



## LABORATORIO 3.

### REDES DE COMPUTADORES

Diego Rosales León, 201810531-7, [diego.rosalesl@sansano.usm.cl](mailto:diego.rosalesl@sansano.usm.cl)  
Alan Zapata Silva, 201956567-2, [alan.zapata@usm.cl](mailto:alan.zapata@usm.cl)

27 de Junio del 2022

# Índice

<b>1. Consideraciones</b>	<b>2</b>
<b>2. Problemas</b>	<b>3</b>
2.1. Problema 1: Anillo Simple . . . . .	3
2.1.1. Topología . . . . .	3
2.1.2. Descripción de la topología . . . . .	3
2.1.3. Funcionamiento de la tarea . . . . .	4
2.1.4. Comentario . . . . .	4
2.2. Problema 2: Dos Caminos . . . . .	6
2.2.1. Topología . . . . .	6
2.2.2. Descripción de la topología . . . . .	6
2.2.3. Funcionamiento de la tarea . . . . .	6
2.2.4. Comentario . . . . .	7

# 1. Consideraciones

Para la ejecución correcta de la tarea, se utilizó Ubuntu 22.04 y se siguieron todos los pasos de instalación dados por los ayudantes.

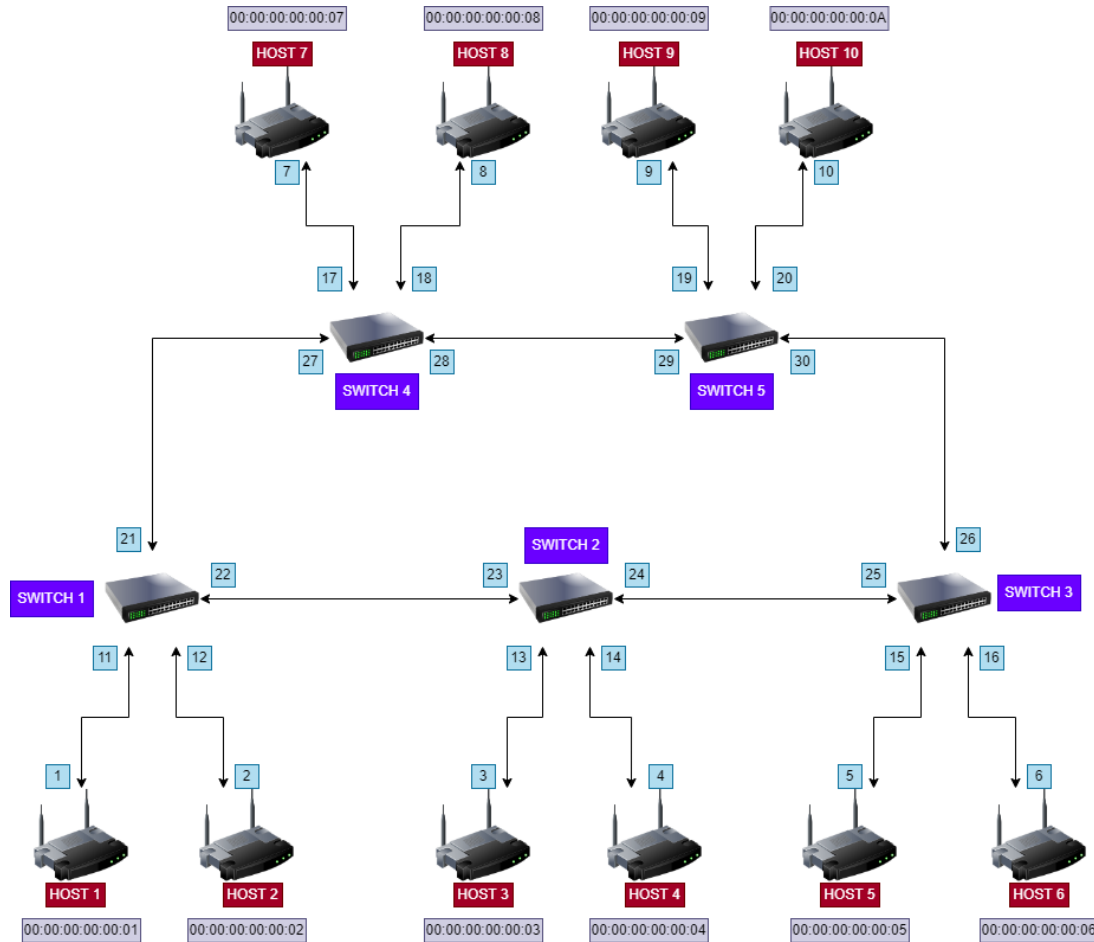
Para los dos problemas se utilizará el mismo archivo de topología, que debe estar en la carpeta **topology**, y archivos que harán de controladores que deben estar en la carpeta **tarea3** (Si se quiere trabajar en la pregunta 1, se debe utilizar el archivo `learning1.py`, si se quiere trabajar con la pregunta 2, se debe utilizar el archivo `learning2.py`.) luego se abren dos terminales para probar el funcionamiento de la tarea.

