



UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Grupo 08

Laboratorio 3

INF256 - 2021-1 - 201

Redes de Computadores

4 de agosto de 2021

Álvaro Ortiz Hermosilla alvaro.ortizh@sansano.usm.cl	201810523-6
---	-------------

José Sansana Parra jose.sansana@sansano.usm.cl	201773535-K
---	-------------

Índice

1. Red 1: Anillo Simple	3
1.1. Caso 1	3
1.2. Caso 2	4
1.3. Caso 3	6
2. Red 2: Dos Caminos	6

Índice de figuras

1. Red de anillo simple.	3
2. Wireshark con h1 ping -c 1 h8 con todos los links.	4
3. pingall caso 1	4
4. Wireshark con h1 ping -c 1 h8 con un link faltante.	5
5. pingall caso 2	5
6. Ping antihorario	6
7. Topología de la red 2.	6

1. Red 1: Anillo Simple

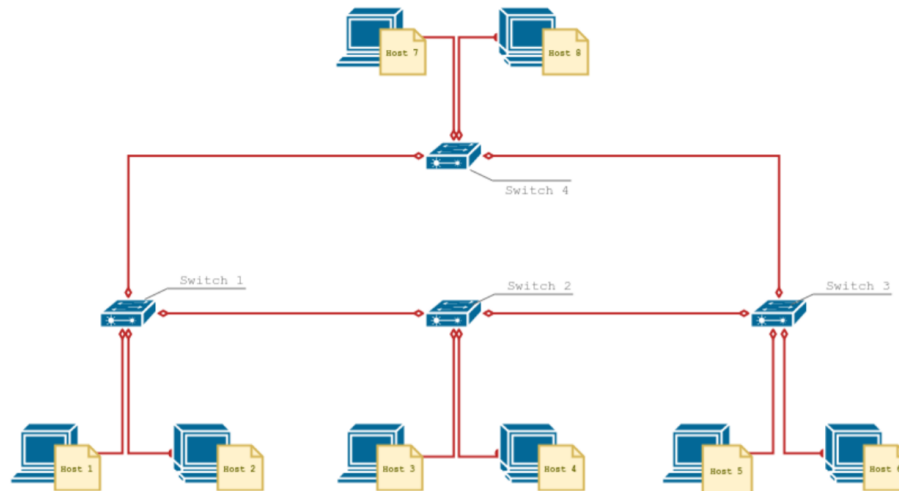


Figura 1: Red de anillo simple.

Esta topología es ejecutada en el terminal utilizando el comando:

```
sudo mn -custom topologia1.py -topo red -controller remote -switch
ovsk -mac
```

Aunque la topología cortando un link se realizó mediante:

```
sudo mn -custom topologia1_2.py -topo red -controller remote -switch
ovsk -mac
```

1.1. Caso 1

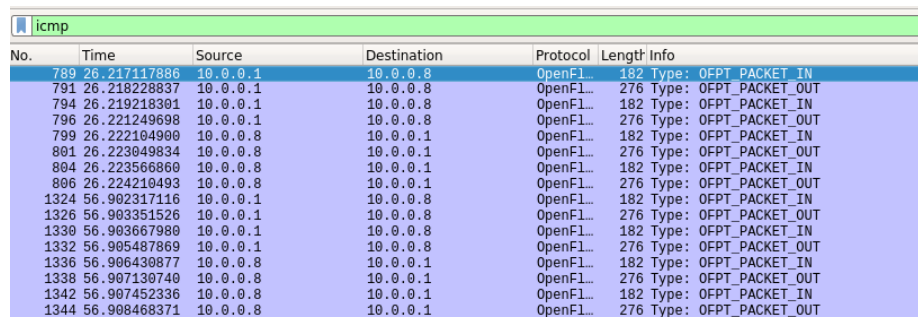
El primer caso consistió en configurar un controlador para que la topología de anillo funcionara correctamente, para ello fueron utilizados los componentes por defecto de POX "l2_learning" y también "spanning_tree" el cual permite eliminar bucles en topologías de anillo. Para su ejecución fue utilizado el comando en terminal (dentro de la carpeta pox):

```
python3 pox.py -verbose openflow.spanning_tree -no-flood -hold-down
openflow.discovery forwarding.l2_learning
```

