

LABORATORIO #4: ACTIVIDAD 2 – SDN Y OPENFLOW

1. OBJETIVO (S)

El Software Defined Networking es una arquitectura de red emergente en la que se le da el control a una aplicación software llamada controlador. El término Software Defined Network (red definida por software) hace referencia a esta arquitectura en la cual se permite separar el plano de control del plano de datos con el objetivo de conseguir redes más programables, flexibles y automatizables. Esta migración del control, permite que la infraestructura subyacente sea abstraída para que aplicaciones y servicios de red puedan tratar a la red como una entidad virtual. El controlador de una SDN es una entidad centralizada, es decir, que puede tratarse de varias instancias físicas o virtuales, pero se comporta como un único componente, que mantendrá un estado global de la red (o fragmento de ésta), permitiendo que empresas y operadores ganen control sobre toda la red desde un único punto lógico, simplificando enormemente el diseño y operación.

Al finalizar la práctica, el estudiante estará en capacidad de:

- Operar una máquina virtual con mininet y controladores externos.
- Entender el funcionamiento de una SDN (Software Defined Network).
- Desarrollar práctica de laboratorio con topologías virtuales para el análisis del protocolo OpenFlow.
- Analizar la interacción entre redes tradicionales y redes definidas por software.

2. LECTURAS PREVIAS

Lecturas recomendadas:

- Software Defined Networking with OpenFlow: Get Hands-on with the Platforms and Development Tools Used to Build OpenFlow Network Applications [1]
- Virtualized Software-defined Networks and Services [2]
- Implementación Software-Defined Networks and OpenFlow - The Internet Protocol Journal, Volume 16, No. 1 [3]
- Introduction to SDN (Software Defined Networking) [4]

3. INFORMACIÓN BÁSICA

Redes Tradicionales [4]

Las redes siempre han sido muy tradicionales. Tenemos dispositivos de red específicos como enrutadores, conmutadores y firewalls que se utilizan para tareas específicas.

Estos dispositivos de red son vendidos por proveedores de redes como Cisco y, a menudo, usan hardware propietario. La mayoría de estos dispositivos se configuran principalmente a través de la CLI, aunque hay algunos productos de GUI como CCP (protocolo de configuración de Cisco) para los enrutadores o ASDM para los firewalls ASA de Cisco.

Un dispositivo de red, por ejemplo, un enrutador tiene diferentes funciones que tiene que realizar. Piense por un momento en algunas de las cosas que debe hacer un enrutador para reenviar un paquete de IP:

- Tiene que verificar la dirección IP de destino en la tabla de enrutamiento para averiguar a dónde reenviar el paquete IP.
- Los protocolos de enrutamiento como RIP, OSPF o BGP son necesarios para conocer las redes que están instaladas en la tabla de enrutamiento.
- Tiene que usar ARP para determinar la dirección MAC de destino del próximo salto o destino y cambiar la dirección MAC de destino en el marco de Ethernet.
- El TTL (Time to Live) en el paquete IP debe reducirse en 1 y la suma de comprobación del encabezado IP debe ser recalculada.
- La suma de comprobación del marco de Ethernet debe ser recalculada.

Todas estas tareas diferentes están separadas por diferentes planos. Hay tres planos:

- Plano de control
- Plano de datos
- Plano de gestión

Plano de control

El plano de control es responsable de intercambiar información de ruta, construir la tabla ARP, etc. Aquí hay algunas tareas que realiza el plano de control:

- Aprendizaje de direcciones MAC para construir una tabla de direcciones MAC de cambio.
- Ejecutando STP para crear una topología libre de bucles.
- Construyendo tablas ARP.
- Ejecución de protocolos de enrutamiento como RIP, OSPF y BGP, y creación de la tabla de enrutamiento.

Plano de datos

El plano de datos es responsable de reenviar el tráfico. Se basa en la información que proporciona el plano de control. Aquí hay algunas tareas que el plano de datos se ocupa:

- Encapsular y desencapsular paquetes.
- Agregar o eliminar encabezados como el encabezado 802.1Q.
- Coincidencia de direcciones MAC para el reenvío.
- Coincidencia de destinos IP en la tabla de enrutamiento.
- Cambie las direcciones de origen y destino cuando use NAT.

- Eliminar el tráfico debido a las listas de acceso.
- Las tareas del plano de datos deben realizarse lo más rápido posible, por lo que el reenvío del tráfico se realiza mediante hardware especializado como ASIC y tablas TCAM.

Plano de gestión

El plano de gestión/administración se usa para acceder y administrar nuestros dispositivos de red. Por ejemplo, acceder a nuestro dispositivo a través de telnet, SSH o el puerto de la consola.

Cuando se habla de SDN, el plano de control y el plano de datos son los más importantes a tener en cuenta. Aquí hay una ilustración del plano de control y datos para ayudarlo a visualizar los diferentes planos:

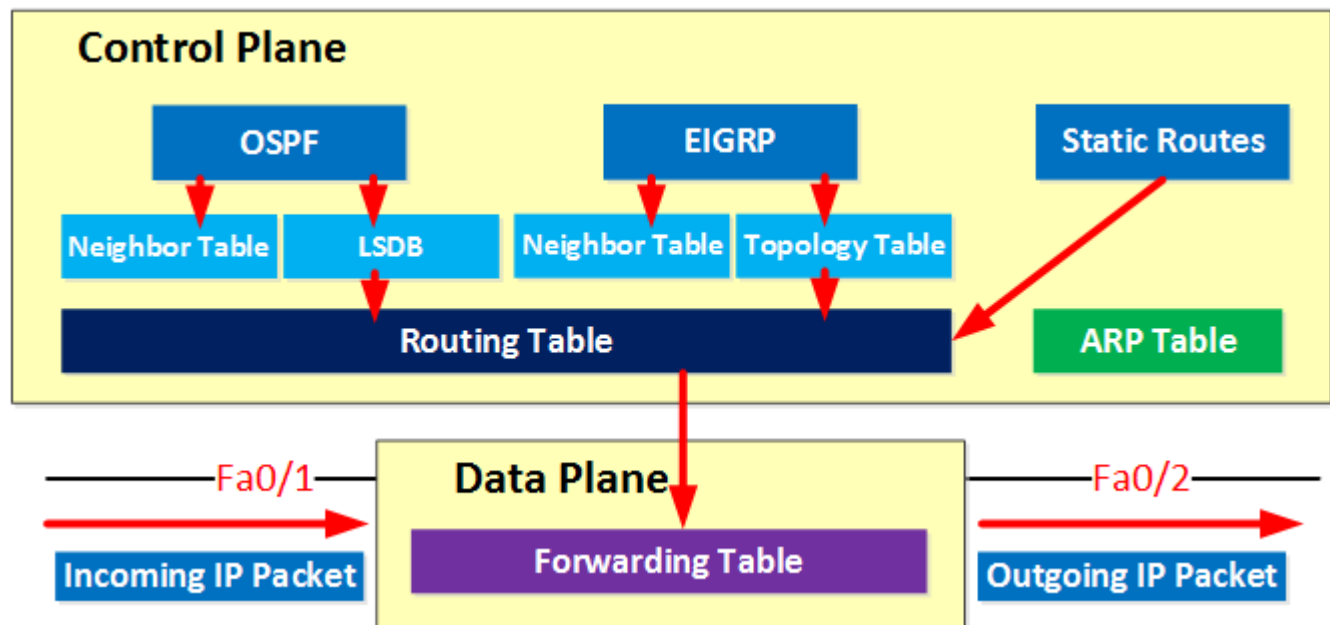


Ilustración 1. Plano de Control vs Plano de Datos

Arriba puede ver el plano de control donde usamos protocolos de enrutamiento como OSPF y EIGRP y algunos enrutamientos estáticos. Las mejores rutas están instaladas en la tabla de enrutamiento. Otra tabla que el enrutador debe construir es la tabla ARP.

