

Trabajo práctico sobre Netkit, STP, Port Bonding y VLAN

Mauro Bachur, Francisco Elías, y Camilo Jimenez

Mayo de 2019

1 Introducción teórica

En este trabajo práctico tenemos como objetivo probar cómo funciona el bridge/switch, el protocolo stp, el vlan y el port bonding, a través de la utilización de la herramienta de creación de dispositivos virtuales llamado netkit.

1.1 Netkit

Netkit es una herramienta que permite la creación de dispositivos de red virtuales donde podemos realizar diferentes experimentos.

1.2 Bridge

Un bridge es un dispositivo de interconexión de redes de computadoras que opera en la capa 2 del modelo OSI. Se encarga de interconectar dos segmentos de red haciendo el pasaje de datos de una red a la otra.

1.3 Switch

Un switch es un dispositivo de interconexión encargado de unir dos o mas equipos de red entre si para formar una red de area local (LAN) que tiene que seguir el estándar conocido como Ethernet (IEEE 802.3).

1.4 STP

El STP es un protocolo que usa un algoritmo denominado “Spanning-tree Algorithm” el cual crea la topología de la red en donde se aplica. Con esto creado, se procede a la eleccion de un bridge como root, que va a ser elegida a partir del “bridge priority”, un numero que posee cada uno que por default va a ser 32,768. Si el valor no es modificado, todos los bridge van a tener el mismo valor, entonces en este caso se procede a elegir con la MAC ID que tiene cada bridge (unica), siendo el de menor MAC ID el que va a ser el bridge root. Una vez establecido el root, se procede a establecer cuales conexiones va a estar bloqueadas y cuales van a estar en funcionamiento. Esto lo hace viendo la mejor conexion para llegar al bridge root y se lo pone en estado de funcionamiento, todas las demas que no se usan para una mejor llegada, son puestas en estado de bloqueo.

Si la configuración de STP cambia, o si un segmento en la red redundante llega a ser inalcanzable, el algoritmo reconfigura los enlaces y restablece la conectividad, activando uno de los enlaces de reserva. Es decir, que el algoritmo vuelve a crear la topología, vuelve a crear un bridge root, y vuelve a crear conexiones en estado de bloqueo y en estado de funcionamiento.

En resumen: “El STP permanece vigente hasta que ocurre un cambio en la topología, situación que el protocolo es capaz de detectar de forma automática. Cuando ocurre uno de estos cambios, el puente raíz actual redefine la topología del árbol de expansión o se elige un nuevo puente raíz.”

1.5 Port Bonding

Port bonding es un método para la combinación de múltiples conexiones de red en paralelo para mejorar lo que una sola conexión podría hacer. Soluciona los problemas de ethernet tales como limitaciones en el ancho de banda y la carencia de resistencia. Port bonding posee 7 modos distintos, cada uno varía en como el tráfico es enviado a través de la interfaz creada. Los mismos son:

- Modo 0 (balance-rr)
- Modo 1 (active-backup)
- Modo 2 (balance-xor)
- Modo 3 (broadcast)
- Modo 4 (802.3ad)
- Modo 5 (balance-tlb)
- Modo 6 (balance-alb)

1.6 VLAN

Vlan es un método para crear un dominio de difusión dentro de una misma red, por ejemplo tengo 4 PCs y hago una vlan entre 2 PCs y otra vlan entre las otras 2 PCs, entonces habrá un dominio de difusión entre las primeras 2 PCs y otro entre las PCs restantes.

1.7 Sniffer

Un sniffer, en este contexto, es una PC que se encuentra en modo promiscuo, o sea, que se encuentra conectada a una red compartida y se encarga de capturar todo el tráfico que circula por la misma.

2 Experimentos

Funcionamiento del Netkit

Para armar un laboratorio de Netkit, se debe crear una carpeta con el nombre deseado y crear dentro de la misma los siguientes archivos y carpetas:

- Un archivo lab.conf en donde se introduzca la topología del laboratorio de forma escrita.
- La misma cantidad de carpetas que de dispositivos queramos. Las mismas deberán tener el mismo nombre que el especificado en el archivo lab.conf.

- De forma opcional, podemos crear archivos `.startup` para cada dispositivo que se encargan de ejecutar los comandos escritos en el al iniciar el laboratorio.

Una vez cumplidos estos requisitos, basta con escribir el siguiente comando para iniciar el laboratorio:

```
lstart
```

En el caso de que queramos iniciar los dispositivos en forma paralela, se usa el comando:

```
lstart -p
```

Por último, si queremos iniciar solo un dispositivo, basta con escribir:

```
lstart <dispositivo>
```

Cuando terminemos de trabajar con los dispositivos, existen dos comandos para "apagarlos":

```
lcrash  
lhalt
```

El primero apaga los dispositivos borrando el historial de comandos en los mismos, por lo que no se mantiene ningún comando usados en los mismo. El segundo apaga los dispositivos de forma que al inciarlos en otro momento, todos los comandos ingresados se mantienen.

Siguiendo este procedimiento, realizamos los experimentos que se encuentran detallados abajo.