Una vez ejecutado dicho comando, se utilizó la herramienta Wireshark para ver que sucedía con el ejemplo de comunicación entre host1 (h1) y host8 (h8):

```
h1 ping -c 1 h8
```

Esta tabla es relativamente corta ya que desde el host 1 había un link directo que co-

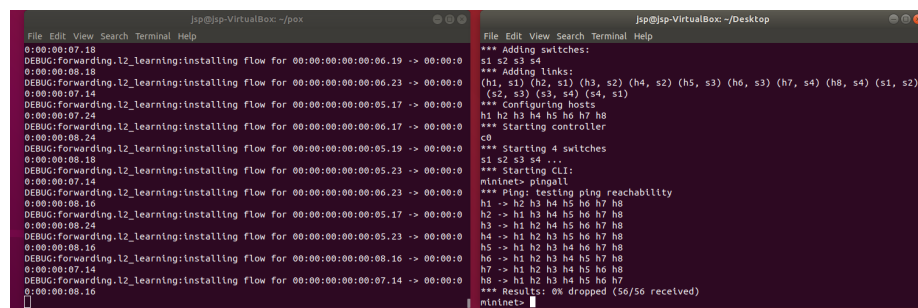


No.	Time	Source	Destination	Protocol	Length	Info
789	26.217117886	10.0.0.1	10.0.0.8	OpenFL...	182	Type: OFPT_PACKET_IN
791	26.218228837	10.0.0.1	10.0.0.8	OpenFL...	276	Type: OFPT_PACKET_OUT
794	26.219218301	10.0.0.1	10.0.0.8	OpenFL...	182	Type: OFPT_PACKET_IN
796	26.221240908	10.0.0.1	10.0.0.8	OpenFL...	276	Type: OFPT_PACKET_OUT
799	26.223049834	10.0.0.8	10.0.0.1	OpenFL...	182	Type: OFPT_PACKET_IN
801	26.223049834	10.0.0.8	10.0.0.1	OpenFL...	276	Type: OFPT_PACKET_OUT
804	26.223566860	10.0.0.8	10.0.0.1	OpenFL...	182	Type: OFPT_PACKET_IN
806	26.224210493	10.0.0.8	10.0.0.1	OpenFL...	276	Type: OFPT_PACKET_OUT
1324	56.902317116	10.0.0.1	10.0.0.8	OpenFL...	182	Type: OFPT_PACKET_IN
1326	56.903351526	10.0.0.1	10.0.0.8	OpenFL...	276	Type: OFPT_PACKET_OUT
1330	56.903667980	10.0.0.1	10.0.0.8	OpenFL...	182	Type: OFPT_PACKET_IN
1332	56.905487869	10.0.0.1	10.0.0.8	OpenFL...	276	Type: OFPT_PACKET_OUT
1336	56.906430877	10.0.0.8	10.0.0.1	OpenFL...	182	Type: OFPT_PACKET_IN
1338	56.907130740	10.0.0.8	10.0.0.1	OpenFL...	276	Type: OFPT_PACKET_OUT
1342	56.907452336	10.0.0.8	10.0.0.1	OpenFL...	182	Type: OFPT_PACKET_IN
1344	56.908468371	10.0.0.8	10.0.0.1	OpenFL...	276	Type: OFPT_PACKET_OUT

Figura 2: Wireshark con h1 ping -c 1 h8 con todos los links.

nectaba con el switch 4, y por ende con el host 8.

Ejemplo de ping exitoso entre todos los hosts para caso 1:



```

jdp@jdp-VirtualBox: ~/box
0:00:00:07.18 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:06.19 -> 00:00:00:00:00:00:00:00
0:00:00:08.18 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:06.23 -> 00:00:00:00:00:00:00:00
0:00:00:07.14 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:05.17 -> 00:00:00:00:00:00:00:00
0:00:00:07.24 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:06.17 -> 00:00:00:00:00:00:00:00
0:00:00:08.24 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:05.19 -> 00:00:00:00:00:00:00:00
0:00:00:08.18 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:05.23 -> 00:00:00:00:00:00:00:00
0:00:00:07.14 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:06.23 -> 00:00:00:00:00:00:00:00
0:00:00:08.16 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:05.17 -> 00:00:00:00:00:00:00:00
0:00:00:08.16 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:05.23 -> 00:00:00:00:00:00:00:00
0:00:00:07.14 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:06.16 -> 00:00:00:00:00:00:00:00
0:00:00:07.14 DEBUG:forwarding_l2_learning:installing flow for 00:00:00:00:00:07.14 -> 00:00:00:00:00:00:00:00
0:00:00:08.16 mininet>

jdp@jdp-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (s1, s2)
(s2, s3) (s3, s4) (s4, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>
  
