Estudio comparativo de la implementación de una regla de flujo para una red SDN mediante controladores POX y OpenDayLight respectivamente

Reinaldo Scappini
Universidad Tecnológica Nacional - FRRE
rscappini@gmail.com

Sergio Gramajo
Universidad Tecnológica Nacional - FRRE
sergiogramajo@gmail.com

Diego Bolatti
Universidad Tecnológica Nacional - FRRE
diegobolatti@gmail.com

Facundo Alarcon
Universidad Tecnológica Nacional - FRRE
facundoalarcon93@gmail.com

Resumen

El objetivo principal de este artículo es presentar una metodología sencilla para la simulación de una SDN (Software Defined Net-working) [1], configurada con el protocolo OpenFlow [2] controladores POX [3] y OpenDayLight (ODL) [4], utilizando la herramienta de simulación Mininet [5], realizar pruebas comparativas de implementación de una regla de flujo utilizando ambos controladores, y estudiar el comportamiento del protocolo mediante capturas de tráfico con el software Wireshark [6]

1. Introducción

El presente trabajo, está inserto en una línea de I/D presentada en la Universidad Tecnológica Nacional con código: UTN-2422. Título: "Modelo para la evaluación de performance mediante identificación de tráfico y atributos críticos en Redes Definidas por Software". Dicho proyecto se lleva a cabo en el ámbito del Dpto. de Ingeniería en Sistemas de Información perteneciente a la Facultad Regional Resistencia de la Universidad Tecnológica Nacional. Si bien el objetivo no radica en la descripción de tecnologías aplicadas en SDN, se brinda un breve contexto descriptivo del ámbito de trabajo, a los efectos de una mejor comprensión. Las Redes Definidas por Software proponen un modelo para cubrir nuevas demandas de usuarios y organizaciones. Ésta es una arquitectura de red emergente, donde el control de la infraestructura de red está desacoplado del reenvío de datos y, a su vez, es directamente programable. La inteligencia de red es (lógicamente) centralizada en controladores SDN basados en software que mantienen una visión global de la red. Como resultado, las organizaciones controlan la red independiente del proveedor en un único punto lógico lo que simplifica, en gran medida, el diseño de la red y su operación. Se pone énfasis en que el trabajo se enfoca directamente en la operación y ejecución de una regla de flujo aplicada a los conmutadores de una red, mediante un ejemplo simulado en una red SDN en un entorno de máquinas virtuales implementadas con el software VirtualBox [7]. Se da por entendido que el lector conoce la tecnología SDN y está familiarizado con la terminología y software utilizados para el mismo, por lo tanto, no se discuten detalles de instalación ni configuración de estos, siendo el principal objetivo mostrar concretamente los comandos en el entorno Mininet y el funcionamiento del protocolo OpenFlow, en el contexto de la red simulada.

El ejemplo muestra la implementación de una regla de flujo consistente en la denegación de tráfico para un host específico en una red SDN, simulada con el software Mininet, y controlada mediante un controlador POX y ODL respectivamente. Para mayor claridad, se incluyen capturas de pantalla mostrando la ejecución y el resultado de las distintas operaciones llevadas a cabo para la implementación de la regla de flujo.

2. Escenario de simulación

El trabajo descrito en este artículo se elaboró con un equipo de las siguientes características:

- Fabricante: Dell
- Modelo: Inspiron 15 Serie: 5000
- Procesador: Intel Core i7 (8th Gen)
- Memoria RAM: 16 GB
- Disco Rígido: 1 TB
- Sistema operativo: Windows 10 Pro-64 bits
- Virtualización: VMWare Workstation 15 PRO

Para la simulación se utiliza una máquina virtual con VMWare 15.0.4 Pro [8] y tres MV con las siguientes especificaciones:

Máquina Virtual VMWare del laboratorio:

- Memoria: 10 GB
- Procesadores: 4
- Espacio en disco: 50GB
- Sistema Operativo: Ubuntu 18.04

A su vez esta máquina virtual tenía un segundo nivel de virtualización en VirtualBox 6.0

Máquina Virtual VirtualBox Mininet:

- Memoria: 2GB
- Procesadores: 2GB
- Especio en disco: 10 GB
- Sistema Operativo: Ubuntu Server 18.04

Máquina Virtual VirtualBox OpenDaylight:

- Memoria: 2 GB
- Procesadores: 2
- Espacio en disco: 10 GB
- Sistema Operativo: Ubuntu Server 14.04

2.1. Simulación con el controlador POX

Se inician cuatro sesiones SSH en la MV Mininet. Cada sesión SSH de la MV Mininet (en el ejemplo, la máquina tiene la IP 192.168.59.2), se ejecuta el comando:

sudo ssh -X mininet@192.168.59.2 En las cuatro sesiones.

Para comprobar la versión de Mininet, una vez dentro de la MV podemos ejecutar el comando:

sudo mn --version

En la primera sesión se ejecuta el comando **sudo wireshark &**, y se selecciona la interfaz loopback**,** esto permite iniciar la captura de todos los paquetes intercambiados en la MV.

En la segunda sesión se arranca el controlador POX, ejecutando el comando:

sudo ~/pox/pox.py forwarding.l2_pairs info.packet_dump samples.pretty_log log.level --DEBUG. (Figura 1).

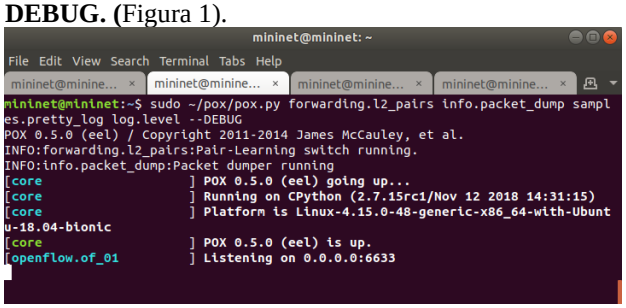


Figura 1 - Controlador POX corriendo en el puerto 6633

Se inicia la simulación en Mininet en otra sesión SSH de la misma MV, una vez allí se crea una nueva topología con el siguiente comando:

sudo mn --controller=remote,ip=127.0.0.1:6633 --topo tree,2,2

Esto crea topología de árbol, con dos niveles y dos hosts por cada switch como muestra la figura 2.

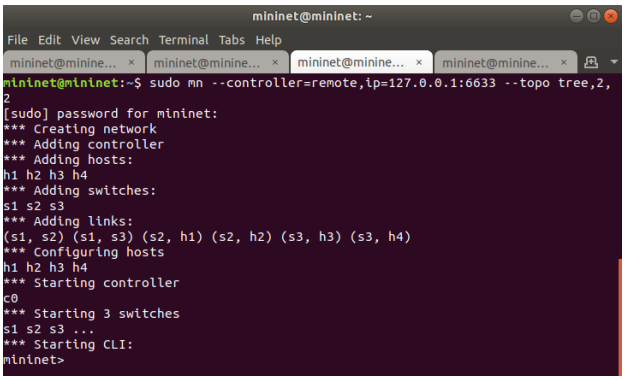


Figura 2 - Topología simulada en Mininet

Se ejecuta el comando

mininet> net

Se obtiene lo mostrado en la Figura 3

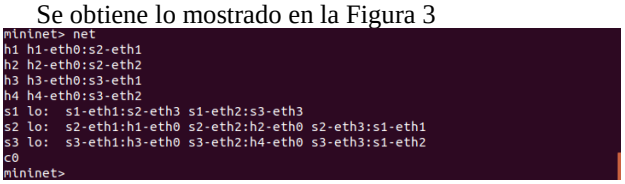


Figura 3 - Resultado del comando “net”

La topología creada es la que muestra la Figura 4

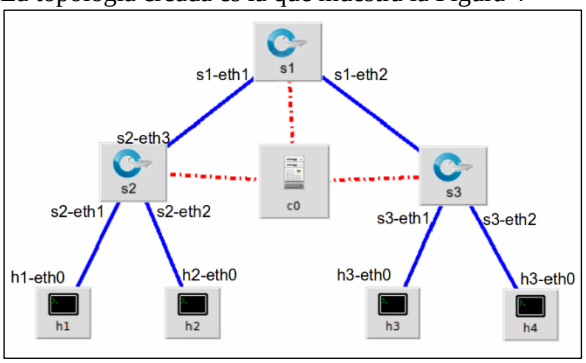


Figura 4 - Topología de la red SDN simulada con Mininet

Por último, la cuarta terminal se utiliza para ejecutar comandos ovs [9]; desde fuera del cli Mininet.