Link a GitHub con todos los escenarios: <https://github.com/franelias/netkit.git>

2.1 Escenario Bridge/switch

Para este primer experimentos, armamos un laboratorio utilizando 4 PCs (PC1, PC2, PC3, PC4), en el cual cada par está conectado a un cable A y B (PC1 y PC2 al cable A y PC3 y PC4 al cable B). Estos cables están unidos a un switch y a un sniffer. Los `.startup` del switch contenía los siguientes comandos:

```
brctl addbr br0  
brctl addif br0 eth0  
brctl addif br0 eth1  
ifconfig eth0 up
```

```

ifconfig eth1 up
ifconfig br0 up

```

Realizamos este primer experimento para poder aprender como armar un switch y ver como funciona. Para ver esto, pingueamos desde una PC a cualquier otra, y observamos el sniffer, que nos mostraba como el switch enviaba el broadcast a todas las PCs, pero solo le llegaba el paquete a la PC pingueada.

```

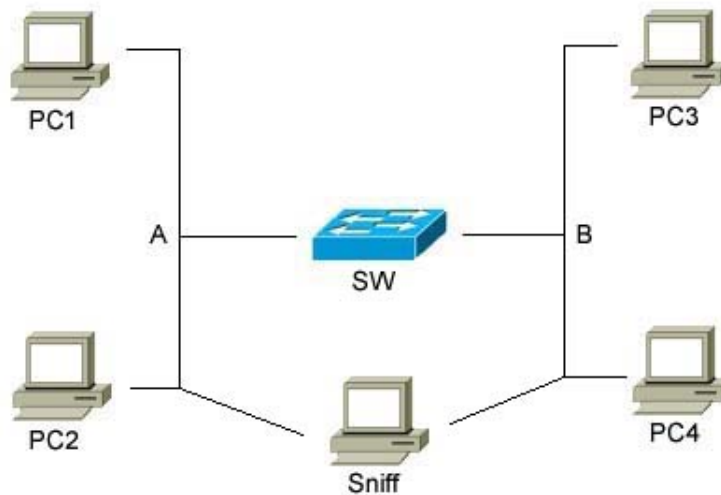
sniff
stener report v2, 1 group record(s), length 28
05:11:07.497303 IP6 fe80::f01b:4eff:fe46:e31a > ff02::16: HBH ICMP6, multicast l
istener report v2, 1 group record(s), length 28
05:11:07.757204 IP6 fe80::f01b:4eff:fe46:e31a > ip6-allrouters: ICMP6, router so
licitation, length 16
05:11:07.770753 IP6 fe80::f081:23ff:fe4d:f529 > ip6-allrouters: ICMP6, router so
licitation, length 16
05:11:07.850003 IP6 fe80::8cd3:81ff:fe2b:643e > ip6-allrouters: ICMP6, router so
licitation, length 16
05:11:08.029955 IP6 fe80::984b:7eff:fe0e:37e3 > ff02::16: HBH ICMP6, multicast l
istener report v2, 1 group record(s), length 28
05:11:08.629888 IP6 fe80::345f:5eff:fe38:15e3 > ip6-allrouters: ICMP6, router so
licitation, length 16
05:11:09.098849 IP6 fe80::d832:1eff:fe3c:5c5 > ip6-allrouters: ICMP6, router sol
icitation, length 16
05:11:09.129940 IP6 fe80::2cfb:99ff:fe22:3462 > ip6-allrouters: ICMP6, router so
licitation, length 16
05:11:09.478879 IP6 fe80::3cba:81ff:fe48:2be > ip6-allrouters: ICMP6, router sol
icitation, length 16
05:11:09.558885 IP6 fe80::d832:1eff:fe3c:5c5 > ff02::16: HBH ICMP6, multicast li
stener report v2, 1 group record(s), length 28
05:11:09.659997 IP6 fe80::984b:7eff:fe0e:37e3 > ip6-allrouters: ICMP6, router so
licitation, length 16

```

Switch aprendiendo los dispositivos que tiene conectado.

sniff	PC1
, length 64	192.168.0.2 ping statistics ---
00:04:20.663936 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1282, seq 12,	3 packets transmitted, 3 received, 0% packet loss, time 2008ms
length 64	rtt min/avg/max/ndev = 0.230/0.477/0.642/0.258 ms
00:04:21.663716 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1282, seq 13	PC1: # ping 192.168.0.2
, length 64	PING 192.168.0.2 (192.168.0.2) 56(84) bytes of data.
00:04:21.663775 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1282, seq 13,	64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=0.164 ms
length 64	64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=0.286 ms
00:04:22.663827 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1282, seq 14	64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=0.256 ms
, length 64	64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=0.313 ms
00:04:22.663932 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1282, seq 14,	64 bytes from 192.168.0.2: icmp_seq=5 ttl=64 time=0.279 ms
length 64	64 bytes from 192.168.0.2: icmp_seq=6 ttl=64 time=0.262 ms
00:04:23.663774 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1282, seq 15	64 bytes from 192.168.0.2: icmp_seq=7 ttl=64 time=0.377 ms
, length 64	64 bytes from 192.168.0.2: icmp_seq=8 ttl=64 time=0.216 ms
00:04:23.663838 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1282, seq 15,	64 bytes from 192.168.0.2: icmp_seq=9 ttl=64 time=0.271 ms
length 64	64 bytes from 192.168.0.2: icmp_seq=10 ttl=64 time=0.158 ms
00:04:24.663833 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1282, seq 16	64 bytes from 192.168.0.2: icmp_seq=11 ttl=64 time=0.261 ms
, length 64	64 bytes from 192.168.0.2: icmp_seq=12 ttl=64 time=0.262 ms
00:04:24.663949 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1282, seq 16,	64 bytes from 192.168.0.2: icmp_seq=13 ttl=64 time=0.160 ms
length 64	64 bytes from 192.168.0.2: icmp_seq=14 ttl=64 time=0.273 ms
00:04:25.663833 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1282, seq 17	64 bytes from 192.168.0.2: icmp_seq=15 ttl=64 time=0.195 ms
, length 64	64 bytes from 192.168.0.2: icmp_seq=16 ttl=64 time=0.303 ms
00:04:25.663928 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1282, seq 17,	64 bytes from 192.168.0.2: icmp_seq=17 ttl=64 time=0.266 ms
length 64	

