Universidad Técnica Federico Santa María
Departamento de Informática
INF256 - Redes de Computadores

Laboratorio 3 - Grupo 01

Red Anillo Simple y Red Dos Caminos

Catalina Sierra H. - catalina.sierrah@usm.cl - Rol 201973557-8
José Southerland S. - jose.southerland@usm.cl - Rol 201973526-8

1. Red Anillo Simple

1.1. Cómo ejecutar

Para ejecutar se deben seguir los siguientes pasos:

1.1.1. Terminal 1: POX

Ejecutar el siguiente comando:

```
docker run -it -rm -privileged -e DISPLAY
-name tarea3
-v /tmp/.X11-unix:/tmp/.X11-unix
-v /lib/modules:/lib/modules
-v "$(pwd)/tarea3/":/home/tarea3/pox/pox/tarea3/:ro
-v "$(pwd)/topologia/":/home/tarea3/pox/pox/topology/:ro
tarea3
```

Y luego:

```
python3 pox.py log.level -DEBUG misc.full_payload tarea3.l2_learning openflow.discovery
openflow.spanning_tree -no-flood -hold-down
```

1.1.2. Terminal 2: Mininet

En una terminal distinta a la anterior, ejecutar el comando:

```
docker exec -it tarea3 bash
cd pox/topology
mn -custom topology.py -topo AnilloSimple -mac -controller remote -switch ovsk
```

1.2. Respuestas

A continuación se detalla la topología de la red de anillo simple:

Host	MAC
h1	00:00:00:00:00:01
h2	00:00:00:00:00:02
h3	00:00:00:00:00:03
h4	00:00:00:00:00:04
h5	00:00:00:00:00:05
h6	00:00:00:00:00:06
h7	00:00:00:00:00:07
h8	00:00:00:00:00:08
h9	00:00:00:00:00:09
h10	00:00:00:00:00:0A

Switch	DPID
s1	1
s2	2
s3	3
s4	4
s5	5

Host	Puerto de Host	Switch	Puerto de Switch	Dirección
h1	0	s1	1	↔
h2	0	s1	2	↔
h3	0	s2	3	↔
h4	0	s2	4	↔
h5	0	s3	5	↔
h6	0	s3	6	↔
h7	0	s4	7	↔
h8	0	s4	8	↔
h9	0	s5	9	↔
h10	0	s5	10	↔

Switch 1	Puerto de Switch 1	Switch 2	Puerto de Switch 2	Dirección
s1	11	s2	12	↔
s2	13	s3	14	↔
s3	15	s5	16	↔
s5	17	s4	18	↔
s4	19	s1	20	↔

Al implementar los host numerados del 1 al 10 junto con sus respectivos switches se observa como es que estos se encontaban correctamente conectados.

```
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (h5, s3) (h6, s3) (h7, s4) (h8, s4) (h9, s5) (h10, s5) (s1, s2)
(s2, s3) (s3, s5) (s4, s1) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s2-eth3
h4 h4-eth0:s2-eth4
h5 h5-eth0:s3-eth5
h6 h6-eth0:s3-eth6
h7 h7-eth0:s4-eth7
h8 h8-eth0:s4-eth8
h9 h9-eth0:s5-eth9
h10 h10-eth0:s5-eth10
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth11:s2-eth12 s1-eth20:s4-eth19
s2 lo:  s2-eth3:h3-eth0 s2-eth4:h4-eth0 s2-eth12:s1-eth11 s2-eth13:s3-eth14
s3 lo:  s3-eth5:h5-eth0 s3-eth6:h6-eth0 s3-eth14:s2-eth13 s3-eth15:s5-eth16
s4 lo:  s4-eth7:h7-eth0 s4-eth8:h8-eth0 s4-eth18:s5-eth17 s4-eth19:s1-eth20
s5 lo:  s5-eth9:h9-eth0 s5-eth10:h10-eth0 s5-eth16:s3-eth15 s5-eth17:s4-eth18
c0
```

Figura 1: Anillo Simple

Luego para experimentar que sucede al eliminar una conexión utilizamos el comando `link s1 s2 down` desechando la conexión entre el switch 1 y switch 2.