La información de la tabla de enrutamiento y ARP luego se usa para construir la tabla de reenvío. Cuando el enrutador recibe un paquete IP, podrá reenviarlo rápidamente ya que la tabla de reenvío ya se ha creado.

Limitaciones de la red tradicional

Todo lo que se describió anteriormente es la forma en que hemos hecho las cosas durante los últimos 30 años, así que no es que haya algo "incorrecto" en las redes tradicionales. Sin embargo, hoy en día existen algunos desafíos comerciales que requieren soluciones diferentes.

La "tendencia" hoy en día es que todo debería ser virtual. No es extraño ver que esto también le está sucediendo al trabajo en red. Las grandes empresas, como Cisco, que solían vender solo hardware propietario, ahora también ofrecen enrutadores virtuales, ASA, controladores LAN inalámbricos, etc. que puede ejecutar en servidores VMWare.

SDN (redes definidas por software) [4]

Al igual que la palabra de moda "nube" hace unos años, cada organización o proveedor tiene una opinión diferente sobre lo que SDN es exactamente y los diferentes productos que ofrecen.

La red tradicional usa un modelo distribuido para el plano de control. Protocolos como ARP, STP, OSPF, EIGRP, BGP y otros se ejecutan por separado en cada dispositivo de red. Estos dispositivos de red se comunican entre sí, pero no existe un dispositivo central que tenga una visión general o que controle toda la red.

Una excepción aquí (para aquellos que están familiarizados con la red inalámbrica) son los Controladores LAN Inalámbricos (WLC). Cuando configura una red inalámbrica, configura todo en el WLC que controla y configura los puntos de acceso. Ya no tenemos que configurar cada punto de acceso por separado, todo lo hace el WLC.

Con SDN, utilizamos un controlador central para el plano de control. Dependiendo de la solución SDN del proveedor, esto podría significar que el controlador SDN toma el control del plano al 100% o que solo tiene una idea en el plano de control de todos los dispositivos de red de la red. El controlador SDN podría ser un dispositivo de hardware físico o una máquina virtual.

Aquí hay una ilustración para ayudarlo a visualizar esto:

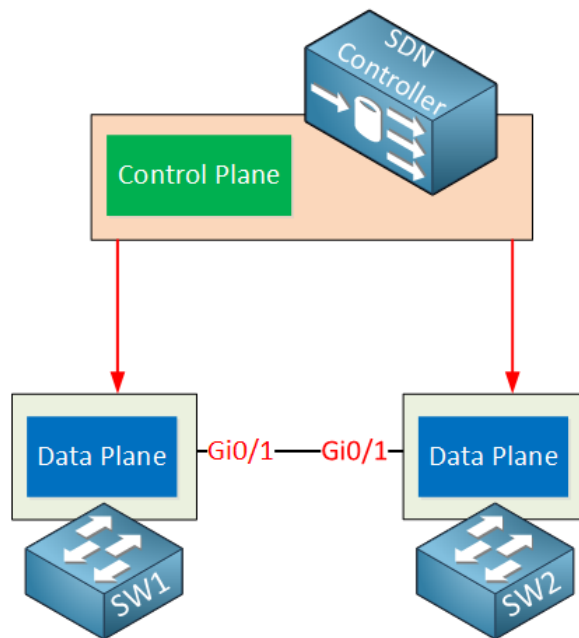


Ilustración 2. Controlador SDN del plano de datos de los Switches.

Arriba puede ver el controlador SDN que es responsable del plano de control. Los conmutadores ahora son dispositivos "tontos" que solo tienen un plano de datos, no un plano de control. El controlador SDN es responsable de alimentar el plano de datos de estos conmutadores con información de su plano de control.

Hay algunas ventajas y desventajas de tener un plano de control distribuido vs uno central. Una de las ventajas de tener un controlador central es que podemos configurar toda la red desde un solo dispositivo. Este controlador tiene acceso completo y una visión de todo lo que está sucediendo en nuestra red.

Agreguemos más detalles a esta historia. El controlador SDN utiliza dos interfaces especiales, de un vistazo a la imagen presentada a continuación:

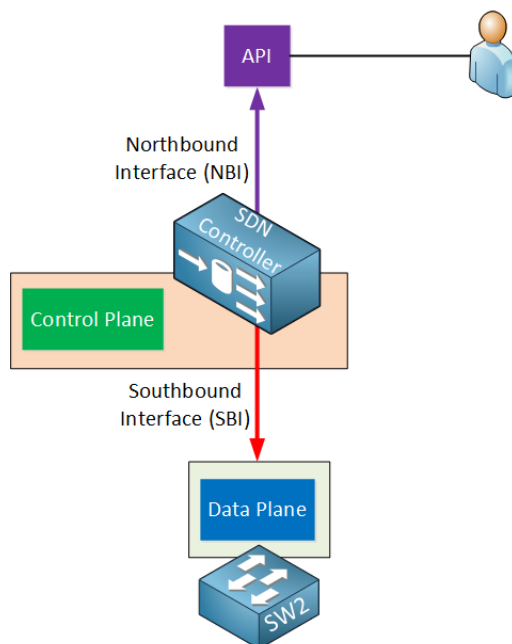


Ilustración 3. Controlador SDN con sus interfaces.

Las interfaces se conocen como la interfaz Northbound (NBI) y la interfaz Southbound (SBI).

Interfaz Southbound

El controlador SDN debe comunicarse con nuestros dispositivos de red para programar el plano de datos. Esto se hace a través de la interfaz hacia el sur. Esta no es una interfaz física sino una interfaz de software, a menudo una API (Interfaz de programación de aplicaciones).

Una API es una interfaz de software que permite que una aplicación brinde acceso a otras aplicaciones mediante el uso de funciones predefinidas y estructuras de datos.

Algunas interfaces populares son:

- OpenFlow: este es probablemente el SBI más popular en este momento, es un protocolo de fuente abierta de Open Networking Foundation. Hay bastantes dispositivos de red y controladores SDN que ya admiten OpenFlow.
- Cisco OpFlex: esta es la respuesta de Cisco a OpenFlow. También es un protocolo de fuente abierta que ha sido enviado al IETF para su estandarización.
- CLI: Cisco ofrece APIC-EM que es una solución SDN para la generación actual de enrutadores y conmutadores. Utiliza protocolos que están disponibles en hardware de generación actual como telnet, SSH y SNMP.