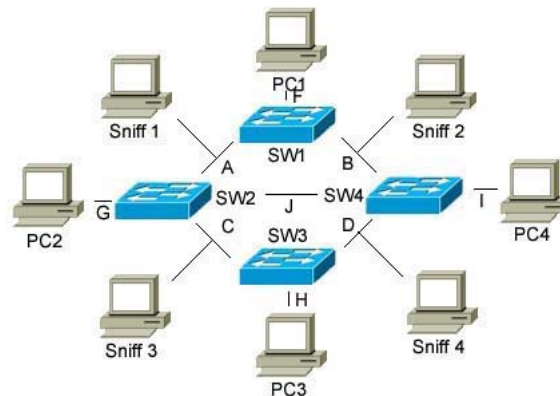
El sniffer pudo captar que las conexiones estaban realizando de forma correcta y el switch creado estaba funcionando de manera adecuada.



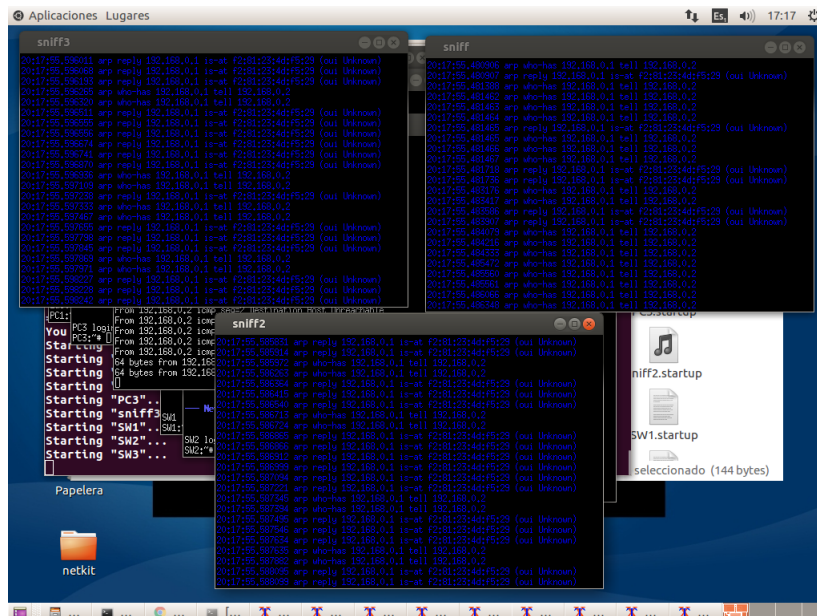
Topología del laboratorio.

2.2 Escenario STP

Utilizamos 4 PCs (PC1, PC2, PC3, PC4) y las conectamos mediante 4 switches (SW1, SW2, SW3, SW4), formando un bucle. Por último, colocamos 4 sniffers, que serán los encargados de registrar todo lo que pase por los cables.

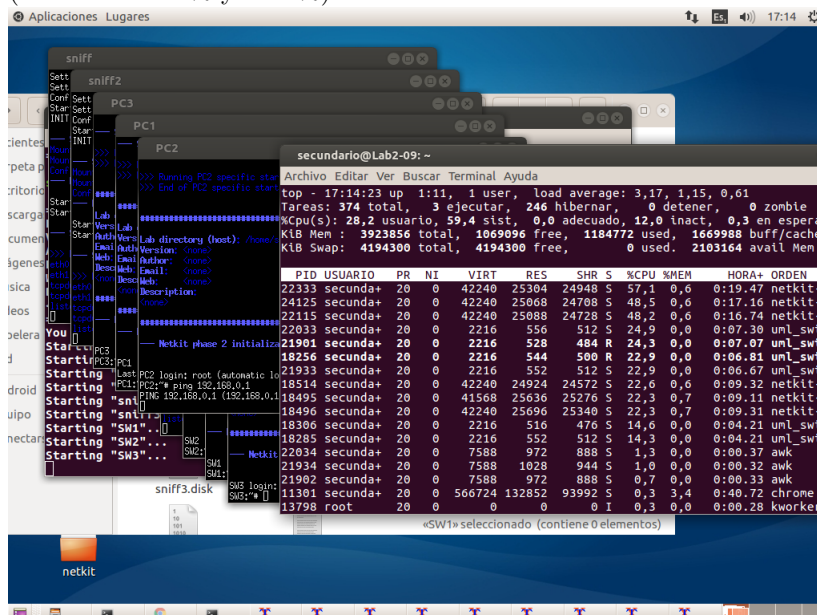


Cuando encendimos el laboratorio y pingueamos la PC2 a través de la PC1, los sniffers mostraron la tormenta de broadcast que se produjo por conectar los switches de esa manera.