Ahora bien, si se configura Wireshark para que capture el tráfico a través de la interfaz de loopback de la MV Mininet, y se aplica el filtro openflow_v1, que es el protocolo con el que trabaja el controlador POX (es el que viene incorporado en la MV Mininet V2.2.), se observa el intercambio de mensajes entre dicho controlador y los switches desde el inicio de sesión, tal como muestra la figura 5.

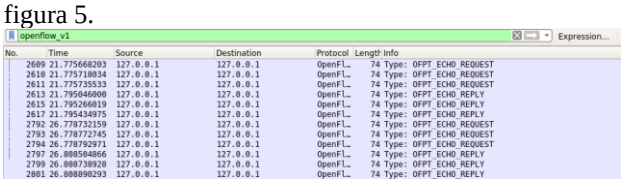


Figura 5 - Intercambio de mensajes OpenFlow V1

Se puede observar los flujos instalados en el simulador Mininet ejecutando el comando:

mininet> dpctl dump-flows (Figura 6)

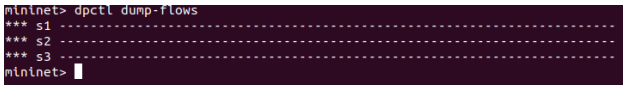


Figura 6 - Salida del comando dpctl dump-flows

La figura 6 muestra que no existe ningún flujo establecido, Se ejecuta el comando **h1 ping h2**, limitando la cantidad de mensajes a dos paquetes utilizando el comando:

mininet> h1 ping -c 2 h2 (Figura 7)

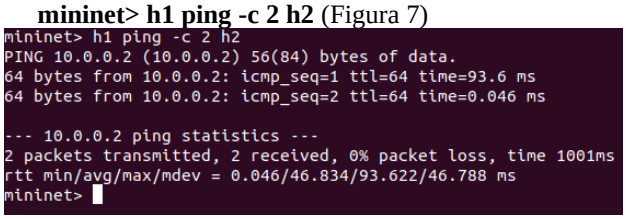


Figura 7 - Resultado del comando ping h1 ping -c2 h2

Se ejecuta el comando **dpctl dump-flows** desde el cli mininet y se ven los flujos, tal como muestra la Figura 8

Figura 8 - Flujos en los switch (comando dpctl dump flow)

Como se trata de un switch del tipo learning, al hacer ping, automáticamente se crea la regla de reenvío de esos paquetes. Como el switch se encuentra ejecutando en el kernel de la MV (**ovs**), también es posible visualizar los flujos mediante el comando **sh ovs-ofctl dump-flows s2** desde el prompt de la MV Mininet. (Figura 9).

Figura 9 - Flujos en el switch S2

Lo mismo se verifica para los switches restantes, el switch s1 y el switch s3. Notar que Donde **s1** es el nombre del switch. Por otro lado, también se puede especificar el protocolo incorporando el parámetro **-O OpenFlow1X** donde la **X** representa la versión de OpenFlow que se utiliza, si no se utiliza **-O**, indica que la versión del protocolo es por defecto, en el ejemplo mostrado es versión 1.

La figura 10 muestra una captura del paquete que envía el switch al controlador para iniciar el proceso **"PACKET IN".** De la misma manera, en el siguiente paquete se puede ver en detalle el paquete "PAQKET OUT".

