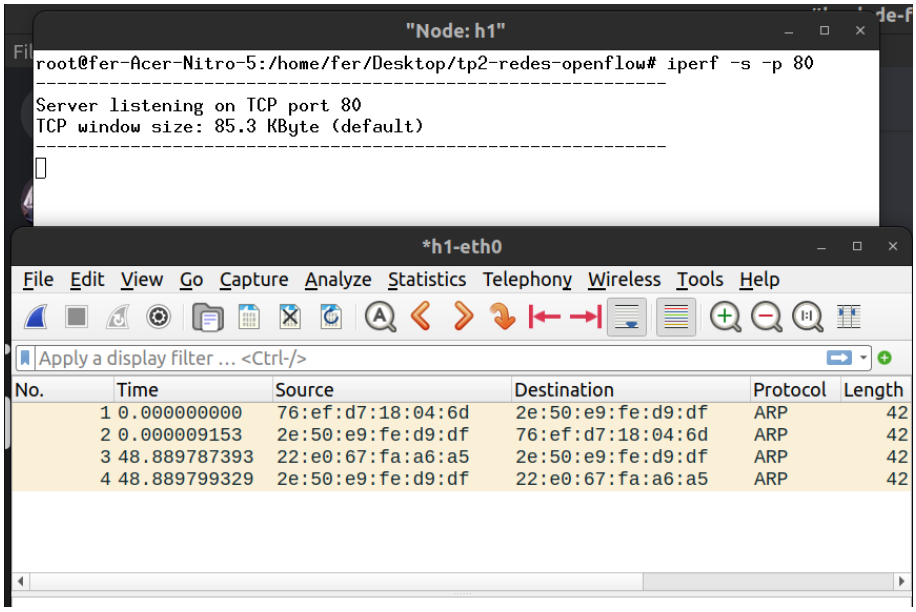

Figura 2: Servidor en host 1. La captura de wireshark evidencia que no le llegan los paquetes que le manda el host 3 porque tienen de destino el puerto 80.

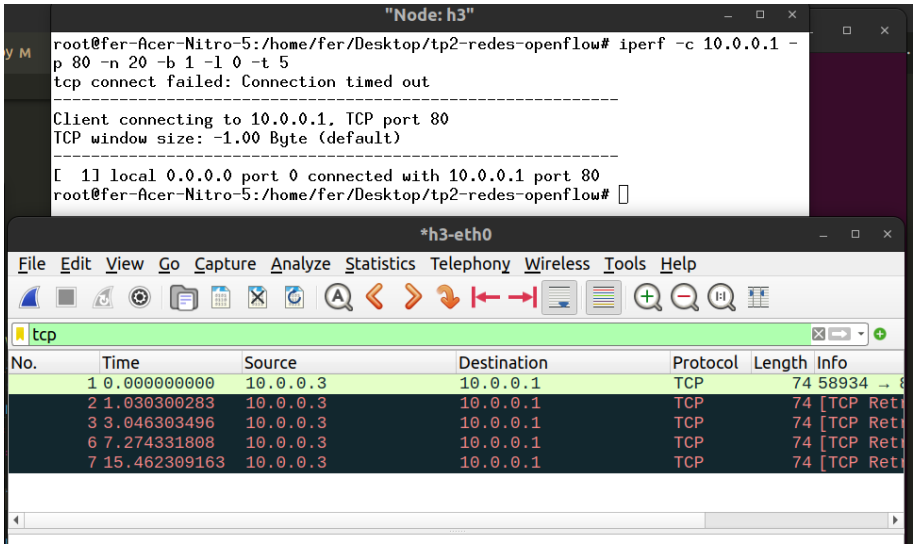

Figura 3: Cliente en host 3. La captura de wireshark evidencia que se realiza el envío de los paquetes al host 1.

5.2. Regla 2

En esta regla se deben descartar todos los paquetes que provengan del host 1, tengan como puerto destino el 5001, y estén utilizando el protocolo UDP. Por lo tanto para probarlo, levantamos un servidor en un host, en este caso el host 3, que escuche en el puerto 5001. Luego levantamos un cliente en el host 1 para que le envíe mensajes. El host 3 nunca va a recibir los paquetes.