Interfaz Northbound

La interfaz Northbound se utiliza para acceder al controlador SDN. Esto permite que un administrador de red acceda al controlador SDN para configurarlo o para recuperar información de él. Esto podría hacerse a través de una GUI, pero también ofrece una API que permite a otras aplicaciones acceder al controlador SDN. Puede usar esto para escribir scripts y automatizar la administración de su red. Aquí hay unos ejemplos:

- Enumere la información de todos los dispositivos de red en su red.

- Muestra el estado de todas las interfaces físicas en la red.
- Agregue una nueva VLAN en todos sus conmutadores.
- Muestra la topología de toda tu red.
- Configure automáticamente las direcciones IP, el enrutamiento y las listas de acceso cuando se crea una nueva máquina virtual.

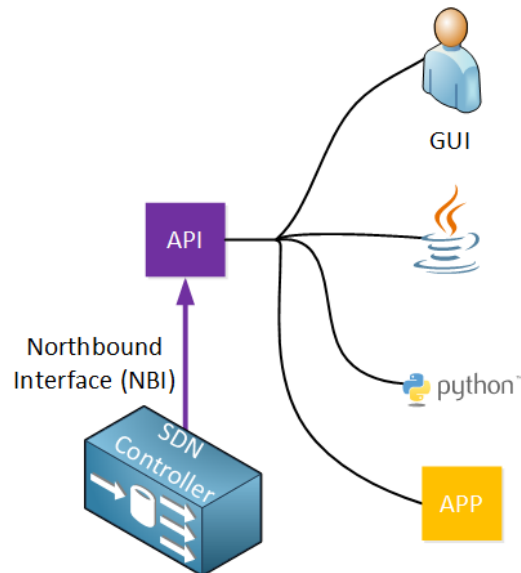


Ilustración 4. Opciones de API de la interfaz Northbound.

A través de la API, múltiples aplicaciones pueden acceder al controlador SDN:

- Un usuario que está usando una GUI para recuperar información sobre la red desde el controlador SDN. Detrás de escena, la GUI está usando la API.
- Los scripts que están escritos en Java o Python pueden usar la API para recuperar información del controlador SDN o configurar la red.
- Otras aplicaciones pueden acceder al controlador SDN. Tal vez una aplicación que configura automáticamente la red una vez que se crea una nueva máquina virtual en un servidor VMware ESXi.

En esta práctica se desarrollarán topologías de forma virtual orientadas a la visualización del protocolo OpenFlow y el comportamiento general en SDN.

La práctica inicia con la familiarización de Mininet y el controlador externo OpenDayLight, luego se trabaja con la unión de ambos que se encuentran en MVs diferentes. Se trabajará una topología desde mininet y se efectuará una configuración en el controlador SDN para gestionar el tráfico. Se trabajará nueva topologías para analizar enrutamiento con el controlador ODL y el Ryu.

4. PROCEDIMIENTO

El entorno de trabajo de esta práctica está compuesto por un equipo de cómputo, y dos máquinas virtuales desplegadas en él. Una de las máquinas contiene un sistema operativo Linux con el controlador Opendaylight instalado. La otra contiene la herramienta de virtualización de red, Mininet.

Las dos máquinas virtuales tienen usuario: *mininet* y contraseña: *infracom*.

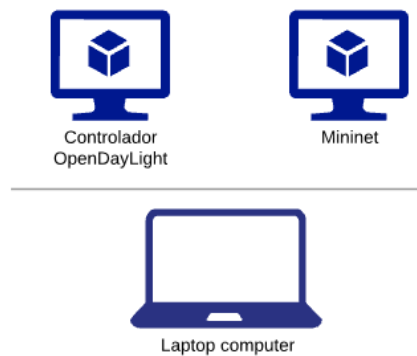


Ilustración 5. Esquema de la práctica.

4.1. Introducción a Mininet y OpenDayLight

Una vez adquiera las MVs que les proveen en el laboratorio ML-340, debe añadirlas a VirtualBox y VMWare del equipo en donde esté trabajando. Debería tener mínimo 2 Gb de memoria RAM. Las máquinas tienen dos interfaces de red, la primera está el adaptador con NAT y la segunda en adaptador sólo anfitrión (esta última permite la conexión entre la topología de red que se ejecute en mininet y el controlador externo OpenDayLight).

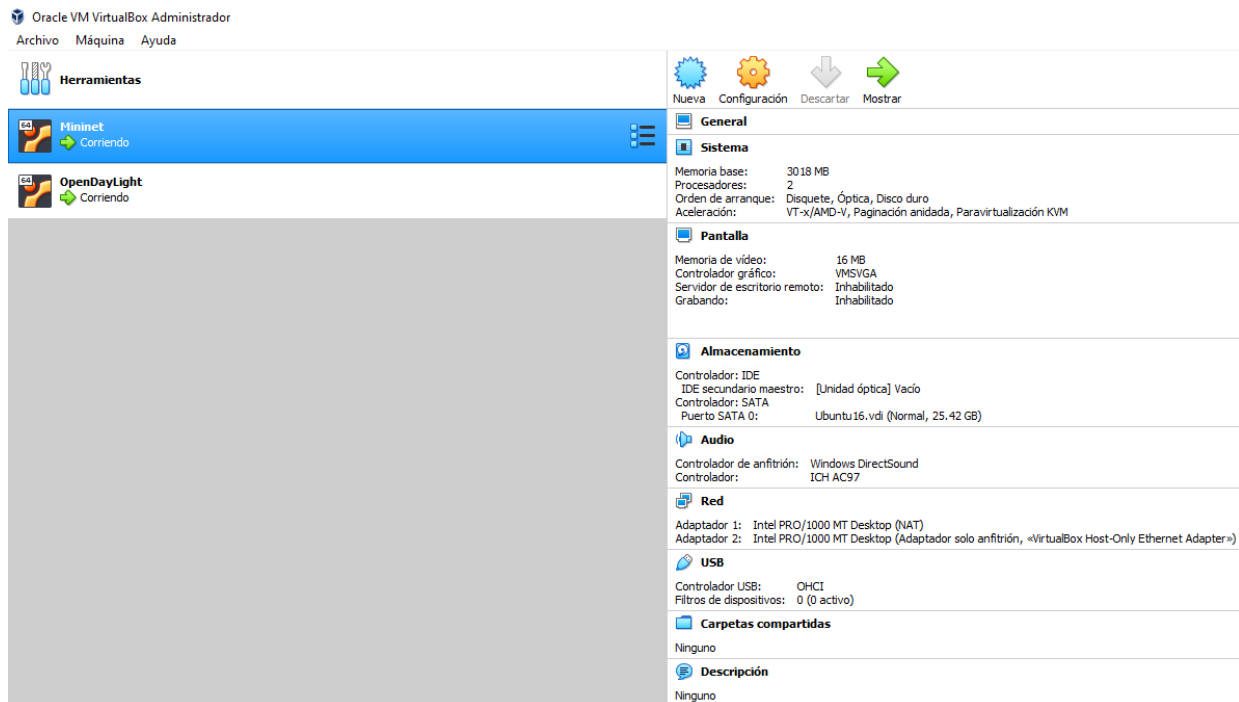


Ilustración 6. Máquinas virtuales en ejecución con los detalles de sus características.