```

Figura 3: pingall caso 1

1.2. Caso 2

En este caso se eliminó un link entre el switch 1 y el switch 4 y se utilizó la misma configuración de controlador que en el anterior caso.

Utilizando Wireshark podemos notar que en relación al caso anterior, la tabla es mas grande utilizando el ping:

h1 ping h8

La intuición insinúa que esto es debido a que se cortó una ruta directa desde el switch 1 al 4, y por tanto, el controlador debe redirigir el paquete por una ruta mas larga a través del switch 2, 3 y finalmente el 4 para poder llegar al host requerido

Ejemplo de ping exitoso entre todos los hosts para caso 1:

Time	Source	Destination	Protocol	Length	Info
187	7.762232788	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
189	7.763131790	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
192	7.763579448	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
194	7.764489434	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
197	7.764984261	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
199	7.765913320	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
202	7.766403231	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
204	7.767275687	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
207	7.767775756	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN
209	7.768692557	10.0.0.8	10.0.0.1	OpenFl...	276 Type: OFPT_PACKET_OUT
212	7.769345526	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN
214	7.770252463	10.0.0.8	10.0.0.1	OpenFl...	276 Type: OFPT_PACKET_OUT
217	7.770844231	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN
219	7.771743799	10.0.0.8	10.0.0.1	OpenFl...	276 Type: OFPT_PACKET_OUT
222	7.772211077	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN
224	7.773080828	10.0.0.8	10.0.0.1	OpenFl...	276 Type: OFPT_PACKET_OUT
722	38.420572657	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
723	38.421749834	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
726	38.422178848	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
727	38.422981488	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
730	38.423418087	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
731	38.424201879	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
734	38.424571931	10.0.0.1	10.0.0.8	OpenFl...	182 Type: OFPT_PACKET_IN
735	38.425459895	10.0.0.1	10.0.0.8	OpenFl...	276 Type: OFPT_PACKET_OUT
738	38.425856168	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN
740	38.426690178	10.0.0.8	10.0.0.1	OpenFl...	276 Type: OFPT_PACKET_OUT
743	38.427055462	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN
745	38.428159355	10.0.0.8	10.0.0.1	OpenFl...	276 Type: OFPT_PACKET_OUT
748	38.428603373	10.0.0.8	10.0.0.1	OpenFl...	182 Type: OFPT_PACKET_IN

Figura 4: Wireshark con h1 ping -c 1 h8 con un link faltante.

```

jisp@jisp-VirtualBox: ~/box
File Edit View Search Terminal Help
0:00:00:07:21
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:03:22 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:04:22 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:03:20 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:04:22 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:03:22 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:00:16 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:07:14 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:08:16 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:07:21 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:08:21 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:06:12 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:06:22 -> 00:00:00:00:00:00:00:16
DEBUG: forwarding_l2_learning: installing flow for 00:00:00:00:00:06:22 -> 00:00:00:00:00:00:00:16
jisp@jisp-VirtualBox: ~/box

jisp@jisp-VirtualBox: ~/Desktop
File Edit View Search Terminal Help
jisp@jisp-VirtualBox:~/Desktop$ sudo mn --custom topologia1.2.py --topo red --controller remote
--switch ovsk --mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (s2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (s1, s2) (s2, s3) (s3, s4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
mininet>

```

Figura 5: pingall caso 2

1.3. Caso 3

Finalmente se nos pidió configurar un controlador que transmitiera los paquetes por el anillo de switches de forma antihoraria (debido a que, al ser grupo 08, el numero 8 es par, y por ende, dictado como antihorario), para ello fue modificado el componente “l2_learning” agregando al código el redireccionamiento antihorario, definiendo los puertos de salida para cada caso posible. Además se reparó el link roto del caso anterior. El nuevo comando para POX fue:

```
python3 pox.py -verbose openflow.spanning_tree -no-flood -hold-down
openflow.discovery forwarding.l2_antihorario
```