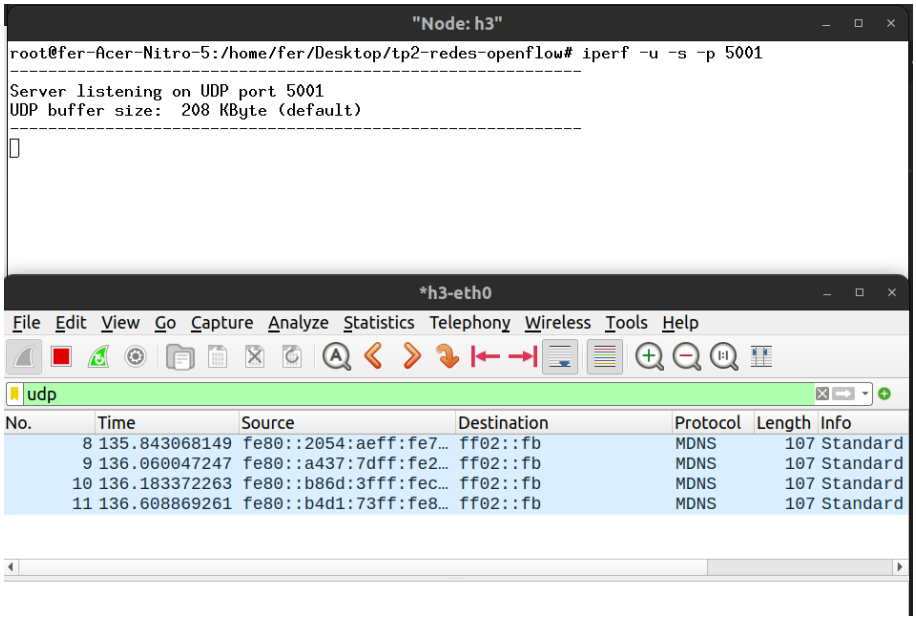

Figura 4: Servidor en host 3 escuchando del puerto 5001. La captura de wireshark evidencia que no le llegan los paquetes del cliente levantado en el host 1.

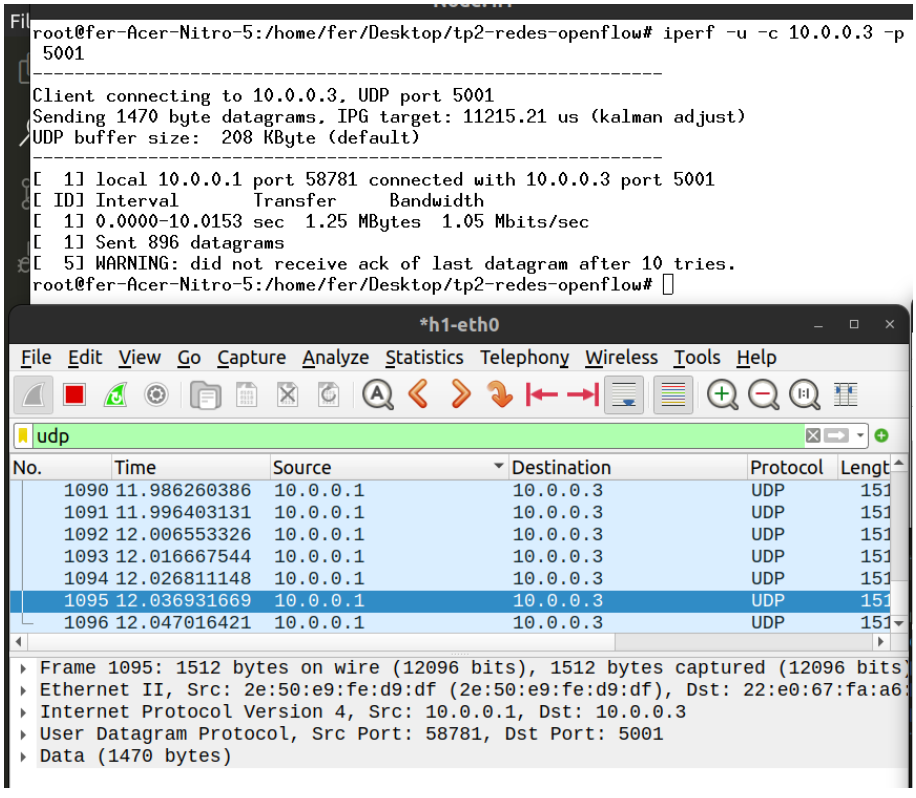

Figura 5: Cliente en host 1. La captura de wireshark evidencia que se realiza el envío de los paquetes al host 3.

5.3. Regla 3

En esta regla se debe elegir dos hosts cualquiera, y los mismos no deben poder comunicarse de ninguna forma. Por lo tanto para probarlo, elegimos los host 1 y 4. A continuación, levantamos un servidor en el host 4 y luego levantamos un cliente en el host 1 para que le envíe mensajes.