Tormenta de broadcast

Paralelamente, otra consola que ejecutaba el comando top mostró que al realizar este comando en la PC1, el uso de la cpu aumentó exponencialmente (alrededor el 40% y el 60%).



Comando top

Esto se puede arreglar activando el protocolo STP con el siguiente procedimiento:

- Apagar el switch:
`ifconfig br0 down`
- Activar el protocolo STP en el mismo:
`brctl stp br0 on`
- Activarlo de vuelta:
`ifconfig br0 up`

Realizado el procedimiento, a través del comando:

```
brctl showstp br0
```

vimos como el SW2 fue el seleccionado como root, con la característica adicional de que era el único que no poseía interfaces bloqueadas por el protocolo.

SW1				SW4			
port id	8001	state	forwarding	port id	8002	state	forwarding
designated root	8000,06dfb0cc4366	path cost	100	designated root	8000,06dfb0cc4366	path cost	100
designated bridge	8000,06dfb0cc4366	message age timer	18,67	designated bridge	8000,1e3cec37ee62	message age timer	0,00
designated port	8001	forward delay timer	0,00	designated port	8002	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	100	hold timer	0,88
flags				flags			
eth1 (2)				eth2 (3)			
port id	8002	state	blocking	port id	8003	state	forwarding
designated root	8000,06dfb0cc4366	path cost	100	designated root	8000,06dfb0cc4366	path cost	100
designated bridge	8000,1e3cec37ee62	message age timer	18,66	designated bridge	8000,1e3cec37ee62	message age timer	0,00
designated port	8002	forward delay timer	0,00	designated port	8003	forward delay timer	0,00
designated cost	100	hold timer	0,00	designated cost	100	hold timer	0,88
flags				flags			
eth2 (3)				eth3 (4)			
port id	8003	state	forwarding	port id	8004	state	forwarding
designated root	8000,06dfb0cc4366	path cost	100	designated root	8000,06dfb0cc4366	path cost	100
designated bridge	8000,3a370dd0b666	message age timer	0,00	designated bridge	8000,06dfb0cc4366	message age timer	13,44
designated port	8003	forward delay timer	0,00	designated port	8004	forward delay timer	0,00
designated cost	100	hold timer	0,09	designated cost	0	hold timer	0,00
flags				flags			
SW1:~# []				SW4:~# []			
SW2				SW3			
port id	8002	state	forwarding	port id	8001	state	forwarding
designated root	8000,06dfb0cc4366	path cost	100	designated root	8000,06dfb0cc4366	path cost	100
designated bridge	8000,06dfb0cc4366	message age timer	0,00	designated bridge	8000,06dfb0cc4366	message age timer	13,65
designated port	8002	forward delay timer	0,00	designated port	8002	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	0	hold timer	0,00
flags				flags			
eth2 (3)				eth1 (2)			
port id	8003	state	forwarding	port id	8002	state	blocking
designated root	8000,06dfb0cc4366	path cost	100	designated root	8000,06dfb0cc4366	path cost	100
designated bridge	8000,06dfb0cc4366	message age timer	0,00	designated bridge	8000,1e3cec37ee62	message age timer	15,64
designated port	8003	forward delay timer	0,00	designated port	8001	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	100	hold timer	0,00
flags				flags			
eth3 (4)				eth2 (3)			
port id	8004	state	forwarding	port id	8003	state	forwarding
designated root	8000,06dfb0cc4366	path cost	100	designated root	8000,06dfb0cc4366	path cost	100
designated bridge	8000,06dfb0cc4366	message age timer	0,00	designated bridge	8000,06dfb0cc4366	message age timer	0,00
designated port	8004	forward delay timer	0,00	designated port	8003	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	100	hold timer	1,25
flags				flags			
SW2:~# []				SW3:~# []			

Comando mostrando las interfaces bloqueadas en cada switch, y el switch root designado

