Introducción a los Sistemas Distribuidos (75.43)

Trabajo Práctico N°2

Software Defined Networks

1°C 2024

Integrantes del Grupo		
Nombre y apellido	Padrón	Correo
Rafael Berenguel	108225	rberenguel@fi.uba.ar
Juan Ignacio Biancuzzo	106005	jbiancuzzo@fi.uba.ar
Agustina Fraccaro	103199	afraccaro@fi.uba.ar
Sofia Javes	107677	sjaves@fi.uba.ar
Agustina Schmidt	103409	aschmidt@fi.uba.ar

Índice

1. Introducción

En el presente trabajo práctico se desarrolló una topología dinámica que junto con OpenFlow se implementó un Firewall a nivel capa de enlace. Para testear esto se utilizó la herramienta de mininet junto con iperf que imitan el funcionamiento y comportamiento de la topología que creamos junto con el envío de mensajes para verificar que efectivamente el Firewall construido cumpla su función: permitir o bloquear el paso de tráfico que cumplan con aquellas reglas que se nos propuso en el enunciado.

2. Implementación y Herramientas

2.1. Mininet

En el desarrollo del presente trabajo, con el objetivo de simular una red en donde pudiéramos controlar sus diversos componentes, hicimos uso de la herramienta **mininet**.

Mininet nos permite, con un script de python, tener definida una topología de red que luego podemos cargar fácilmente con un comando

```
mn --custom mininet_topology.py --topo redesTopo --mac --arp --switch ovsk
--controller remote
```

De esta forma se carga una topología personalizada definida en el archivo *miniet_topology.py* llamada *redesTopo*. Para este trabajo, la topología que definimos consiste en una cadena de switches de tamaño definido en un archivo de configuración, que tiene cada extremo de la cadena dos hosts conectados.

Hay un detalle del comando que se usó para levantar esta topología que en el contexto de este trabajo no es para nada menor. Normalmente, al definir una topología en mininet, definimos los hosts, los links y los switches, y al ejecutarla, se levanta un **controlador** default en el puerto 6653. Sin embargo, nosotros cuando lo ejecutamos hacemos uso de la opción *–controller remote*, la cual hace que no se genere ningún controlador default, y los switches se intentarán conectar a algún controlador en el puerto 6653, el cual nosotros instanciamos con la herramienta **POX**, descripta en la siguiente sección.

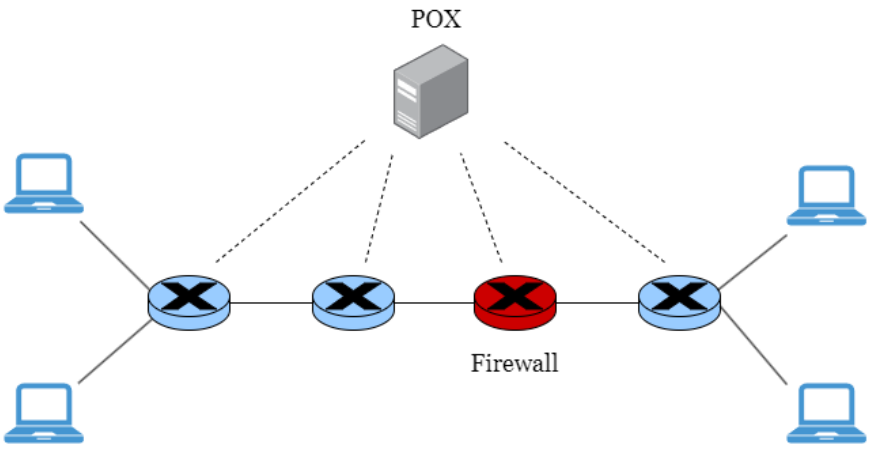

Figura 1: Switches en mininet conectados al controlador POX

2.2. Pox

POX es una libreria open-source de Python. Brinda una base para interactuar con OpenFlow, un protocolo de comunicación que permite el control de switches de red. Para implementar el Firewall, utilizamos la API que ofrece POX.

En primer lugar, inicializamos la clase Firewall que hereda de EventMixin, en el método *init* para poder escuchar distintos eventos. Guardamos referencia al archivo de configuración externo donde se indica, mediante un id, cuál de todos los switches será el Firewall, la cantidad de switches y las reglas.

Luego, intervenimos el evento *ConnectionUp*: En este punto se setean los parámetros provenientes del archivo: que switch es el Firewall y reglas. Para el caso del switch que tendrá el rol de Firewall, se setean las reglas preestablecidas.