## 2. Problemas

### 2.1. Problema 1: Anillo Simple

#### 2.1.1. Topología

Para describir la topología del problema, utilizamos la siguiente imagen:



Donde se aprecian los diferentes host, switches, MACs, y sus puertos necesarios para la ejecución de esta parte de la tarea.

#### 2.1.2. Descripción de la topología

Para definir el controlador, se definieron 5 switches,  $s1$ ,  $s2$ ,  $s3$ ,  $s4$ ,  $s5$ , a cada uno se le agregó un ID (dpid) diferente (1,2,3,4,5, respectivamente).

Los host y enlaces se definieron, según la imagen anterior, el primero tiene un número de host con una MAC correspondiente, y para el segundo, cada enlace tiene un puerto respectivo.

### 2.1.3. Funcionamiento de la tarea

Para el funcionamiento de la tarea, se debe correr:

- **Controlador:** El controlador, se debe correr en una de las consolas el archivo `learning1.py` (el archivo debe estar en la carpeta `tarea3`). Con el código:  
`python3 pox.py log.level -DEBUG misc.full_payload tarea3.learning1  
openflow.discovery openflow.spanning_tree -no-flood -hold-down.`
- **Topología:** La topología, se debe ejecutar en otra consola `pox` el archivo `topology.py` (el archivo debe estar en la carpeta `topologia`). Con el código:  
`mn -custom topology.py -topo Red1 -mac -controller remote -switch ovsk.`

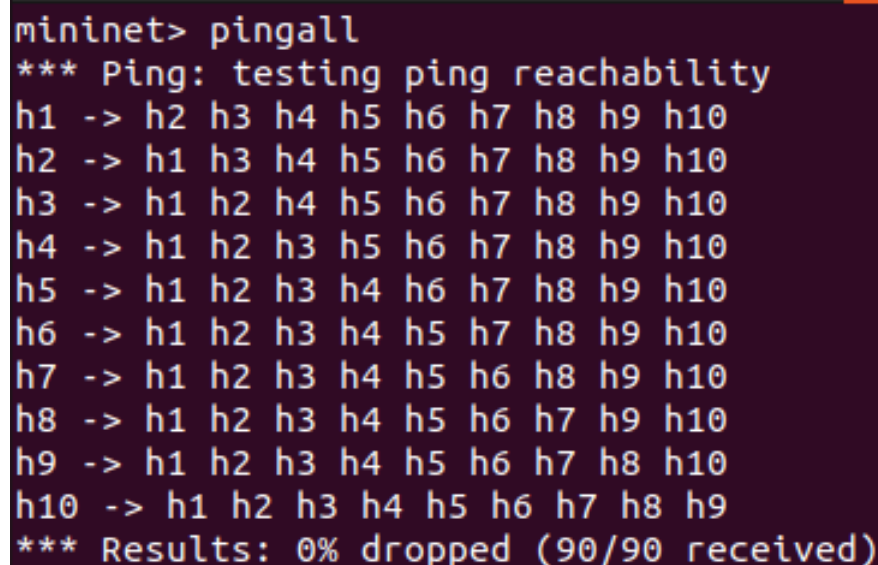
Al tener ambos archivos corriendo, existirá la comunicación entre ellos.

Para comprobar la comunicación entre ambos archivos, se debe aplicar el comando **pingall** en la consola de topología, esto mostrará una tabla indicando todas las conexiones de cada host y una X en caso de que la conexión no exista.

Luego, para borrar uno de los enlaces entre switches, en la misma línea de comando se debe utilizar el comando **link s p down** y para levantar otro enlace se debe usar **link s p up**, siendo **s y p** los switches entre los que se borrará o levantará el enlace.

### 2.1.4. Comentario

Al aplicar el primer `pingall`, se puede apreciar que todos los host se comunican con todos los host.



```
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)
```

Luego se aplicó el comando para quitar el link entre el switch 1 y el switch 2, Y nuevamente se aplicó un pingall, obteniendo:

```
mininet> link s1 s2 down
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X X h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 X 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
```

Donde se puede apreciar que algunos host no pudieron comunicarse con otros host. Al aplicar nuevamente un pingall se obtiene:

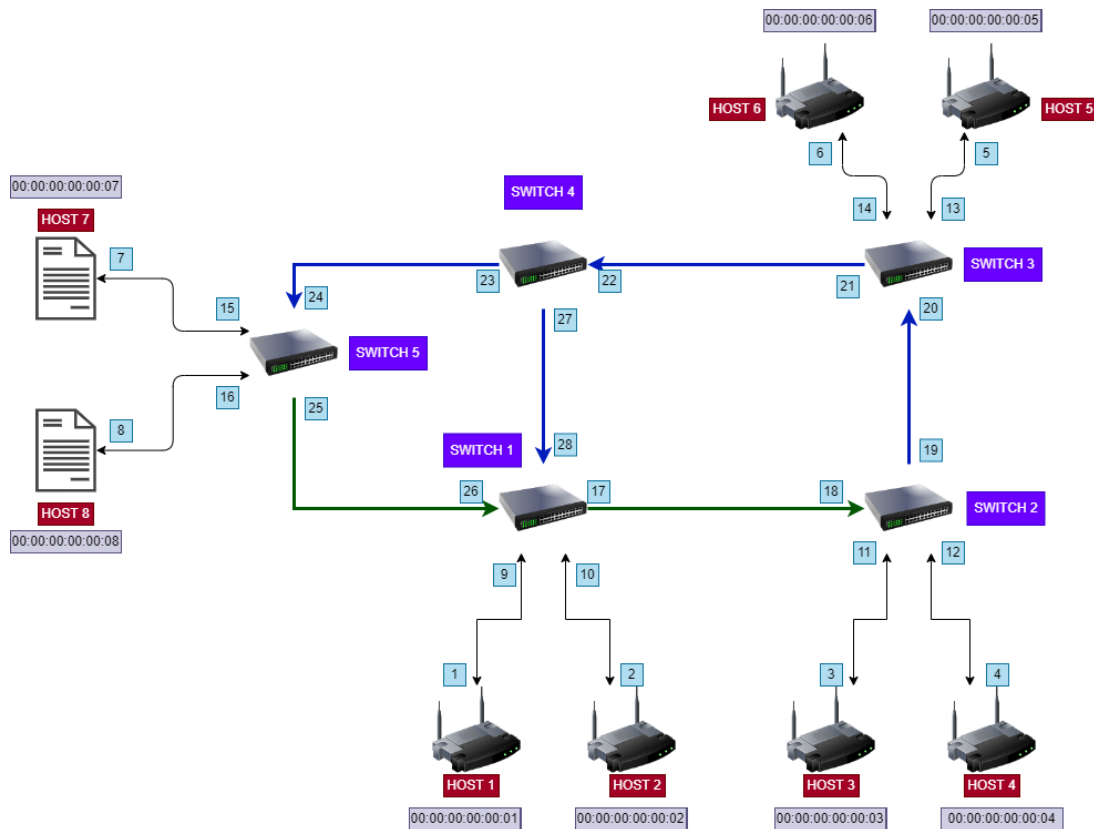
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> 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)
```

Es decir, que aunque algunos host no pudieran comunicarse entre sí, los switchs aprenden nuevos caminos para poder comunicarse, esto se debe a que están bajo el protocolo de Openflow y los switch actualizan sus tablas de flujo.

## 2.2. Problema 2: Dos Caminos

### 2.2.1. Topología

Para describir la topología del problema, utilizamos la siguiente imagen:



Donde se aprecian los diferentes host, switches, MACs, y sus puertos necesarios para la ejecución de esta parte de la tarea.

### 2.2.2. Descripción de la topología

Para definir el controlador, se definieron 5 switches,  $s1$ ,  $s2$ ,  $s3$ ,  $s4$ ,  $s5$ , a cada uno se le agregó un ID (dpid) diferente (1,2,3,4,5, respectivamente).

Los host y enlaces se definieron, según la imagen anterior, el primero tiene un número de host con una MAC correspondiente, y para el segundo, cada enlace tiene un puerto respectivo.

### 2.2.3. Funcionamiento de la tarea

Para el funcionamiento de la tarea, se debe correr:

- **Controlador:** El controlador, se debe correr en una de las consolas el archivo `learning2.py` (el archivo debe estar en la carpeta `tarea3`). Con el código:  

```
python3 pox.py log.level -DEBUG misc.full_payload tarea3.learning2
openflow.discovery openflow.spanning_tree -no-flood -hold-down.
```

- **Topología:** La topología, se debe ejecutar en otra consola pox el archivo topology.py (el archivo debe estar en la carpeta topologia). Con el código:  
`mn -custom topology.py -topo Red2 -mac -controller remote -switch ovsk.`

Para convertir los host a un servidor de tipo http, se debe escribir lo siguiente en la terminal de la topología:

`hostx python3 -m http.server 80 &` siendo **hostx** el host que se convertirá a un servidor http.

Luego, para realizar una consulta http, se debe poner en la terminal de la topología el código, `s wget -O hX`, siendo **s** un host definido y **hX** el host de tipo http.

#### 2.2.4. Comentario

Para correr lo anterior, se usó la siguiente imagen:

```
mininet> h7 python3 -m http.server 80 &
mininet> h8 python3 -m http.server 80 &
mininet> h2 wget -O - h7
```

Donde primero se hicieron los servidores de tipo http (host 7 y 8). Luego se aplicó una consulta de tipo http, obteniendo:

```
--2022-06-27 20:41:12-- http://10.0.0.7/
Connecting to 10.0.0.7:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 344 [text/html]
Saving to: 'STDOUT'

-          0%[          ]          0  --.-KB/s
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/s
trict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Directory listing for /</title>
</head>
<body>
<h1>Directory listing for /</h1>
<hr>
<ul>
<li><a href="topology.py">topology.py</a></li>
</ul>
<hr>
</body>
</html>
-          100%[=====]          344  --.-KB/s    in 0s

2022-06-27 20:41:13 (1.98 MB/s) - written to stdout [344/344]
```

Donde es una consulta realizada con éxito.

Finalmente, si se aplica un pingall, se puede comprobar que ninguno de los hosts pueden comunicarse entre sí.