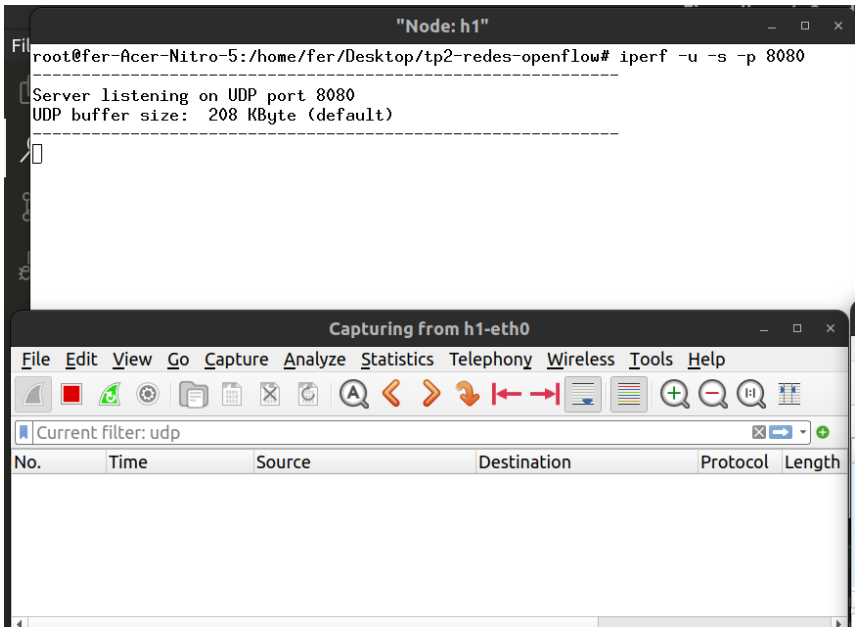

Figura 6: Servidor en host 1. La captura de wireshark evidencia que no le llegan los paquetes del cliente levantado en el host 4.

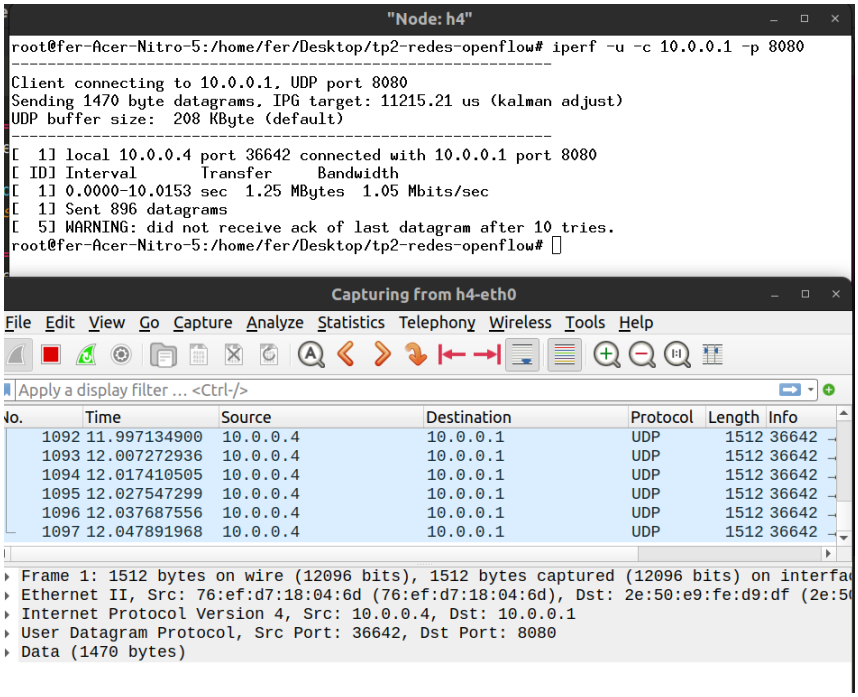

Figura 7: Cliente en host 4. La captura de wireshark evidencia que se realiza el envío de los paquetes al host 1.

6. Preguntas a responder

6.1. ¿Cuál es la diferencia entre un switch y un router? ¿Qué tienen en común?

Un switch y un router son elementos fundamentales en redes de computadoras, ya que ambos facilitan la conexión entre diversos dispositivos. Los routers son conmutadores de paquetes de almacenamiento y reenvío ("store-and-forward") que reenvían paquetes utilizando direcciones de capa de red. Aunque un switch también es un conmutador de almacenamiento y reenvío de paquetes, es fundamentalmente diferente de un router porque reenvía paquetes utilizando direcciones MAC. Mientras que un router es un conmutador de paquetes de capa 3, un switch es un conmutador de paquetes de capa 2. Los routers no son plug-and-play: ellos y los hosts que se conectan a ellos necesitan que sus direcciones IP estén configuradas. Además, los routers suelen tener un tiempo de procesamiento por paquete mayor que los switches, porque tienen que procesar hasta campos de capa 3.

