

Máster en Ingeniería Informática

Redes de Nueva Generación

Profesor:

Juan Carlos Fabero Jiménez (UCM)



Contenidos

- **Tema 1: IP de nueva generación: IPv6**
- **Tema 2: Encaminamiento interno: OSPF**
- **Tema 3: Encaminamiento externo: BGPv4**
- **Tema 4: MultiProtocol Label Switching**
- **Tema 5: Redes Definidas por Software (SDN)**

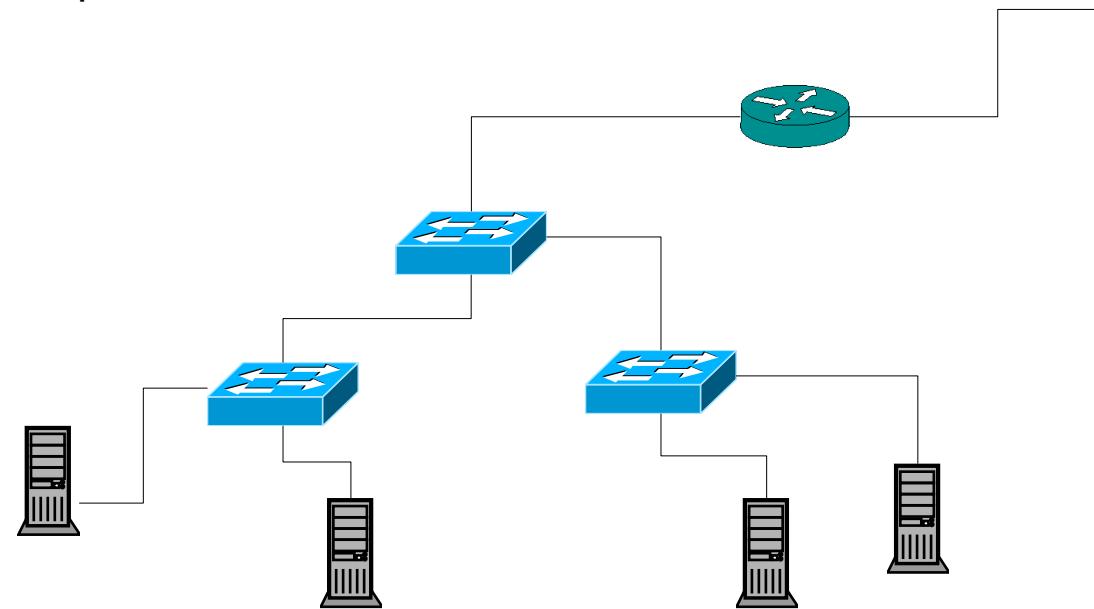
LAN Virtuales (VLAN)

Introducción

Introducción

■ Organización de la red de área local

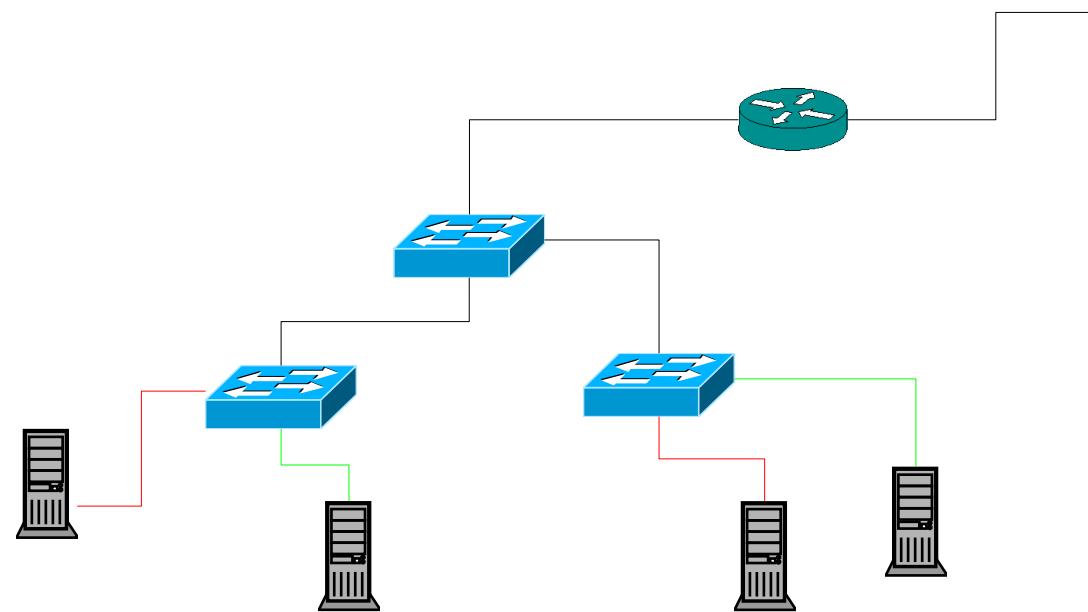
- Redes segmentadas (comutadores/switches)
 - Cada departamento / Grupo de trabajo en una LAN.
 - Las LAN se comunican entre sí de manera “jerárquica”.
 - Pero...
 - Sigue siendo un solo dominio de difusión.
 - No hay restricciones de tráfico.
 - Puede haber puertos no utilizados.