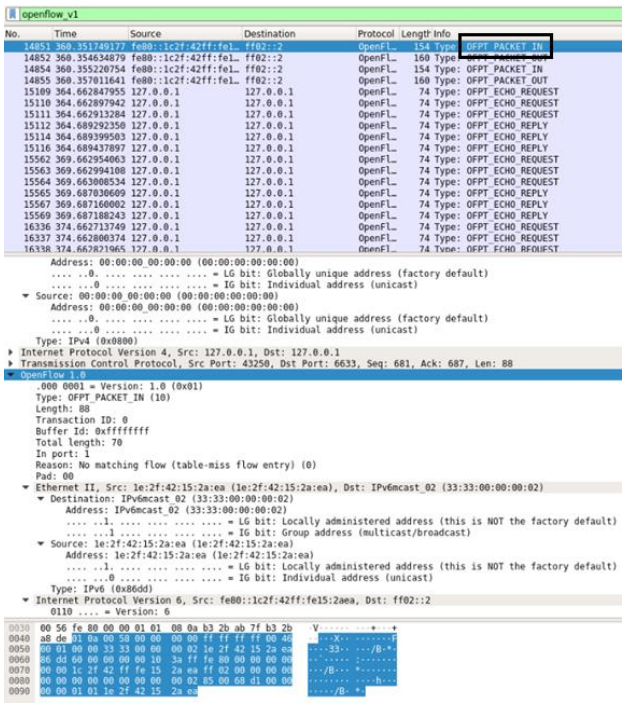


Figura 10 - Captura de tráfico "PACKET IN"

En la sesión donde está el simulador Mininet, se ejecuta el comando **h2 ifconfig** que muestra la configuración de h2 (Figura 11).

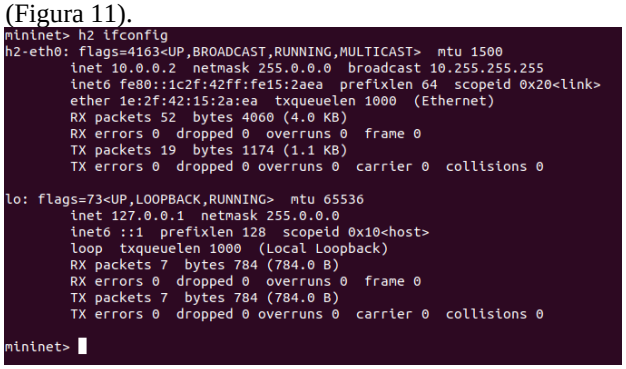


Figura 11 - Salida del comando h2 ifconfig

Con la información proporcionada anteriormente, en la sesión donde se ejecutan los comandos ovs, se procede a borrar todos los flujos del switch s2, mediante el comando **sudo ovs-ofctl del-flow s2,** luego se instala la siguiente regla de flujo mediante el comando **sudo ovs-ofctl add-flow s2 dl_src= 1e:2f:42:15:2a:ea,actions=drop**

A continuación, con el comando: **mininet> dpctl dump-flows,** se puede ver solo el flujo instalado, esto se debe a que anteriormente borraron todos los flujos (Figura 12).

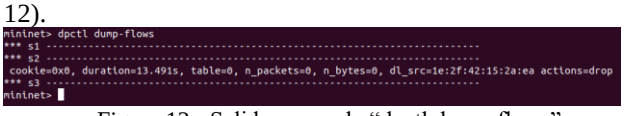


Figura 12 - Salida comando "dpctl dump-flows"

Si en estas condiciones se ejecuta h2 ping h1, se observa que los paquetes no llegan debido a que

denegamos el tráfico a cualquier host que contenga la dirección MAC de h2 (Figura 13).

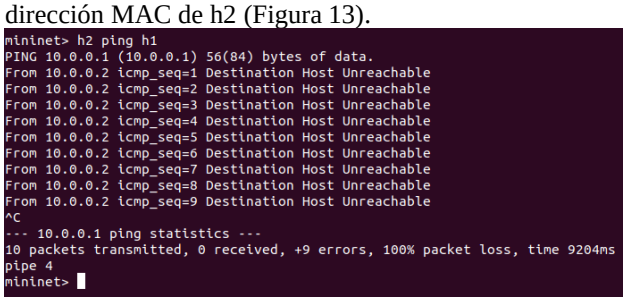


Figura 13 - Salida comando h2 ping h1

Ahora se ejecuta h1 ping h4, y se observa que los paquetes llegan correctamente (Figura 14)..