Mininet es un emulador de red que crea redes con hosts virtuales, switches, controladores y enlaces. Los hosts utilizan el software de red de un sistema operativo Linux estándar y los switches soportan el protocolo OpenFlow lo que permite gran flexibilidad y personalización de enrutamientos y SDN.

Mininet soporta multitud de tareas que se benefician al tener un despliegue de red completo en un portátil o PC. Sus principales características son:

- Proporciona un entorno de pruebas sencillo y económico para desarrollar aplicaciones OpenFlow.
- Permite a los desarrolladores trabajar simultáneamente y de forma independiente en la misma topología.
- Soporta pruebas de regresión a nivel de sistema, que son repetibles y empaquetables.
- Permite realizar pruebas de topologías complejas sin necesidad de una red física.
- Incluye un CLI para pruebas de red y debugging.
- Soporta topologías personalizadas e incluye topologías predeterminadas, además de una API Python para la creación de redes y pruebas. [5]

OpenDayLight es un proyecto open-source promovido por la Linux Foundation, una asociación de empresas que desarrollan proyectos en conjunto. Este controlador está implementado en software contenido dentro de su propia máquina virtual de Java, por lo tanto, puede ser desplegado en cualquier plataforma que soporte Java. Al ser open-source contiene APIs abiertas que pueden ser utilizadas por los desarrolladores para aplicaciones. Opendaylight soporta el framework OSGi y REST bidireccional.

ODL es una plataforma en la cual sus módulos reutilizan servicios e interfaces comunes a través de Java. Muchos de estos servicios están contruidos en un modelo proveedor-consumidor, a través de la capa de adaptación de servicio MD-SAL, que reúne las funcionalidades de muchas aplicaciones, las cuales prestan sus servicios, para que desarrolle las tareas del controlador SDN. [5]

4.1.1. Ejecutar topología en mininet

En la ruta que se muestra a continuación, se debe ejecutar el comando **sudo mn**, con este comando se inicia la ejecución de mininet en la MV, y por defecto se crea una topología con un controlador local (c0), un switch (s1), dos hosts (h1 y h2) y dos enlaces entre ellos.

```
mininet@mininet-VirtualBox:~/mininet$ sudo mn
[sudo] password for mininet:
*** 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> █
```

Ilustración 7. Inicio predeterminado de mininet.

Una vez allí se pueden ejecutar diferentes comandos para ver el estado de la red. Para salir debe ejecutarse el comando **quit**. Y para limpiar lo ejecutado en mininet en la sesión anterior se debe usar el comando **mn -c**.


```

mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=8061>
<Host h2: h2-eth0:10.0.0.2 pid=8063>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=8068>
<Controller c0: 127.0.0.1:6653 pid=8054>
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>

```

Ilustración 8. Algunos comandos de mininet.

Se puede especificar un comando para uno de los nodos de la topología, por ejemplo, para ver la configuración de red de los hosts se ejecuta:

```

mininet> h2 ifconfig
h2-eth0  Link encap:Ethernet direcciónHW 4e:ce:f9:af:1e:18
        Direc. inet:10.0.0.2  Difus.:10.255.255.255  Másc:255.0.0.0
        Dirección inet6: fe80::4cce:f9ff:feaf:1e18/64 Alcance:Enlace
        ACTIVO DIFUSION FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
        Paquetes RX:62 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:22 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatx:1000
        Bytes RX:6027 (6.0 KB)  TX bytes:1636 (1.6 KB)

lo       Link encap:Bucle local
        Direc. inet:127.0.0.1  Másc:255.0.0.0
        Dirección inet6: ::1/128 Alcance:Anfitrión
        ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
        Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatx:1000
        Bytes RX:0 (0.0 B)  TX bytes:0 (0.0 B)

mininet> h1 ifconfig
h1-eth0  Link encap:Ethernet direcciónHW 5a:23:25:f1:72:cf
        Direc. inet:10.0.0.1  Difus.:10.255.255.255  Másc:255.0.0.0
        Dirección inet6: fe80::5823:25ff:fe1:72cf/64 Alcance:Enlace
        ACTIVO DIFUSION FUNCIONANDO MULTICAST  MTU:1500  Métrica:1
        Paquetes RX:62 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:22 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatx:1000
        Bytes RX:6027 (6.0 KB)  TX bytes:1636 (1.6 KB)

lo       Link encap:Bucle local
        Direc. inet:127.0.0.1  Másc:255.0.0.0
        Dirección inet6: ::1/128 Alcance:Anfitrión
        ACTIVO BUCLE FUNCIONANDO  MTU:65536  Métrica:1
        Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0
        Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0
        colisiones:0 long.colatx:1000
        Bytes RX:0 (0.0 B)  TX bytes:0 (0.0 B)

```

Ilustración 9. Ver configuración de red.

Para realizar un ping entre un host hacia otro se ejecuta un comando como se muestra a continuación:

```

mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.11 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=1.70 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.092 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.119 ms
^C
--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3044ms
rtt min/avg/max/mdev = 0.092/0.758/1.707/0.685 ms
mininet> h2 ping h1
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=1.81 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.195 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.056 ms
^C
--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2013ms
rtt min/avg/max/mdev = 0.056/0.688/1.814/0.798 ms

```

Ilustración 10. Ping entre los hosts de la red.

4.1.2. Ejecutar controlador OpenDayLight

Para ejecutar este controlador externo a mininet deben ubicarse en la máquina virtual del controlador, abrir una ventana del terminal, seguir la siguiente ruta */distribution-karaf-0.4.0-Beryllium* y ejecutar el comando **./bin/karaf**

Al terminar la inicialización, debe visualizarse como se muestra a continuación:

```
mininet@mininet-VirtualBox:~$  
mininet@mininet-VirtualBox:~$ cd distribution-karaf-0.4.0-Beryllium  
mininet@mininet-VirtualBox:~/distribution-karaf-0.4.0-Beryllium$  
mininet@mininet-VirtualBox:~/distribution-karaf-0.4.0-Beryllium$ ./bin/karaf  
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512m; support was removed in 8.0
```



The Karaf logo is a stylized representation of the word "karaf" in a monospace font, where each letter is constructed from yellow geometric shapes like triangles and rectangles.

```
Hit '<tab>' for a list of available commands  
and '[cmd] --help' for help on a specific command.  
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
```

```
opendaylight-user@root>
```

Ilustración 11. Ejecución de controlador OpenDayLight.

Finalizada la ejecución del comando se puede acceder a la dirección IP que tenga la interfaz enp0s8 desde el navegador de nuestra máquina anfitriona de las MVs. Por ejemplo, <http://192.168.56.102:8181/index.html> , en ella se muestra la página de login del controlador. Las credenciales de acceso a la interfaz web son admin, admin.