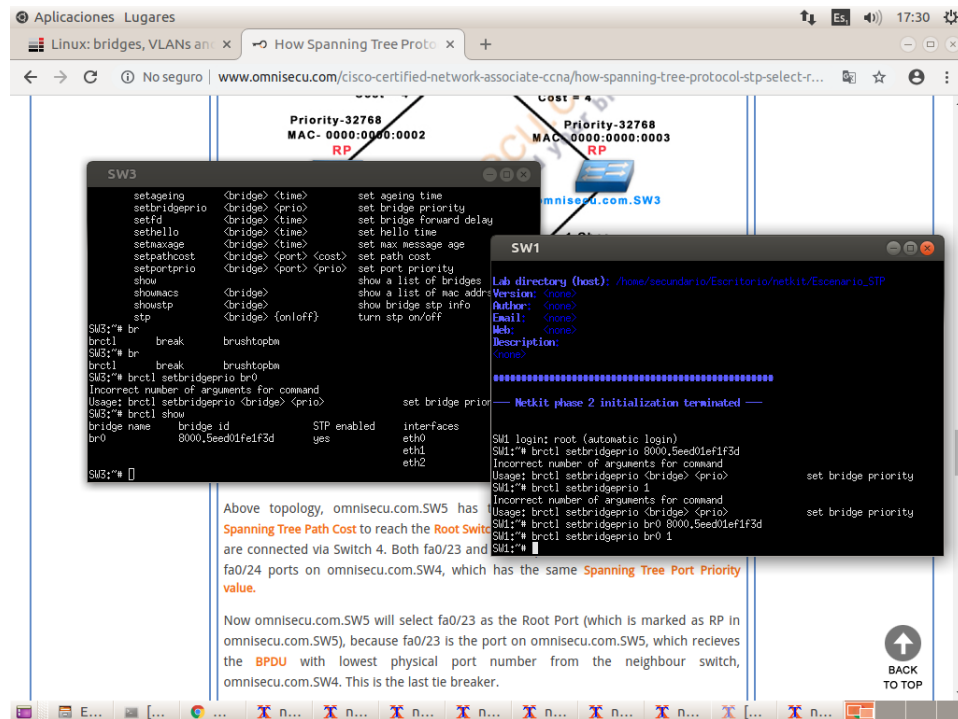
Qué hace

Evita la tormenta de broadcast asignándole estados a los cables que conectan en las computadoras. Un estado es el de bloqueado, el cual los datos no fluyen por ese cable y va a ser activado (o no) en caso de que se modifique el STP y el otro estado es el de funcionamiento el cual permite los flujos de datos por el mismo. La elección de los estados de los cables fue explicado en el punto 1.4

Cómo establecer un root manualmente

Puede llegar un momento en que nosotros queramos elegir de forma manual un switch que actúe como root, para ello utilizamos el siguiente comando en el switch elegido:

```
brctl setbridgeprio br0 <número>
```



Cambio del número del switch, convirtiendolo en el root

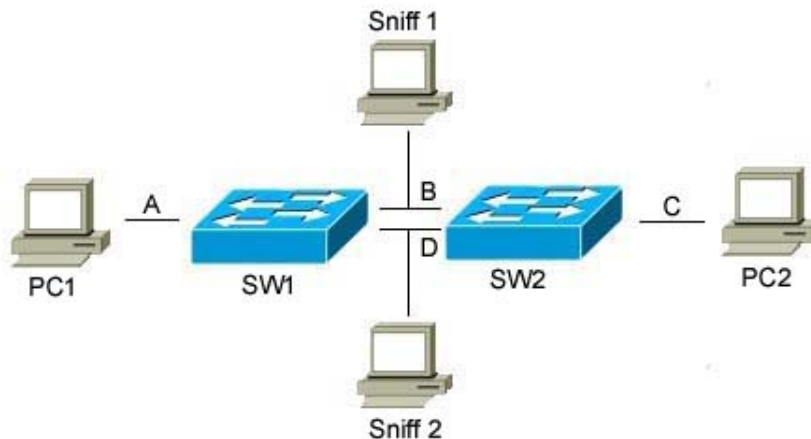
En nuestro laboratorio, cambiamos el numero del SW4 a 1, convirtiendolo en el root, y a través del comando `brctl showstp br0`, analizamos como quedo el arbol topologico.

SW1				SW4			
port id	8001	state	blocking	port id	8002	state	forwarding
designated root	0001.1e3cec97ee62	path cost	100	designated root	0001.1e3cec97ee62	path cost	100
designated bridge	8000.06dfbbcc4366	message age timer	19,07	designated bridge	0001.1e3cec97ee62	message age timer	0,00
designated port	8001	forward delay timer	0,00	designated port	8002	forward delay timer	0,00
designated cost	100	hold timer	0,00	designated cost	0	hold timer	0,40
flags				flags			
eth1 (2)				eth2 (3)			
port id	8002	state	forwarding	port id	8003	state	forwarding
designated root	0001.1e3cec97ee62	path cost	100	designated root	0001.1e3cec97ee62	path cost	100
designated bridge	0001.1e3cec97ee62	message age timer	19,08	designated bridge	0001.1e3cec97ee62	message age timer	0,00
designated port	8002	forward delay timer	0,00	designated port	8003	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	0	hold timer	0,40
flags				flags			
eth2 (3)				eth3 (4)			
port id	8003	state	forwarding	port id	8004	state	forwarding
designated root	0001.1e3cec97ee62	path cost	100	designated root	0001.1e3cec97ee62	path cost	100
designated bridge	8000.3a2700de0b66	message age timer	0,00	designated bridge	0001.1e3cec97ee62	message age timer	0,00
designated port	8003	forward delay timer	0,00	designated port	8004	forward delay timer	0,00
designated cost	100	hold timer	0,06	designated cost	0	hold timer	0,40
flags				flags			
SW1:~* []				SW4:~* []			
SW2				SW3			
port id	8002	state	forwarding	port id	8001	state	blocking
designated root	0001.1e3cec97ee62	path cost	100	designated root	0001.1e3cec97ee62	path cost	100
designated bridge	8000.06dfbbcc4366	message age timer	0,00	designated bridge	8000.06dfbbcc4366	message age timer	18,17
designated port	8002	forward delay timer	0,00	designated port	8002	forward delay timer	0,00
designated cost	100	hold timer	0,34	designated cost	100	hold timer	0,00
flags				flags			
eth2 (3)				eth1 (2)			
port id	8003	state	forwarding	port id	8002	state	forwarding
designated root	0001.1e3cec97ee62	path cost	100	designated root	0001.1e3cec97ee62	path cost	100
designated bridge	8000.06dfbbcc4366	message age timer	0,00	designated bridge	0001.1e3cec97ee62	message age timer	18,18
designated port	8003	forward delay timer	0,00	designated port	8001	forward delay timer	0,00
designated cost	100	hold timer	0,34	designated cost	0	hold timer	0,00
flags				flags			
eth3 (4)				eth2 (3)			
port id	8004	state	forwarding	port id	8003	state	forwarding
designated root	0001.1e3cec97ee62	path cost	100	designated root	0001.1e3cec97ee62	path cost	100
designated bridge	0001.1e3cec97ee62	message age timer	19,37	designated bridge	8000.5eed01felf3d	message age timer	0,00
designated port	8004	forward delay timer	0,00	designated port	8003	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	100	hold timer	0,00
flags				flags			
SW2:~* []				SW3:~* []			