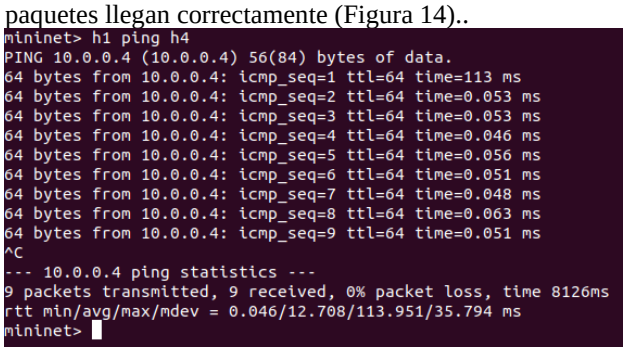


Figura 14 - Resultado de h1 ping h4

Se puede comprobar el estado de los flujos mediante el comando: **mininet> sh ovs-ofctl -O OpenFlow11 dump-flows s2** (Figura 15).

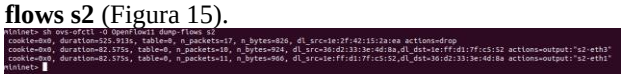


Figura 15 - Estado de los flujos instalados

2.2. Simulación con el controlador OpenDayLight

Para la simulación con el controlador OpenDaylight (se mantienen las cuatro sesiones utilizadas anteriormente abiertas, se procede a cerrar el controlador POX y seestablece una sesión SSH con la MV que contiene el controlador ODL con el siguiente comando:

sudo ssh -X odl@192.168.59.10

Se ejecuta el comando para arrancar el servidor ODL(Figura 16), desde la ubicación donde se encuentra, en nuestro caso:

./distribution-karaf-0.4.4-Beryllium-SR4/bin/karaf

Figura 16 - Interfaz del servidor ODL

Desde la sesión SSH conectada a mininet se crea la misma topología utilizada con el controlador POX con el siguiente comando:

sudo mn --topo tree,2,2 --mac --controller=remote,ip=192.168.59.10 --switch ovsk,protocols=OpenFlow13

Al crear la topología desde mininet utilizando el controlador ODL automáticamente se crean reglas de flujo para los switches (Figura 17). Estos se pueden ver desde Mininet, con el comando:

dpctl dump-flows -O OpenFlow13.

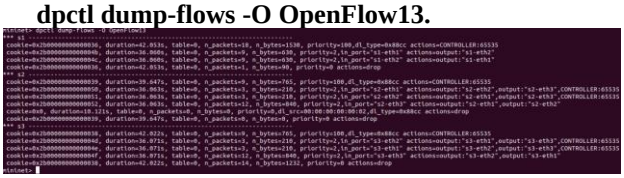


Figura 17 - Reglas de flujo creadas al arrancar la topología

Para crear la misma regla de flujo utilizada en la simulación con POX sobre h2, se procede en primer lugar a ver su configuración de red con el comando:

mininet> h2 ifconfig (Figura 18).

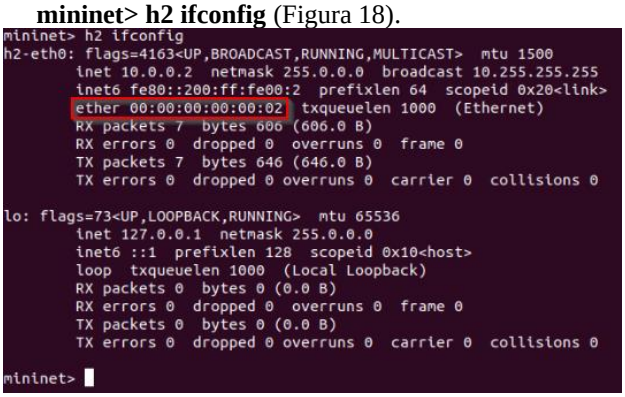


Figura 18 - Detalles del host h2

La dirección ethernet es 00:00:00:00:00:00:02. Teniendo en cuenta esto se procede a crear una regla de flujo en la interfaz gráfica de ODL. Desde el navegador se accede a la interfaz web **DLUX de ODL** ingresando a:

http://192.168.59.10:8181/index.html.