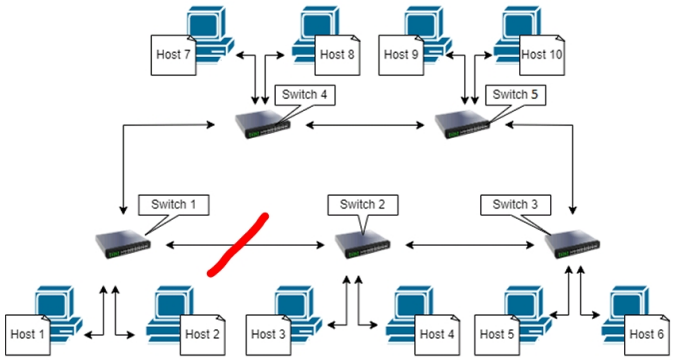

Figura 2: Representación del corte del link entre Switch 1 y Switch 2

Posteriormente utilizando el comando `pingall` se chequea lo que sucede en las conexiones de los Host, parte superior de la figura 3, donde se evidencia la pérdida de 8 % paquetes enviados. Podemos ver que el host 1 no tiene conexión con los host que están a la derecha del Switch 2. Mientras que cuando pasamos al host 2 se podría decir que el controlador encuentra otro camino (link) para poder tener una conexión exitosa, la cual corresponde a dar la vuelta pasando por los Switch 3, 5 y 4 para llegar al switch 1. Finalmente desde el host 3 en adelante el controlador ya conoce los caminos por donde proceder, por lo tanto logra entregar los paquetes restantes. Por último realizamos un `pingall` final y evidenciamos que se logra tener una conexión exitosa entre todos los host y switches.

```
mininet> link s1 s2 down
mininet> ping all
*** Unknown command: ping all
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X X X h7 h8 X X
h2 -> h1 X h4 h5 X h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 8% dropped (82/90 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
```

Figura 3: Ejecución de `pingall`

2. Red Dos Caminos

2.1. Cómo ejecutar

Para ejecutar se deben seguir los siguientes pasos:

2.1.1. Terminal 1: POX

Ejecutar el siguiente comando:

```
docker run -it -rm -privileged -e DISPLAY
-name tarea3
-v /tmp/.X11-unix:/tmp/.X11-unix
-v /lib/modules:/lib/modules
-v "$(pwd)/tarea3/":/home/tarea3/pox/pox/tarea3/:ro
-v "$(pwd)/topologia/":/home/tarea3/pox/pox/topology/:ro
tarea3


python3 pox.py log.level -DEBUG misc.full_payload tarea3.red2 openflow.discovery
openflow.spanning_tree -no-flood -hold-down
```

2.1.2. Terminal 2: Mininet

En una terminal distinta a la anterior, ejecutar el comando:

```
docker exec -it tarea3 bash
cd pox/topology
mn -custom topology.py -topo DosCaminos -mac -controller remote -switch ovsk
```

2.2. Respuestas

A continuación se presenta la topología de la red de dos caminos:

Host	MAC
h1	00:00:00:00:00:01
h2	00:00:00:00:00:02
h3	00:00:00:00:00:03
h4	00:00:00:00:00:04
h5	00:00:00:00:00:05
h6	00:00:00:00:00:06
h7 (servidor)	00:00:00:00:00:07
h8 (servidor)	00:00:00:00:00:08

Switch	DPID
s1	1
s2	2
s3	3
s4	4
s5	5

Host	Puerto de Host	Switch	Puerto de Switch	Dirección
h1	0	s1	1	↔
h2	0	s1	2	↔
h3	0	s2	3	↔
h4	0	s2	4	↔
h5	0	s3	5	↔
h6	0	s3	6	↔
h7	22	s5	21	↔
h8	20	s5	19	↔

Switch 1	Puerto de Switch 1	Switch 2	Puerto de Switch 2	Dirección
s1	7	s2	8	→
s2	9	s3	10	→
s3	11	s4	12	→
s4	13	s5	14	→
s4	15	s1	16	→
s5	17	s1	18	→

En la siguiente imágen se evidencia como es que h7 y h8 se convierten en servidores de tipo HTTP.

```
mininet> h7 python3 -m http.server 80 &
mininet> h8 python3 -m http.server 80 &
```

Figura 4: Conversión a servidor de Host 7 y Host 8

Ahora utilizando el comando `wget` es que se logra evidenciar la comunicación entre el Host 1 y/o Host 2 y el Host 7, ya transformado a servidor.

Figura 5: Comunicacón Host 1 y servidor h7

Figura 6: Comunicación Host 2 y servidor h7

Luego dado que en la tarea se especifica que Host 1 y Host 2 debe comunicarse únicamente con el Host 7 realizamos una consulta desde el Host 1 o Host 2 hacia el Host 8, evidenciando el rechazo.

Figura 7: Conexión rechazada entre Host 1 y servidor h8

De la misma forma ocurre este proceso entre el Host 3 y/o Host 4 y el Host 8.

Figura 8: Comunicación entre Host 4 y servidor h8

Figura 9: Comunicación entre Host 3 y servidor h8

Por último se muestra como esque la comunicación host-host se encuentra bloqueada.

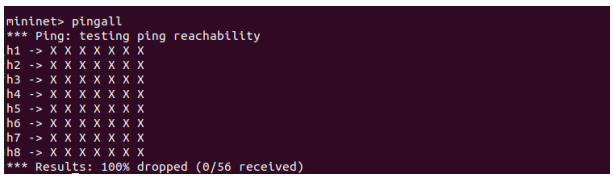

Figura 10: Comunicación bloqueada entre host