brctl showstp br0 mostrando que interfaces estan bloqueadas.

2.3 Port bonding

Para realizar el laboratorio, utilizamos dos PCs, nombradas PC1 y PC2 respectivamente y dos switches unidos entre si por dos cables.



Topologia del laboratorio.

En el archivo .startup de los dos switches utilizamos los siguientes comandos:

```

brctl addbr br0
modprobe bonding
echo balance-rr > /sys/class/net/bond0/bonding/mode
echo +eth0 > /sys/class/net/bond0/bonding/slaves
echo +eth1 > /sys/class/net/bond0/bonding/slaves
brctl addif br0 eth2
brctl addif br0 bond0
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig bond0 up
ifconfig br0 up

```

Estos activan el port bonding en modo “Balance Round Robin” utilizando los dos cables que los conectan entre si. Al pinguear a la PC2 desde la PC1, los sniffers conectados en cada uno de los cables que unían los switches mostraron que los paquetes se alternaban para enviarse, uno iba por el cable B y otro por el D.

```
sniff2
length 64
20:48:31.327718 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 7, l
length 64
20:48:33.337153 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 9,
length 64
20:48:33.337412 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 9, l
length 64
20:48:35.346917 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 11
length 64
20:48:35.347123 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 11,
length 64
20:48:37.347155 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 13
length 64
20:48:37.347377 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 13,
length 64
20:48:39.347287 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 15
length 64
20:48:39.347570 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 15,
length 64
20:48:41.357287 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 17
length 64
20:48:41.357582 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 17,
length 64
[]

sniff1
length 64
20:48:30.304907 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 6, l
length 64
20:48:32.324495 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 8,
length 64
20:48:32.324893 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 8, l
length 64
20:48:34.324521 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 10
length 64
20:48:34.325056 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 10,
length 64
20:48:36.334463 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 12
length 64
20:48:36.334680 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 12,
length 64
20:48:38.334698 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 14
length 64
20:48:38.334906 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 14,
length 64
20:48:40.334457 IP 192.168.0.1 > 192.168.0.2: ICMP echo request, id 1538, seq 16
length 64
20:48:40.334836 IP 192.168.0.2 > 192.168.0.1: ICMP echo reply, id 1538, seq 16,
length 64
[]
```

Sniffers mostrando los paquetes enviados. Se puede ver que se alternan en por la secuencia en que se envían.

Posteriormente, procedimos a apagar la interfaz eth0 de uno de los switches, que hizo que al pinguear a la otra computadora, solo el 50% de los paquetes lleguen a destino.

Luego, cambiamos el port bonding a modo "active-backup" con el comando:

```
echo active-backup > /sys/class/net/bond0/bonding/mode
```

Observando los sniffers, pudimos ver que los paquetes viajan a través de un solo cable.

Desgraciadamente, Netkit no nos permite probar el experimento deseado con este modo, tirar abajo uno de los cables, por lo que no pudimos probarlo, pero cuando pasa esto, los paquetes viajan por el otro.


```
ifconfig eth0.20 192.168.0.<Nro de PC>/24 up
```

Para id = 10

```
ifconfig eth0 up
vconfig add eth0 10
ifconfig eth0.10 192.168.0.<Nro de PC>/24 up
```

Luego, colocamos los siguientes comandos en el switch, para crear la VLAN en el mismo:

```
brctl addbr br0.10
brctl addbr br0.20
```

```
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth3 up
```

```
vconfig add eth0 20
vconfig add eth1 10
vconfig add eth2 10
vconfig add eth3 20
```

```
brctl addif br0.10 eth1.10
brctl addif br0.10 eth2.10
brctl addif br0.20 eth0.20
brctl addif br0.20 eth3.20
```

```
ifconfig eth0.20 up
ifconfig eth1.10 up
ifconfig eth2.10 up
ifconfig eth3.20 up
```

```
ifconfig br0.10 up
ifconfig br0.20 up
```