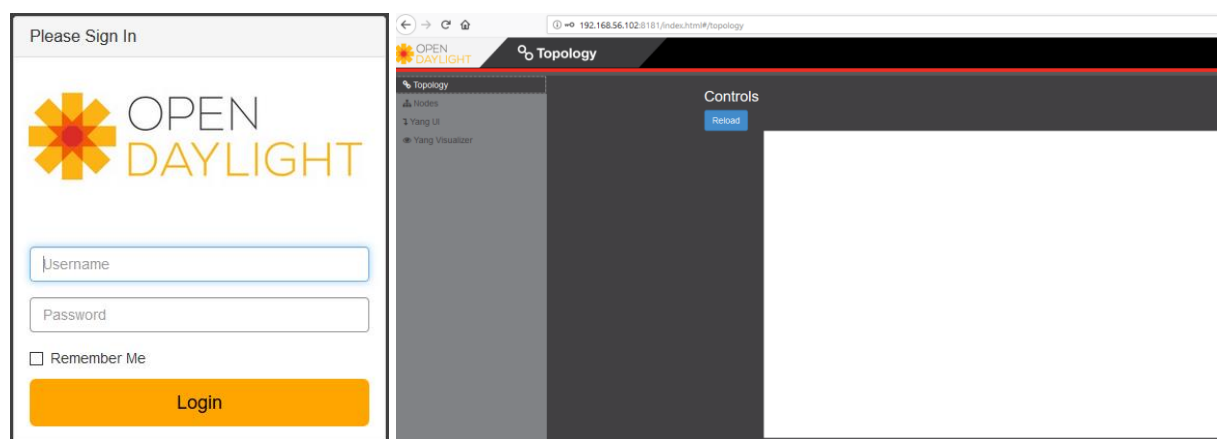


Ilustración 12. Login e Interfaz Web OpenDayLight.

Como no se ha configurado una topología desde mininet que esté apuntando a la dirección IP del controlador remoto, no se visualizará nada en la interfaz web. Ahora con el comando de la imagen se está generando una topología con este controlador externo.

```

mininet@mininet-VirtualBox:~/mininet$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.56.102,port=6633 --switch ovs,protocols=OpenFlow13
*** 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>

```

Ilustración 13. Comando para crear topología que trabaje con controlador externo.

Luego de dar clic en Reload de la interfaz gráfica del controlador se muestran los switches creados en la topología.

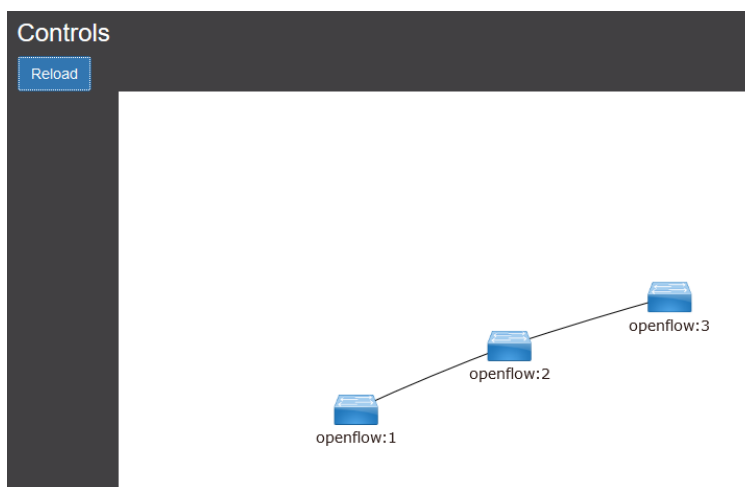


Ilustración 14. Topología incompleta en ODL.

Después de ejecutar pingall en mininet y dar clic en Reload de la interfaz gráfica del controlador se obtiene la topología de red completa con las direcciones MAC de los hosts:

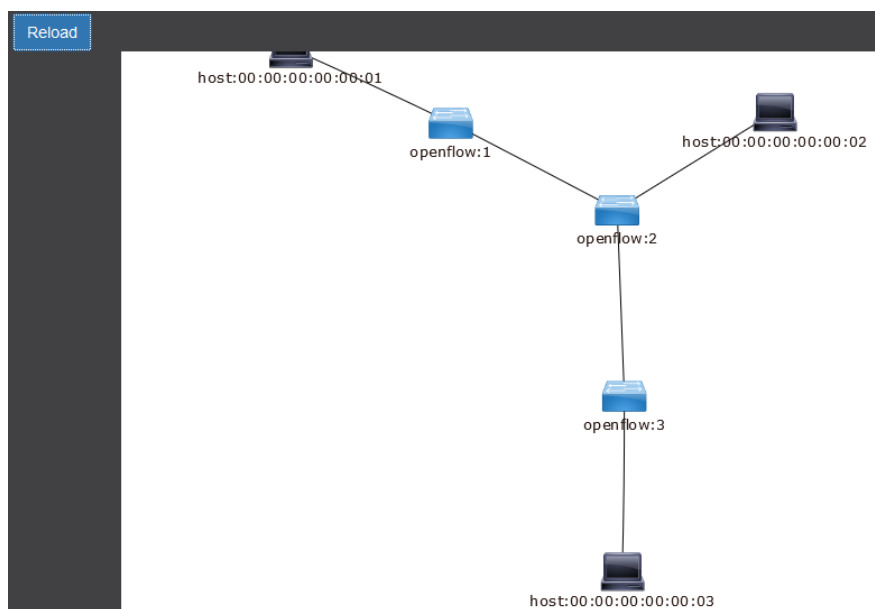
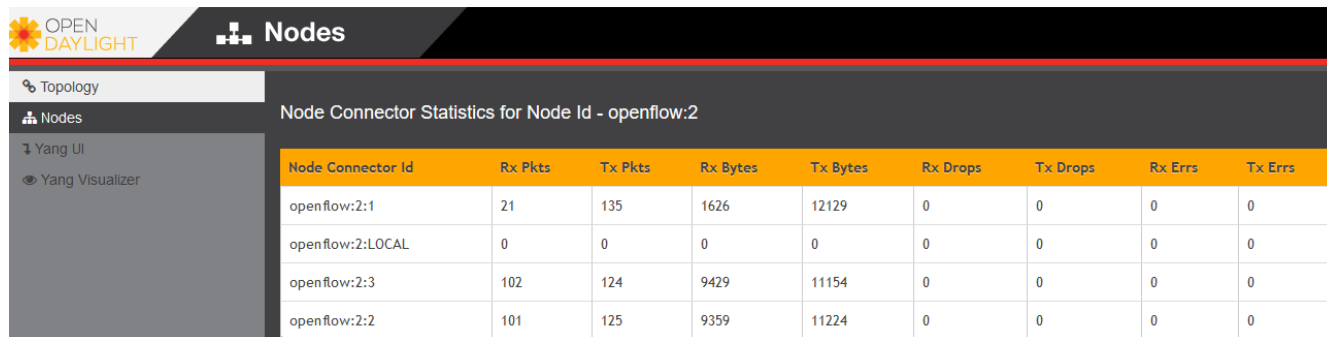


Ilustración 15. Topología completa en ODL.