6.2. ¿Cuál es la diferencia entre un Switch convencional y un Switch OpenFlow?

La diferencia clave entre un switch convencional y uno OpenFlow radica en cómo manejan el tráfico de red. Mientras que un switch convencional integra el plano de datos y el de control internamente, un switch OpenFlow establece comunicación a través de un canal dedicado hacia un controlador externo. Este controlador toma decisiones basadas en las acciones especificadas en las tablas de flujo del switch, brindando flexibilidad y programabilidad. El switch OpenFlow externaliza el plano de control, permitiendo decisiones más dinámicas y ofreciendo ventajas notables en entornos de redes definidas por software (SDN). Los switches que utilizan el estándar OpenFlow pueden realizar un reenvío generalizado de paquetes basado en cualquiera de los once campos diferentes de encabezado: frame, datagrama ip y capa de transporte. Los routers no tienen la restricción del árbol de expansión, y han permitido que Internet se construya con una rica topología que incluye, por ejemplo, múltiples enlaces activos entre Europa y América del Norte.

6.3. ¿Se pueden reemplazar todos los routers de la Intenet por Switches OpenFlow? Piense en el escenario interASes para elaborar su respuesta

Normalmente, las redes pequeñas que constan de unos cientos de hosts tienen unos pocos segmentos de LAN. Los switches son suficientes para estas pequeñas redes, ya que localizan el tráfico y aumentan el rendimiento agregado sin requerir ninguna configuración de direcciones IP. Sin embargo, las redes más grandes que constan de miles de hosts suelen incluir routers dentro de la red (además de switches). Los routers proporcionan un aislamiento más robusto del tráfico, controlan la transmisión de tormentas y utilizan rutas más "inteligentes" entre los hosts de la red. Podríamos enumerar los siguientes aspectos:

1- Económico: La transición hacia switches OpenFlow implicaría la sustitución masiva de una infraestructura existente, lo cual resultaría en costos significativos. Además, muchos equipos de red actuales no son compatibles con OpenFlow, lo que requeriría una inversión adicional.

2- Fiabilidad: La falta de pruebas en entornos de red a gran escala genera incertidumbre sobre la confiabilidad de los switches OpenFlow en situaciones del mundo real. La estabilidad y la capacidad de gestionar grandes volúmenes de tráfico son factores cruciales que aún deben ser validados.

3- Rendimiento: Mientras que los routers de borde están optimizados para el enrutamiento basado en IP, OpenFlow utiliza reglas para tomar decisiones de reenvío. Esto podría resultar en tablas de flujo sustancialmente extensas, lo que podría afectar el rendimiento al tener que consultar grandes conjuntos de reglas para determinar el destino del tráfico de Internet.

4- Seguridad: La centralización del control en un controlador maestro en OpenFlow plantea desafíos de seguridad. La posibilidad de intrusiones o manipulación de rutas por terceros podría ser

una vulnerabilidad potencial, especialmente en comparación con la diversidad de rutas y acuerdos de peering en la arquitectura actual de Internet.

7. Dificultades encontradas

La mayor dificultad fue la curva de aprendizaje de Pox y la instalación de Mininet. La primera porque tuvimos que leer la documentación a fondo para entender cómo se utilizaba la API para lo que queríamos hacer. La segunda porque habían varias formas de instalar Mininet. Con VM o directamente del repositorio de Github en Linux. Probamos ambas y la más sencilla fue la última.

8. Conclusión

En resumen, este informe detalla la implementación de una topología de red y un firewall utilizando Mininet y el controlador POX. Se exploraron conceptos claves de SDN y OpenFlow, destacando la flexibilidad y programabilidad de estas tecnologías.

Las pruebas realizadas con Wireshark e Iperf validaron el correcto funcionamiento del firewall, aunque se enfrentaron dificultades en la curva de aprendizaje de POX y la instalación de Mininet.

Se reflexionó sobre la viabilidad de reemplazar todos los routers de Internet con switches OpenFlow, considerando aspectos económicos, de fiabilidad y seguridad. Este análisis destaca los desafíos asociados con la transición a nuevas arquitecturas en entornos de red a gran escala.

En conclusión, este trabajo proporciona una visión práctica y reflexiva sobre la implementación de SDN y OpenFlow, demostrando su potencial y señalando desafíos clave en el campo de las redes de comunicación.