Hecho este paso, probamos pingueando de la PC1 a todas las PCs, con el resultado de que solo la Nro 4 recibe los paquetes. Los sniffers en este procedimiento registraron que los paquetes solo iban por los cables cuya PCs tenían el id de la VLAN buscada.

```

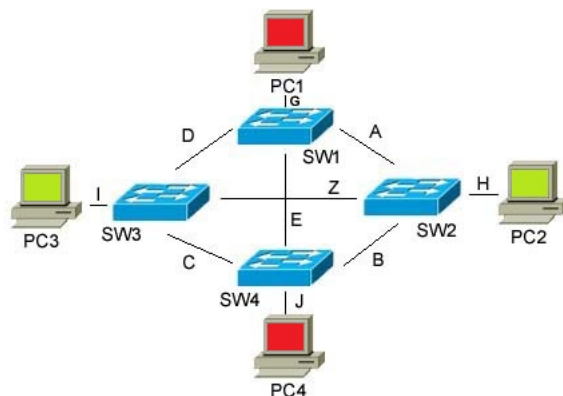
sniff
20:40:58,880903 IP sniff > sniff: ICMP sniff udp port domain unreachable, length
78
20:40:58,881048 IP sniff,55621 > sniff,domain: 6574+ PTR? 1.0,168,192,in-addr,ar
pa. (42)
20:40:58,881056 IP sniff > sniff: ICMP sniff udp port domain unreachable, length
78
20:40:59,883411 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:00,883440 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:01,893404 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:02,893476 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:03,893374 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:04,906044 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:05,913422 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:06,913412 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:07,923907 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:08,933454 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:09,923798 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:11,943692 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:12,953388 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:13,943729 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:15,963609 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:16,963599 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1
20:41:17,963697 vlan 20, p 0, arp who-has 192.168,0,2 tell 192.168,0,1

```

Ventana de uno de los sniffers.

2.5 STP-VLAN

Para este experimento utilizamos 4 PCs y dos switches, donde las PCs 1 y 4 se encontraban en una VLAN, mientras que las 2 y 3 en otra.



Colocamos 4 sniffers en los cables que unen las computadoras con los switches, y realizamos distintas pruebas pingueando desde una PC a la otra. Al igual que en el experimento anterior, los switches derivaban los paquetes solo a las PCs que se encontraban en la misma VLAN.

Analizando cómo estaba funcionando el protocolo STP en el laboratorio, nos dimos cuenta que habían designados dos switches root, que por lo general eran el SW2 y el SW4. Esto se producía debido a como estaba implementada la VLAN en los mismos.

SW1				SW3			
port id	8001	state	forwarding	port id	8002	state	forwarding
designated root	8000,06dfbbcc4366	path cost	100	designated root	8000,06dfbbcc4366	path cost	100
designated bridge	8000,06dfbbcc4366	message age timer	18,59	designated bridge	8000,06dfbbcc4366	message age timer	18,81
designated port	8001	forward delay timer	0,00	designated port	8002	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	0	hold timer	0,00
Flags				Flags			
eth1,10 (2)				eth2,10 (3)			
port id	8002	state	blocking	port id	8003	state	blocking
designated root	8000,06dfbbcc4366	path cost	100	designated root	8000,06dfbbcc4366	path cost	100
designated bridge	8000,1e3cec97ee62	message age timer	18,58	designated bridge	8000,1e3cec97ee62	message age timer	18,80
designated port	8002	forward delay timer	0,00	designated port	8003	forward delay timer	0,00
designated cost	100	hold timer	0,00	designated cost	100	hold timer	0,00
Flags				Flags			
eth2,10 (3)				eth3,10 (4)			
port id	8003	state	forwarding	port id	8004	state	forwarding
designated root	8000,06dfbbcc4366	path cost	100	designated root	8000,06dfbbcc4366	path cost	100
designated bridge	8000,3a3700de0b66	message age timer	0,00	designated bridge	8000,5eed01fe1f3d	message age timer	0,00
designated port	8003	forward delay timer	0,00	designated port	8004	forward delay timer	0,00
designated cost	100	hold timer	0,34	designated cost	100	hold timer	0,41
Flags				Flags			
SW1:"# "				SW3:"# "			
SW4				SW2			
port id	8001	state	forwarding	port id	8002	state	forwarding
designated root	8000,06dfbbcc4366	path cost	100	designated root	8000,06dfbbcc4366	path cost	100
designated bridge	8000,06dfbbcc4366	message age timer	18,22	designated bridge	8000,06dfbbcc4366	message age timer	0,00
designated port	8003	forward delay timer	0,00	designated port	8002	forward delay timer	0,00
designated cost	0	hold timer	0,00	designated cost	0	hold timer	0,60
Flags				Flags			
eth1,10 (2)				eth2,10 (3)			
port id	8002	state	forwarding	port id	8003	state	forwarding
designated root	8000,06dfbbcc4366	path cost	100	designated root	8000,06dfbbcc4366	path cost	100
designated bridge	8000,1e3cec97ee62	message age timer	0,00	designated bridge	8000,06dfbbcc4366	message age timer	0,00
designated port	8002	forward delay timer	0,00	designated port	8003	forward delay timer	0,00
designated cost	100	hold timer	0,00	designated cost	0	hold timer	0,60
Flags				Flags			
eth2,10 (3)				eth3,10 (4)			
port id	8003	state	forwarding	port id	8004	state	forwarding
designated root	8000,06dfbbcc4366	path cost	100	designated root	8000,06dfbbcc4366	path cost	100
designated bridge	8000,1e3cec97ee62	message age timer	0,00	designated bridge	8000,06dfbbcc4366	message age timer	0,00
designated port	8003	forward delay timer	0,00	designated port	8004	forward delay timer	0,00
designated cost	100	hold timer	0,00	designated cost	0	hold timer	0,60
Flags				Flags			
SW4:"# "				SW2:"# "			