En la pestaña Nodes, encuentran estadísticas importantes del flujo de paquetes que ocurre en la red.



Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Rx Drops	Tx Drops	Rx Errs	Tx Errs
openflow:2:1	21	135	1626	12129	0	0	0	0
openflow:2:LOCAL	0	0	0	0	0	0	0	0
openflow:2:3	102	124	9429	11154	0	0	0	0
openflow:2:2	101	125	9359	11224	0	0	0	0

Ilustración 16. Estadísticas de tráfico en ODL.

En la pestaña YANG, se puede ver que es un lenguaje para describir la estructura básica de algunos datos de aplicaciones que se almacenan en una jerarquía de árbol dentro de los contenedores. Yang puede ser utilizado, para definir el formato, de eventos o notificaciones emitidas por los elementos de red, y permite modelar datos para definir la firma de procedimientos remotos, que pueden ser invocados en los elementos de red a través del protocolo NETCONF. YANG es un lenguaje modular que representa la estructura de datos en un árbol XML [6].



Ilustración 17. Pestaña Yang en ODL.

La interfaz de usuario OpenDaylight Yang es un cliente REST gráfico para crear y enviar solicitudes REST a la base de datos OpenDaylight. Podemos usar la interfaz de usuario de Yang para obtener información de la base de datos, o para construir comandos REST para modificar la información en el base de datos, cambiando las configuraciones de red.

Cuando se da clic en el botón Expand all para ver todas las API disponibles. No todos funcionarán porque no se instalaron todas las funciones. Una API que funcionará es la API de inventario. Haga clic en él, luego navegue hacia abajo hasta el atributo de nodos y haga clic en el botón Enviar para enviar el método GET API al controlador.

4.2. Topología con controlador externo ODL, elaboración con Miniedit y configuración del controlador para bloquear flujo de un host a otro.

En esta parte de la práctica, se tendrá una topología de red creada a partir de Miniedit, exportando el archivo y editando el mismo. Dicha topología se ejecutará y conectará con el controlador externo ODL. Se configurará un parámetro en el controlador para que no permita flujo de un host a otro. Por último, se capturará y analizará el tráfico de Wireshark.

4.2.1. Creación de topología en miniedit

1. En el terminal vaya al directorio /mininet/examples/. Desde allí ejecute el comando:
\$ python ./miniedit.py
2. Crear la topología de la siguiente imagen. Para ubicar los elementos de red sobre la ventana hay que dar clic sobre uno de ellos y luego dar clic sobre el espacio en blanco.

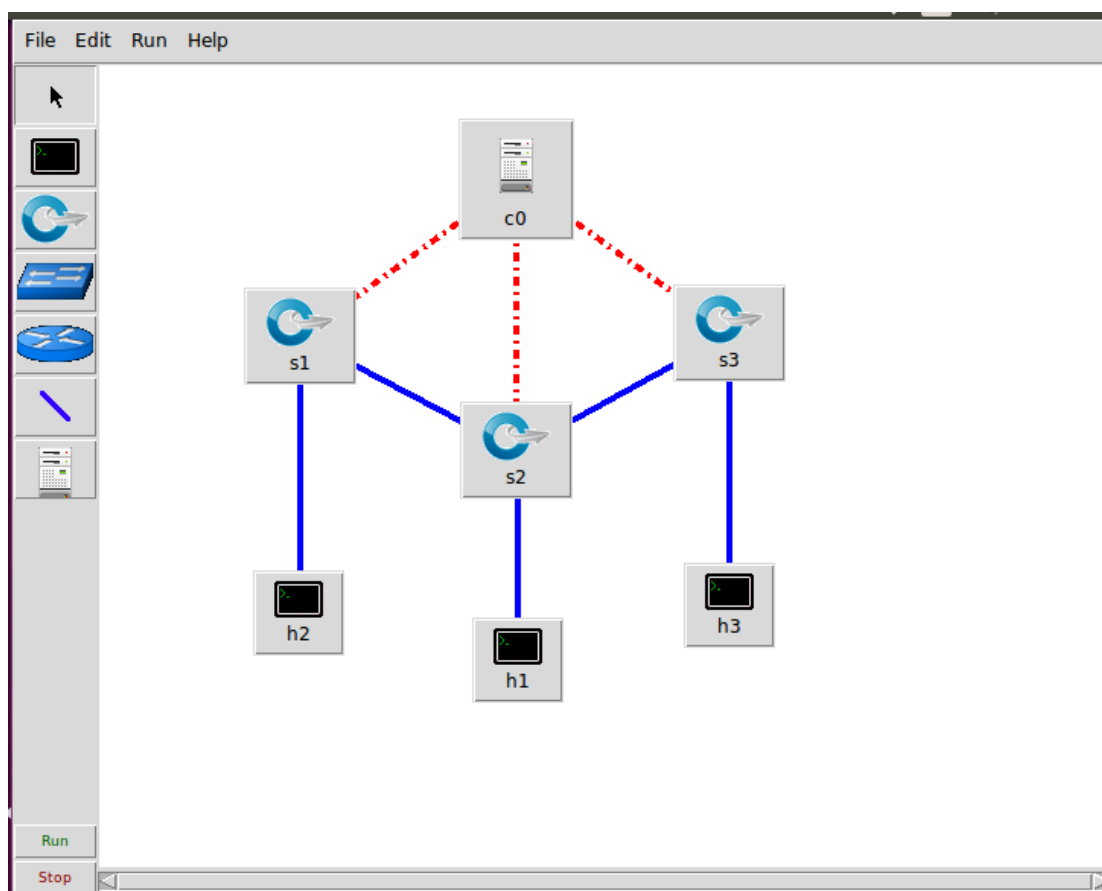
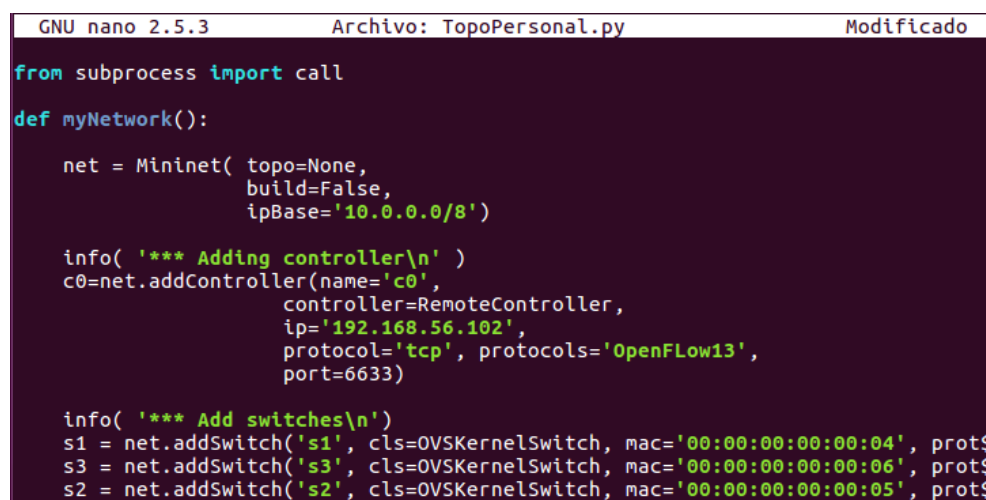


Ilustración 18. Topología sección 4.2.1.