El siguiente paso consiste en dirigirnos a YANG UI, a partir del panel de la izquierda. Este apartado tiene múltiples secciones: **API, HISTORY, COLLECTION y PARAMETERS.** A continuación, se desplaza hasta la sección **opendaylight-inventory**. Una vez allí, se observa que existen dos configuraciones, una llamada **operational** y la otra **config**. La primera lee la configuración directamente de los switches, por tanto, solo podremos hacer una solicitud **GET**. La segunda tiene las configuraciones existentes del controlador, por ende, en principio no tendremos ninguna, aquí se puede hacer peticiones **GET, PUT, POST y DELETE**. Inicialmente la idea es añadir configuración de flujo nuevo a la existente en operational, ya que ambas deben coexistir.

Para añadir una regla de flujo nueva, desde **config** nos movemos hacia nodes y por último a **node {id}.**

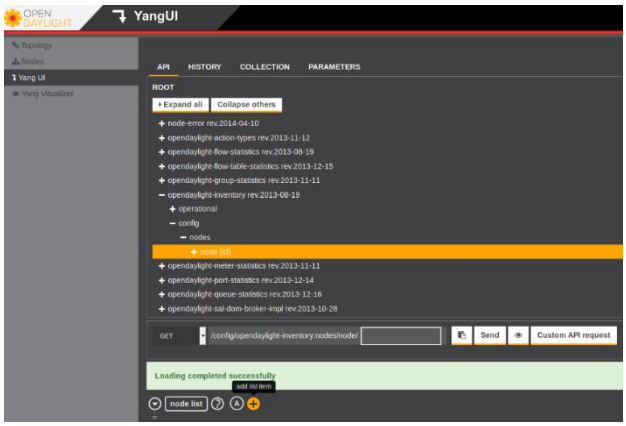


Figura 19 - Interfaz YANGUI

Como muestra la Figura 19, se puede ver un botón con el símbolo "+". Este sirve para añadir un nodo en la lista de **config**. En la lista desplegable se selecciona **PUT**, para crear una nueva entrada en la tabla de flujo (Figura 20).

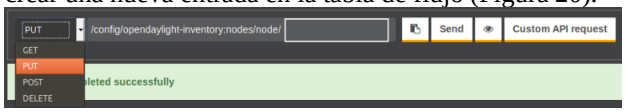


Figura 20 - Creando una entrada de regla de flujo

Para añadir una regla de flujo con el mismo efecto que la simulación con POX, se debe crear una regla sobre el switch 2 (Figura 21), el mismo es el nodo llamado openflow:2, de esta forma lo acoplaremos a las reglas que se encuentra en **operational** creadas al utilizar ODL.

Figura 21 - Lista de Nodos

Para denegar el tráfico al host h2, en la sección **address de ethernet-source** se coloca la dirección ethernet de este (Figura 22)

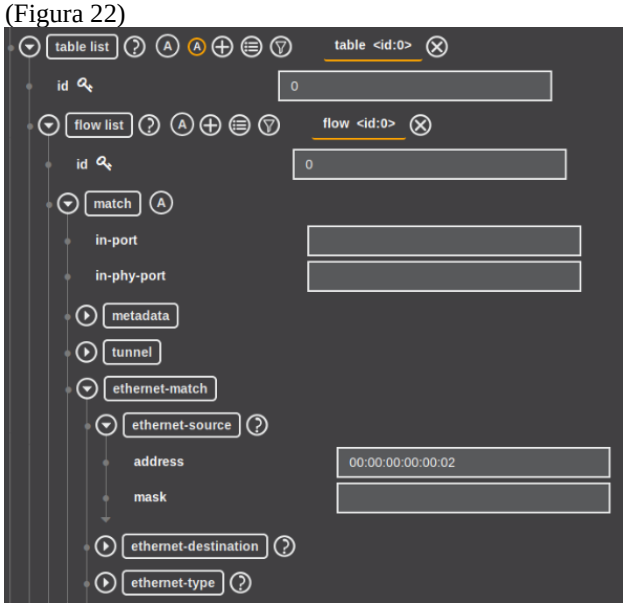


Figura 22 - Identificación de dirección MAC

A continuación, se configura la instrucción para cuando el flujo coincida con la dirección previamente indicada. Para esto nos dirigimos a **instructions** y se configura como muestra la Figura 23.