Para instanciar las reglas, se utiliza *ofp_flow_mod*: sirve como la base para configurar y modificar las reglas de flujo dentro de los switches. Especifica los criterios que un paquete debe cumplir. En nuestra implementación, las posibles reglas pueden ser: protocolo, ipOrigen, puertoOrigen, ipDestino, puertoDestino.

Al iniciar el Firewall, se ejecuta el método *launch*, éste registra una nueva instancia de la clase Firewall en el framework POX.

2.3. Wireshark

Utilizamos Wireshark para realizar las pruebas del Firewall. Estas estarán detalladas en la sección de **Pruebas**.

2.4. IPerf

Utilizamos el comando de iperf para crear el flujo de mensajes y poder corroborar le funcionamiento del Firewall y las conexiones en la topología.

En este caso se utilizaron los siguientes comandos:

Para simular un servidor que sera aquel que reciba los mensajes lo hacemos con el siguiente comando:

```
# iperf -u -s -p [PUERTO]
```

Donde:

- `-u`: Mensajes UDP
- `-s`: Servidor
- `-p`: Especificación del puerto a conectarse

Cabe aclarar que sacando la flag `-u`, este usará mensajes TCP.

Para simular un cliente, será aquel que envíe los mensajes:

```
# iperf -u -c [IP SERVIDOR] -p [PUERTO SERVIDOR]
```

Donde:

- `-u`: Mensajes UDP
- `-c`: Cliente
- `-p`: Especificación del puerto del servidor a quien se conectará para enviarle los mensajes.

Cabe aclarar que sacando la flag `-u`, este usará mensajes TCP.

3. Pruebas

Para confirmar el correcto funcionamiento del Firewall se realizaron algunas pruebas con Wireshark en donde se capturaron paquetes con condiciones tanto que cumplen como que no cumplen las reglas establecidas.

En primer lugar recordemos la reglas:

- Se deben descartar todos los mensajes cuyo puerto destino sea 80
- Se deben descartar todos los mensajes que provengan del host 1, tengan como puerto destino 5001, y esten utiliando el protocolo UDP
- Se deben elegir 2 host cualquiera, y los mismos no deben poder comunicarse de ninguna forma. En nuestro caso nosotros elegimos el host 1 y el host 4.

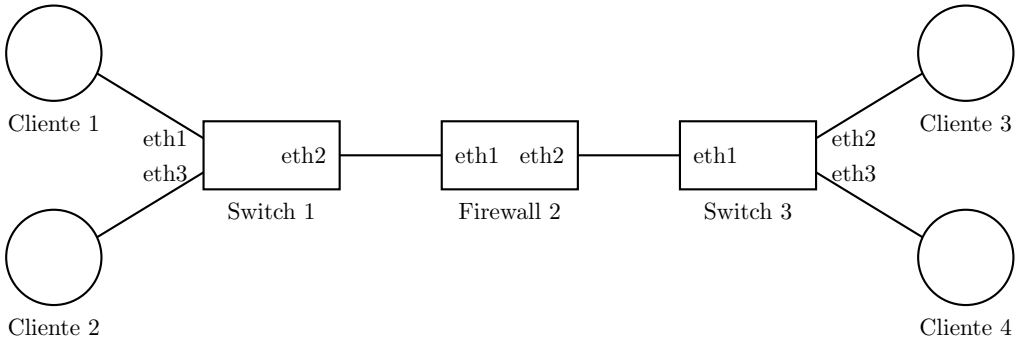

Figura 2: Topologia de ejemplo

En primer lugar corroboramos el correcto envio de mensajes entre el host 3 y host 2. Donde el host 2 en este caso sera el servidor y el host 3 el cliente que envia los mensajes. Como esto no cumple con ninguna regla establecida los mensajes deberian poder enviarse a traves del firewall sin problema:

Figura 3: Captura de mensajes de host 3 a host 2

En este caso se observa como los mensajes son transmitidos normalmente en la conexion que establecimos sin cortes ni problema alguno.