3. Una vez la topología se encuentre de la misma manera, se debe guardar la misma. Debe dar clic en la opción "File", luego clic en "Export Level 2 script". Y guarde el archivo con un nombre que recuerde, se genera un archivo .py. Para este ejemplo, lo llamé "TopoPersonal.py".

- Ubique el archivo generado y modifique el mismo con el editor de su preferencia, para que guarde concordancia con los siguientes datos:

ipBase=	10.0.0.0/8
port=	6633
controller =	RemoteController
ip=	192.168.56.X
(switches) mac=	s1: 00:00:00:00:00:04 s2: 00:00:00:00:00:05 s3: 00:00:00:00:00:06
(hosts) ip y mac	h1: 10.0.0.1 00:00:00:00:00:01 h2: 10.0.0.2 00:00:00:00:00:02 h3: 10.0.0.3 00:00:00:00:00:03



```

GNU nano 2.5.3 Archivo: TopoPersonal.py Modificado
from subprocess import call
def myNetwork():
    net = Mininet( topo=None,
                  build=False,
                  ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                        controller=RemoteController,
                        ip='192.168.56.102',
                        protocol='tcp', protocols='OpenFlow13',
                        port=6633)

    info( '*** Add switches\n' )
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch, mac='00:00:00:00:00:04', prot$
    s3 = net.addSwitch('s3', cls=OVSKernelSwitch, mac='00:00:00:00:00:06', prot$
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch, mac='00:00:00:00:00:05', prot$

```

Ilustración 19. Pestaña Yang en ODL.

- Luego de guardar el archivo .py, ejecute el mismo con **sudo python TopoPersonal.py**.
- Ejecute en otra ventana del terminal **sudo wireshark**. Guarde una captura de tráfico. Verifique conectividad en la topología. Analice los resultados.

4.2.2. Implementación de flujo particular en el controlador

El controlador envía flujos a los switches para configurar directrices de funcionamiento. En esta sección se buscará generar un flujo que descarte los paquetes que tenga como MAC origen la del host 1 y MAC destino la del host 2. El flujo deberá instalarse en el switch 1. [5]

- Reemplace los espacios en blanco según las direcciones MAC de los dispositivos involucrados en la siguiente plantilla para completar el flujo particular deseado.

```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="urn:opendaylight:flow:inventory">
  <instructions>
    <instruction>
      <order>0</order>
      <apply-actions>
        <action>
          <order>0</order>

```

```

        <drop-action/>
      </action>
    </apply-actions>
  </instruction>
</instructions>
  <table_id>0</table_id>
  <id>1</id>
  <cookie_mask>255</cookie_mask>
  <installHw>>false</installHw>
  <match>
    <ethernet-match>
      <ethernet-destination>>
        <address>__:__:__:__:__</address>
      </ethernet-destination>
      <ethernet-source>
        <address>__:__:__:__:__</address>
      </ethernet-source>
    </ethernet-match>
  </match>
  <hard-timeout>120</hard-timeout>
  <cookie>3</cookie>
  <idle-timeout>34</idle-timeout>
  <flow-name>LabSdn</flow-name>
  <priority>15</priority>
  <barrier>>false</barrier>
</flow>

```

Guarde el fichero generado con extensión .xml en el **escritorio**. Por ejemplo, flujop.xml.

2. Cree una nueva ventana de terminal en la máquina virtual del controlador y proceda con la instalación del flujo en el switch 1 de la topología creada anteriormente. Para instalar el flujo deberá ejecutar el siguiente comando:

```

sudo curl --user admin:admin -i -X PUT -H "Content-Type:application/xml;
charset=utf-8" -d @"/home/ubuntu/Escritorio/flujop.xml"
http://192.168.56.102:8181/restconf/config/.opendaylight-
inventory:nodes/node/openflow:1/table/0/flow/1

```

3. Compruebe la instalación del flujo en la interaz web de ODL.
4. Ejecute Wireshark con sudo wireshark y guarde captura de tráfico.
5. ¿Existe conectividad entre h1 y h2? Explique la razón de lo ocurrido, sustentando con las capturas de tráfico.

4.3. Implementación de nueva topología de red (ODL)

Ahora debe implementar la topología de la siguiente figura en miniedit o a través de una línea de código.

1. Deben seguir el esquema presentado, las direcciones MAC no deben ser iguales a las que se muestran en la imagen.
2. Debe conectarse al controlador externo OpenDayLight.
3. Inicie una captura de tráfico con Wireshark. No realice ningún comando, ¿qué paquetes se observan en la captura?, ¿por qué se visualizan estos paquetes?

4. Describa una aplicación de enrutamiento dinámico que considera puede utilizarse en esta topología.
5. Implemente en un controlador el enrutamiento que describió en el punto anterior. **(Opcional)**
6. Realice pruebas de conectividad y realice una captura de tráfico. Analice los paquetes capturados. ¿Puedo tener varios controladores en una topología como esta?

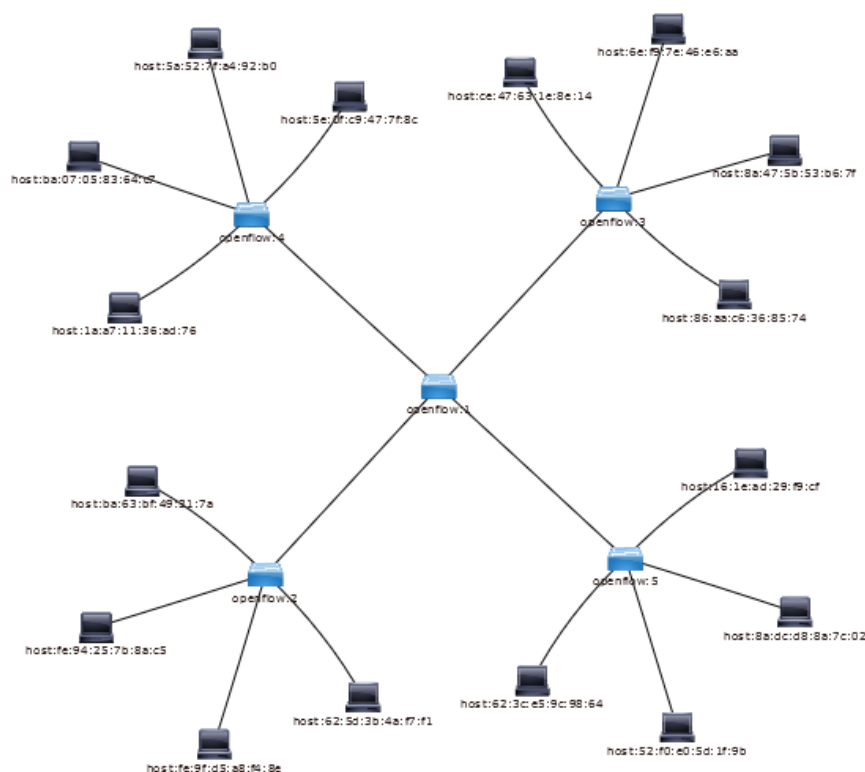


Ilustración 20. Topología sección 4.3.