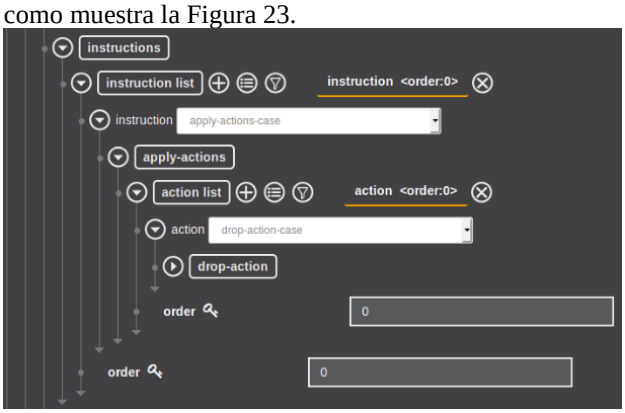


Figura 23 - Configuración de la regla de flujo

La instrucción **apply-actions-case**¸ aplica la acción definida inmediatamente. Esta acción al ser **drop-action-case** descarta los paquetes con la coincidencia anteriormente indicada. Posteriormente se configura la prioridad, el tiempo de inactividad, el tiempo de espera, la cookie y el id de la tabla tal como lo establece el protocolo (Figura 24).

Figura 24 - Parámetros de la regla de flujo

La prioridad define características de QoS al flujo, mientras mayor sea la prioridad mayor precedencia tendrá y se ejecutará primero. El tiempo de inactividad, denotado por **idle-timeout**, indica el número de segundos después de los cuales se elimina una entrada de flujo de la tabla de flujo porque no hay paquetes que lo coincidan, si se fija el tiempo de inactividad en 0, la entrada de flujo no experimenta tiempo de inactividad. El tiempo de espera, denotado por **hard-timeout**, indica el número de segundos después de los cuales se elimina la entrada de flujo de la tabla de flujo, ya sea que los paquetes coincidan o no, si se establece el tiempo de espera en 0, la entrada de flujo no experimenta tiempo de espera. A su vez, con la cookie, se identifica nuestro flujo en la tabla y, para completar la configuración, se indica el id de la tabla a la cual se aplica. Como último paso se envía esta petición al servidor ODL presionando el botón **Send** (Figura 25).

Figura 25 - Aplicando la regla de flujo

Si la solicitud es ejecutada satisfactoriamente, aparece un mensaje indicando esto. En estas condiciones, se

procede a observar las tablas de flujo de los switches desde mininet (Figura 26).

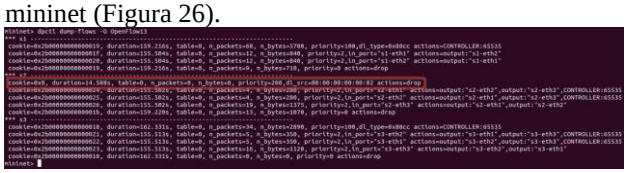
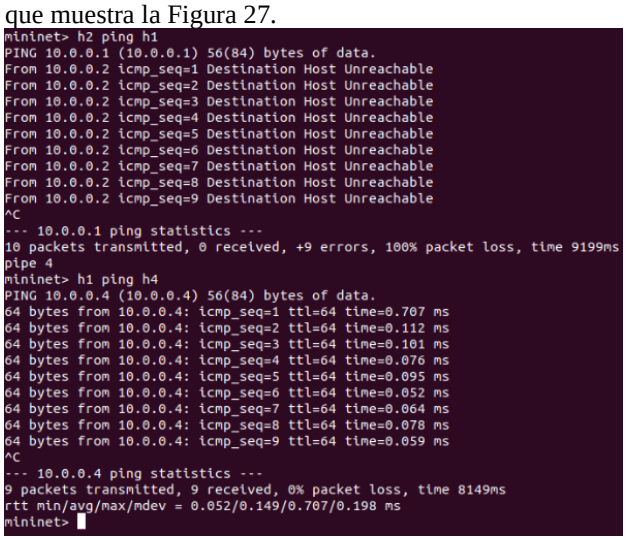