Un ejemplo de ping antihorario entre h1 y h8 efectivo:

```

.17 INFO:openflow.discovery:link detected: 00-00-00-00-00-02,19 -> 00-00-00-00-00-03
.20 INFO:openflow.discovery:link detected: 00-00-00-00-00-03,20 -> 00-00-00-00-00-02
.19 DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-04 2]
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-01 3]
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-03 4]
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-02 5]
DEBUG:openflow.spanning_tree:Spanning tree updated
INFO:openflow.spanning_tree:7 ports changed
DEBUG:openflow.spanning_tree:Spanning tree updated
INFO:openflow.spanning_tree:3 ports changed
DEBUG:openflow.spanning_tree:Spanning tree updated
INFO:openflow.spanning_tree:4 ports changed
DEBUG:openflow.spanning_tree:Spanning tree updated
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-01 3]
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-04 2]
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-03 4]
DEBUG:openflow.spanning_tree:Requested switch features for [00-00-00-00-00-02 5]
DEBUG:openflow.of_01:1 connection aborted

*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (s1, s2)
(s2, s3) (s3, s4) (s4, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
mininet> h1 ping -c 1 h8
PING 10.0.0.8 (10.0.0.8) 56(84) bytes of data:
64 bytes from 10.0.0.8: icmp_seq=1 ttl=64 time=13.7 ms

--- 10.0.0.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/ndev = 13.712/13.712/13.712/0.000 ms
mininet>

```

Figura 6: Ping antihorario

2. Red 2: Dos Caminos

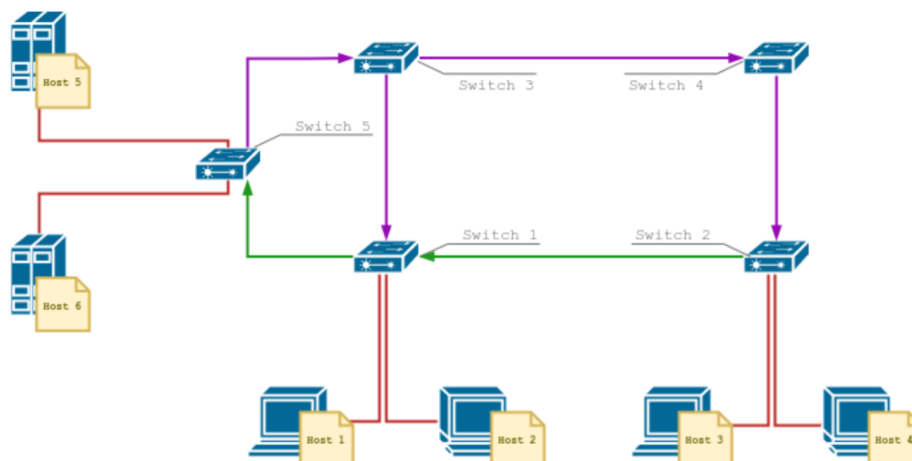


Figura 7: Topología de la red 2.

En este caso se pedía construir una red que requería cumplir con cierto redireccionamiento de rutas en base a lo expuesto en la figura 4, además de considerar que los

host5(h5) y host6(h6) cumplían labor de servidor HTTP, y solo podían recibir mensajes de determinados host (h1 y h2 para h5, y h3 y h4 para h6). Para ello se modificó el componente "l2_learning", agregando código que limitara el envío de paquetes a las restricciones ya nombradas y redireccionando los puertos según la topología presentada.

El comando utilizado en mininet para la ejecución de la topología fue el siguiente:

```
sudo mn -custom topologia2.py -topo red2 -controller remote -switch  
ovsk -mac
```

Y en el CLI de mininet, se inicializaron los servidores HTTP de la siguiente manera:

El comando para la ejecución del controlador con POX fue el siguiente:

```
python3 pox.py -verbose openflow.spanning_tree -no-flood -hold-down  
openflow.discovery forwarding.l2_HTTP
```