Ahora para ver al Firewall en accion decidimos enviar mensajes entre el host 1 al host 4. Que como bien podemos recordar, cumple con la regla 3 que enunciamos mas arriba. Es importante destacar que en esta instancia el Firewall se establecio en el switch 2:

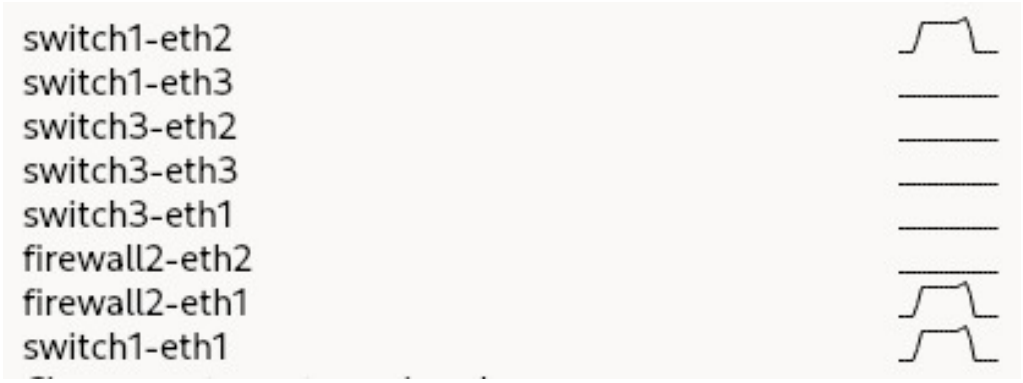

Figura 4: Captura de mensajes de host 1 a host 4

En esta imagen se observa como hay envío de mensajes entre el host 1 y el switch 1 (representado por la conexion: switch1-eth1), la salida de mensajes del switch 1 (representado por la conexion: switch1-eth2) y el recibo de mensajes del switch 1 en el Firewall (representado por la conexion: firewall2-eth1). En este caso como se cumple con una de las reglas el Firewall detendrá el flujo de mensajes cuando lleguen a él, es por esto que no hay flujo de mensajes en las conexiones subsiguientes: no hay salida de mensajes por parte del Firewall (representado por la conexion firewall2-eth2). Y las demás conexiones.

Si hicieramos el envío de mensajes a la inversa, el host 4 envía mensajes al host 1, el Firewall también deberia detener el envío de mensajes pero en este caso sera en la conexion firewall2-eth2. Observemos que es lo que sucede:

Figura 5: Captura de mensajes de host 4 a host 1

Efectivamente el host 4 envía mensajes al switch 3 (conexion switch 3-eth 3), hay salida de mensajes del switch 3 al firewall (conexion switch 3-eth1) y el firewall recibe los mensajes del switch 3 (firewall2-eth2). Pero no hay salida de estos mensajes, el Firewall efectivamente detuvo la transmisión.

Si observamos las 2 imágenes entre los host 1 y 4 son inversas. Lo que tiene sentido ya que estamos probando las conexiones en sentidos opuestos.

4. Preguntas a responder

¿Cuál es la diferencia entre un Switch y un Router? ¿Qué tienen en común?

Los switches y los routers cumplen funciones distintas:

Un switch conecta múltiples dispositivos dentro de la misma red local (LAN). La funcionalidad principal de este es enviar los datos solo a los dispositivos que es destinatario específicamente. Un switch opera en la capa de enlace. En este trabajo práctico para construir la topología dinámica que se pidió, utilizamos switches porque necesitábamos operar a nivel enlace.

Un router tiene una funcionalidad distinta a la de los switches y esta es la conexión de múltiples redes entres si, lo que incluye también la conexión de redes locales (LAN) a redes extensas como el Internet. Los routers intentan determinar la mejor ruta para enviar los datos transmitidos a destino. Un routes opera en la capa de red a diferencia de un switch.

En cuanto a las similitudes podemos decir que ambos utilizan tablas internas a la hora de la toma de decisiones sobre el direccionamiento de datos: un switch utiliza tablas de direcciones MAC (Media Access Control) y un router utiliza tablas de enrutamiento basadas en IP. Ambos dispositivos son configurables para cumplir con las necesidades específicas de una red, un switch tal puede ser configurado tal como hicimos en este trabajo práctico y un router puede ser configurado para tener políticas de enrutamiento especiales. Por ultimo, algunos switches que operan en la capa de red son compatibles con IP tal como los conmutadores por lo que cumplirían en este caso, funciones similares a los routers, y además operan en la misma capa.

Como podemos ver un switch y un router pueden parecer similares pero tienen funcionalidades distintas, tanto que el primero conecta dispositivos dentro de una misma red y el ultimo hace las conexiones de diferentes redes, la forma en la que funcionan son similares pero operan desde diferentes capas.