Figura 26 - Reglas de flujo instaladas en los nodos

Es de notar que mientras mayor sea la prioridad de la regla de flujo instalada aparecerá en primer lugar en la tabla. Al realizar las pruebas de conectividad se observa lo que muestra la Figura 27.



```
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
From 10.0.0.2 icmp_seq=1 Destination Host Unreachable
From 10.0.0.2 icmp_seq=2 Destination Host Unreachable
From 10.0.0.2 icmp_seq=3 Destination Host Unreachable
From 10.0.0.2 icmp_seq=4 Destination Host Unreachable
From 10.0.0.2 icmp_seq=5 Destination Host Unreachable
From 10.0.0.2 icmp_seq=6 Destination Host Unreachable
From 10.0.0.2 icmp_seq=7 Destination Host Unreachable
From 10.0.0.2 icmp_seq=8 Destination Host Unreachable
From 10.0.0.2 icmp_seq=9 Destination Host Unreachable
^C
--- 10.0.0.1 ping statistics ---
10 packets transmitted, 0 received, +9 errors, 100% packet loss, time 9199ms
pipe 4
mininet> h1 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=0.707 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=0.112 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=0.101 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=0.076 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=0.095 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=0.064 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=0.078 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=0.059 ms
^C
--- 10.0.0.4 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8149ms
rtt min/avg/max/mdev = 0.052/0.149/0.707/0.198 ms
mininet>
```

Figura 27 - Resultado de la prueba de conectividad

Los resultados son los mismos que en los vistos con POX

3. Conclusiones

Este trabajo muestra en detalle la implementación de una regla de flujo denegando el tráfico a un nodo dentro de una red SDN, y lo hace mostrando la utilización de dos controladores disponibles bajo licencia GNU, se considera que de esta forma se facilita la reproducción de la experiencia y es un aporte para el estudio del protocolo OpenFlow

4. Referencias

[1] Open Networking Foundation, «Software-Defined Networking (SDN) Definition,» 2018. [En línea]. Available: https://www.opennetworking.org/sdn-definition/. [Último acceso: 6 11 2018].

[2] The Open Networking Foundation,, «openflow-spec-v1.3.0.pdf,» [En línea]. Available: https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf. [Último acceso: 5 11 2018].

[3] github.com att/pox, «ATT/POX,» github.com, 2018. [En línea]. Available: https://github.com/att/pox. [Último acceso: 18 11 2018].

[4] opendaylight.org, «OpenDaylight,» OpenDaylight Project, 2016. [En línea]. Available: https://www.opendaylight.org/. [Último acceso: 18 11 2018].

[5] Mininet Team, «Mininet,» Mininet Team, 2018. [En línea]. Available: http://mininet.org/. [Último acceso: 18 11 2018].

[6] Wireshark Foundation, «Wireshark,» 209. [En línea]. Available: https://www.wireshark.org/. [Último acceso: 5 7 2019].

[7] VirtualBox.org, «VirtualBox,» 2018. [En línea]. Available: https://www.virtualbox.org/. [Último acceso: 6 11 2018].

[8] VMware, Inc, «VMware Workstation 15.0.4 Pro Release Notes,» 2019. [En línea]. Available: https://docs.vmware.com/en/VMware-Workstation-Pro/15/rn/workstation-1504-release-notes.html. [Último acceso: 18 7 2019].

[9] openvswitch.org, «ovs-dpctl.8,» [En línea]. Available: http://www.openvswitch.org/support/dist-docs/ovs-dpctl.8.txt. [Último acceso: 19 11 2018].