Interfaces bloqueadas para las que poseían ID de VLAN nro 20

SW1				SW3			
port id	8003	state	forwarding	port id	8001	state	blocking
designated root	8000.1e3cec97ee62	path cost	100	designated root	8000.1e3cec97ee62	path cost	100
designated bridge	8000.1e3cec97ee62	message age timer	19.43	designated bridge	8000.3a3700de0b66	message age timer	19.04
designated port	8003	forward delay timer	0.00	designated port	8004	forward delay timer	0.00
designated cost	0	hold timer	0.00	designated cost	100	hold timer	0.00
flags				flags			
eth2,20 (4)				eth1,20 (2)			
port id	8004	state	forwarding	port id	8002	state	blocking
designated root	8000.1e3cec97ee62	path cost	100	designated root	8000.1e3cec97ee62	path cost	100
designated bridge	8000.3a3700de0b66	message age timer	0.00	designated bridge	8000.3eb28eff8d6d	message age timer	19.04
designated port	8004	forward delay timer	0.00	designated port	8002	forward delay timer	0.00
designated cost	100	hold timer	0.96	designated cost	100	hold timer	0.00
flags				flags			
eth3,20 (1)				eth2,20 (3)			
port id	8001	state	forwarding	port id	8003	state	forwarding
designated root	8000.1e3cec97ee62	path cost	100	designated root	8000.1e3cec97ee62	path cost	100
designated bridge	8000.3a3700de0b66	message age timer	0.00	designated bridge	8000.1e3cec97ee62	message age timer	19.05
designated port	8001	forward delay timer	0.00	designated port	8004	forward delay timer	0.00
designated cost	100	hold timer	0.96	designated cost	0	hold timer	0.00
flags				flags			
SW1:"#				SW3:"#			
SW4				SW2			
port id	8003	state	forwarding	port id	8001	state	blocking
designated root	8000.1e3cec97ee62	path cost	100	designated root	8000.1e3cec97ee62	path cost	100
designated bridge	8000.1e3cec97ee62	message age timer	0.00	designated bridge	8000.3a3700de0b66	message age timer	18.22
designated port	8003	forward delay timer	0.00	designated port	8002	forward delay timer	0.00
designated cost	0	hold timer	0.15	designated cost	100	hold timer	0.00
flags				flags			
eth2,20 (4)				eth1,20 (2)			
port id	8004	state	forwarding	port id	8002	state	forwarding
designated root	8000.1e3cec97ee62	path cost	100	designated root	8000.1e3cec97ee62	path cost	100
designated bridge	8000.1e3cec97ee62	message age timer	0.00	designated bridge	8000.3eb28eff8d6d	message age timer	0.00
designated port	8004	forward delay timer	0.00	designated port	8002	forward delay timer	0.00
designated cost	0	hold timer	0.15	designated cost	100	hold timer	0.00
flags				flags			
eth3,20 (1)				eth2,20 (3)			
port id	8001	state	forwarding	port id	8003	state	forwarding
designated root	8000.1e3cec97ee62	path cost	100	designated root	8000.1e3cec97ee62	path cost	100
designated bridge	8000.1e3cec97ee62	message age timer	0.00	designated bridge	8000.1e3cec97ee62	message age timer	18.23
designated port	8001	forward delay timer	0.00	designated port	8002	forward delay timer	0.00
designated cost	0	hold timer	0.15	designated cost	0	hold timer	0.00
flags				flags			
SW4:"#				SW2:"#			

Interfaces bloqueadas para las que poseían ID de VLAN nro 10

Al igual que en el primer experimento, al cambiar el switch root, cambiaba las interfaces bloqueadas, aunque la VLAN seguía funcionando correctamente en todo momento.

Fuentes:

- <https://camber1redes.wordpress.com/puente-de-red-o-bridge/>
- <https://www.cloudibee.com/network-bonding-modes/>
- <http://redestelematicas.com/el-switch-como-funciona-y-sus-principales-caracteristicas/>
- <https://backdrift.org/manage-linux-bonding-without-ifenslave-using-sysfs>
- <https://ahelpme.com/linux/how-to-enable-linux-bonding-without-ifenslave/>
- <https://rm-rf.es/configurar-una-vlan-en-linux-con-vconfig/>