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

Un switch convencional ha sido definido en la pregunta anterior. Un switch OpenFlow tiene un control centralizado lo que significa que el reenvío de datos no es autónomo a diferencia de un switch convencional que si lo es, sino que esto se delega por lo general a un controlador centralizado que tiene la vista global de la red. Para poder comunicarse con el controlador los switches OpenFlow se comunican con éste mediante el protocolo de OpenFlow que establece las reglas de reenvío a seguir. Otra diferencia a destacar es la flexibilidad de los switches OpenFlow. Como los switches OpenFlow pueden configurarse dinámicamente a través del controlador estos son mas dinámicos a las políticas de red, permiten que se adapten rápidamente a las necesidades casi en tiempo real. Esto lo podemos ver en el trabajo que realizamos ya que podemos configurar nuestras propias reglas de Firewall en cualquier momento. Los switches convencionales son mas limitados en cuanto a esta característica.

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

No sería conveniente por varias razones. Considerando el escenario de interASes que son las conexiones entre sistemas autónomos, estos utilizan BGP (Border Gateway Protocol) para la comunicación de enrutamiento que se tiene entre sistemas autónomos. Esto es muy importante para la lograr una conectividad global. En cambio, como se mencionó antes, los switches OpenFlow dependen de una controlador centralizado que gestiona estas políticas de enrutamiento y reenvío de paquetes. Esto puede traer varias complicaciones ya que el controlador puede ser muy beneficioso para proveer felixibilidad y capacidad de programación pero no soportaría redes de gran escala tal como el Internet en sí. Otra cosa a tener en cuenta es la centralización, recordemos que si reemplazamos todos los

routers por Switches OpenFlow entonces dependeremos de un controlador por lo que habrá un único punto de falla (haciendo la suposición de que solo dependeremos de este y solo este controlador), si falla el controlador toda la red se verá afectada.

Es por estas razones las cuales no seria conveniente hacer el reemplazo de todos los routers de Internet por Switches OpenFlow por mas beneficiosos que estos sean.

5. Dificultades encontradas

En el proceso de determinar las reglas para el Firewall, nos encontramos en la situación de querer probar si funcionaba. La regla en cuestión que estábamos por probar era simplemente filtrar todos los mensajes que llegan del cliente 1.

Para probarlo, pensando que podría ser una solución rápida, decidimos usar el comando de ping, esperando que si hacíamos el siguiente comando dentro de mininet

```
mininet> cliente1 ping cliente4
```

Este sería bloqueado por el Firewall, mientras que si hacíamos el siguiente comando

```
mininet> cliente4 ping cliente1
```

No sería bloqueado, y por lo tanto veríamos que esta funcionando correctamente el Firewall.

A nuestra sorpresa, ambos casos casos fueron bloqueados. Esto nos bloqueo a nosotros, ya que no era lo que esperábamos.

Nos dimos cuenta que los mensajes de ping funcionan mandando y recibiendo para determinar que un mensaje llego correctamente. Entonces claramente hace que en ambos casos sea bloqueado los mensajes.

La solución es usar la herramienta correcta para probarlo, usar la **IPerf**, dándonos una mejor forma de probar si funcionaba.

6. Conclusión

En este trabajo práctico, hemos implementado y evaluado un Firewall a nivel de la capa de enlace utilizando una topología dinámica usando switches OpenFlow. Para llevar a cabo esta tarea, empleamos las herramientas Mininet, POX, Wireshark e iPerf. Nuestro objetivo principal fue verificar la funcionalidad del Firewall asegurando que las reglas establecidas se cumplieran correctamente, permitiendo o bloqueando el tráfico según lo especificado.

Al llevarlo a cabo, pudimos explorar de una manera más profunda las utilidades de mininet, permitiéndonos simular una red y elegir a qué controlador se iban a conectar los switches, dándonos la posibilidad de hacer uno personalizado y aprender cómo funciona un Firewall.

Wireshark nos permitió cerciorarnos del correcto funcionamiento de nuestro firewall y el uso de Iperf nos permitió simular situaciones reales de envío de mensajes.

En resumen, este trabajo práctico no solo nos permitió implementar y probar un Firewall en una red simulada, sino también profundizar en el conocimiento de las tecnologías y conceptos fundamentales en el ámbito de las redes definidas por software y la gestión del tráfico a nivel de la capa de enlace.