4.4. Aplicación Router configurada por interfaz REST (Con Ryu – controlador externo) [7]

En esta parte de la práctica se busca hacer actuar a los switches como si fueran routers. Se ejecutará la aplicación `router_rest` de Ryu que proporciona la funcionalidad de routing. Aquí se podrá configurar las tablas de routing en cada router y además se pueden dar direcciones IP a sus interfaces. Todo lo anterior se realiza a través de la interfaz REST que proporciona la aplicación `router_rest`. El script con la topología lo encuentra en un archivo de Python dentro del directorio `home` del usuario Ubuntu(**home/Ubuntu**) en su MV Ubuntu14 (sólo usar esta). Debería tener abiertas tres ventanas diferentes de terminal para esta práctica.

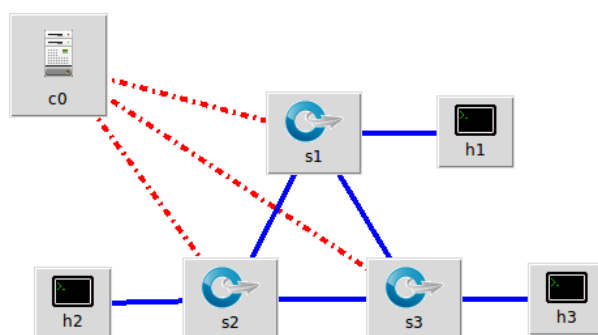


Ilustración 21. Topología sección 4.4

1. Ejecutar Ryu usando la aplicación de router, así:
\$ ryu-manager ryu/ryu/app/rest_router.py
2. Abrir otra terminal, dirigirse a la carpeta mininet y ejecutar la topología correspondiente a esta parte del laboratorio.
\$ sudo mn --controller remote --custom mi-topo-3.py --topo mitopo3
3. Para habilitar la comunicación entre los elementos de la red se deben usar los siguientes comandos REST:

```
curl -X POST -d '{"address":"10.0.1.10/24"}' http://localhost:8080/router/0000000000000001
curl -X POST -d '{"address":"10.100.0.1/30"}' http://localhost:8080/router/0000000000000001
curl -X POST -d '{"gateway":"10.100.0.2"}' http://localhost:8080/router/0000000000000001
curl -X POST -d '{"address":"10.0.2.10/24"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"address":"10.100.0.2/30"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"address":"10.200.0.2/30"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"destination":"10.0.1.0/24","gateway":"10.100.0.1"}'
http://localhost:8080/router/0000000000000002
curl -X POST -d '{"gateway":"10.200.0.1"}' http://localhost:8080/router/0000000000000002
curl -X POST -d '{"address":"10.0.3.10/24"}' http://localhost:8080/router/0000000000000003
curl -X POST -d '{"address":"10.200.0.1/30"}' http://localhost:8080/router/0000000000000003
curl -X POST -d '{"gateway":"10.200.0.2"}' http://localhost:8080/router/0000000000000003
```
4. Para ver la configuración de un router se puede usar:

```
curl -X GET http://localhost:8080/router/0000000000000001
```
5. Inicie una captura de Wireshark, seguidamente ejecute en el Shell de mininet el comando pingall. Detenga la captura de tráfico y analice qué tipo de paquetes OpenFlow están en la captura. Liste los mensajes del controlador al switch, los mensajes asíncronos y los mensajes simétricos, describa la funcionalidad de cada uno.

4.5. Otros controladores

1. Consulte otros controladores disponibles externos a mininet y que no se hayan trabajado en la guía, describa tres de ellos e implemente una topología en uno de los tres controladores.
2. Verifique conectividad entre todos los hosts de su topología y guarde una captura en Wireshark.
3. Analice si existen diferencias entre el tráfico capturado haciendo uso de los controladores propuestos en la guía de laboratorio y el que usted implemento.

4.6. Otras preguntas

1. ¿Cuál es la diferencia entre SDN y Openflow?
2. ¿Qué es NFV y qué tiene que ver con SDN?
3. ¿Qué encuentra favorable en trabajar con SDN en lugar de las redes tradicionales?
4. ¿Qué sería complejo a la hora de implementar SDN en los proveedores de internet?

6. ENTREGABLES

- Informe digital de análisis que contenga el proceso de solución de los requerimientos efectuados en cada una de las secciones de la práctica, además deben resolver las preguntas solicitadas. Si

no se evidencian los procesos de la solución en el informe, se asumirá que no fueron desarrollados.

- Archivos de las capturas solicitadas en Wireshark.

Nota: Deben hacer una sola entrega, en una sola carpeta comprimida con todos archivos.

7. REFERENCIAS

- [1]<http://search.ebscohost.com.ezproxy.uniandes.edu.co:8080/login.aspx?direct=true&db=nlebk&AN=656220&lang=es&site=ehost-live>
- [2]<http://search.ebscohost.com.ezproxy.uniandes.edu.co:8080/login.aspx?direct=true&db=nlebk&AN=1511855&lang=es&site=ehost-live>
- [3]<https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html>
- [4]<https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-sdn-software-defined-networking/>
- [5] Frías, José. Diseño y despliegue de una red definida por software sobre máquinas virtuales. Universidad Carlos III de Madrid. 2016. 97p.
- [6] Rodríguez Farfán, José. Integración de redes IP utilizando SDN. Instituto Tecnológico de Buenos Aires. 2017. 57p.
- [7] Lamallam, Randa. Propuesta de entorno y prácticas de laboratorio para tecnologías SDN. Universidad Politécnica de Madrid. 2017. 116p.
- [8] Serrano Carrera, David. Redes Definidas por Software (SDN): OpenFlow. Universidad Politécnica de Valencia. 2015. 55p.
- [9] <https://github.com/mininet/openflow-tutorial/wiki/Router-Exercise>
- [10] http://cse.iitkgp.ac.in/~sandipc/Courses/CS60008/Assignment_2.pdf
- [11] <https://intronetworks.cs.luc.edu/current/html/mininet.html#>
- [12] <http://www.cs.huji.ac.il/~yotamhc/courses/workshop/2015/ex2/>
- [13] <http://routing.cloud.mobilesdn.org/>

HISTORIAL DE REVISIONES

FECHA	AUTOR	OBSERVACIONES
01/10/2019	Jonatan Legro Pastrana j.legro@uniandes.edu.co	Versión inicial del documento.