Introducción

■ Solución

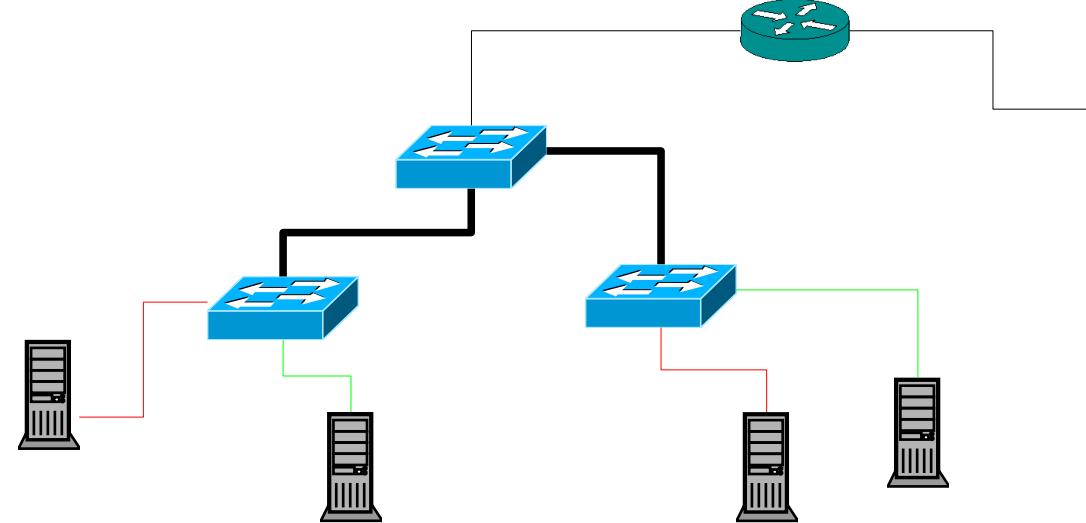
- Redes de área local virtuales (VLAN)
 - Las máquinas de un grupo pertenecen a la misma VLAN.
 - El conmutador separa el tráfico perteneciente a cada VLAN.
 - Un equipo puede pertenecer a varias VLAN.
 - La configuración puede cambiar de manera dinámica.



Introducción

■ Alternativas

- VLAN basadas en puertos
 - El administrador decide qué puerto de cada conmutador pertenece a cada VLAN.
 - Dificultad: comunicación entre varios conmutadores.
- VLAN basadas en etiquetas
 - A cada trama se le añade una etiqueta (*tag*) que indica a qué VLAN pertenece (802.1Q)
 - Es posible unir varios conmutadores entre sí (*trunking*)



VLAN

802.1Q

Formato

■ Una etiqueta 802.1Q tiene 32 bits:

- Campo de Tipo (16 bits): 0x8100
- Campo de Etiqueta (TCI, *Tag Control Information*):
 - Prioridad (4 bits)
 - VID (*VLAN Identifier*) (12 bits, 0..4095)



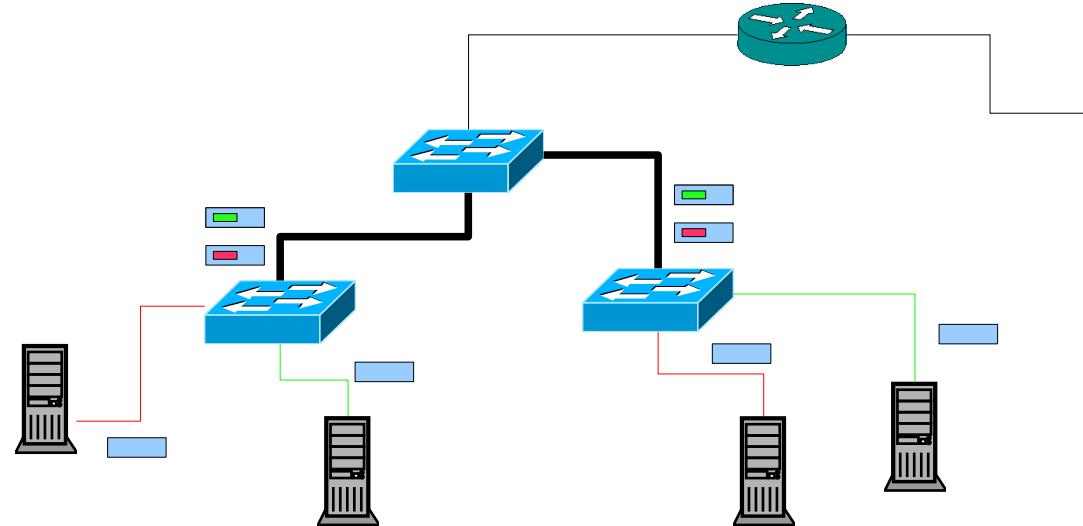
Funcionamiento

■ Comutadores:

- Gestionables.
- Implementan 802.1Q
- El primer comutador añade la etiqueta, el último la elimina.
- Las etiquetas no cambian a lo largo del recorrido.

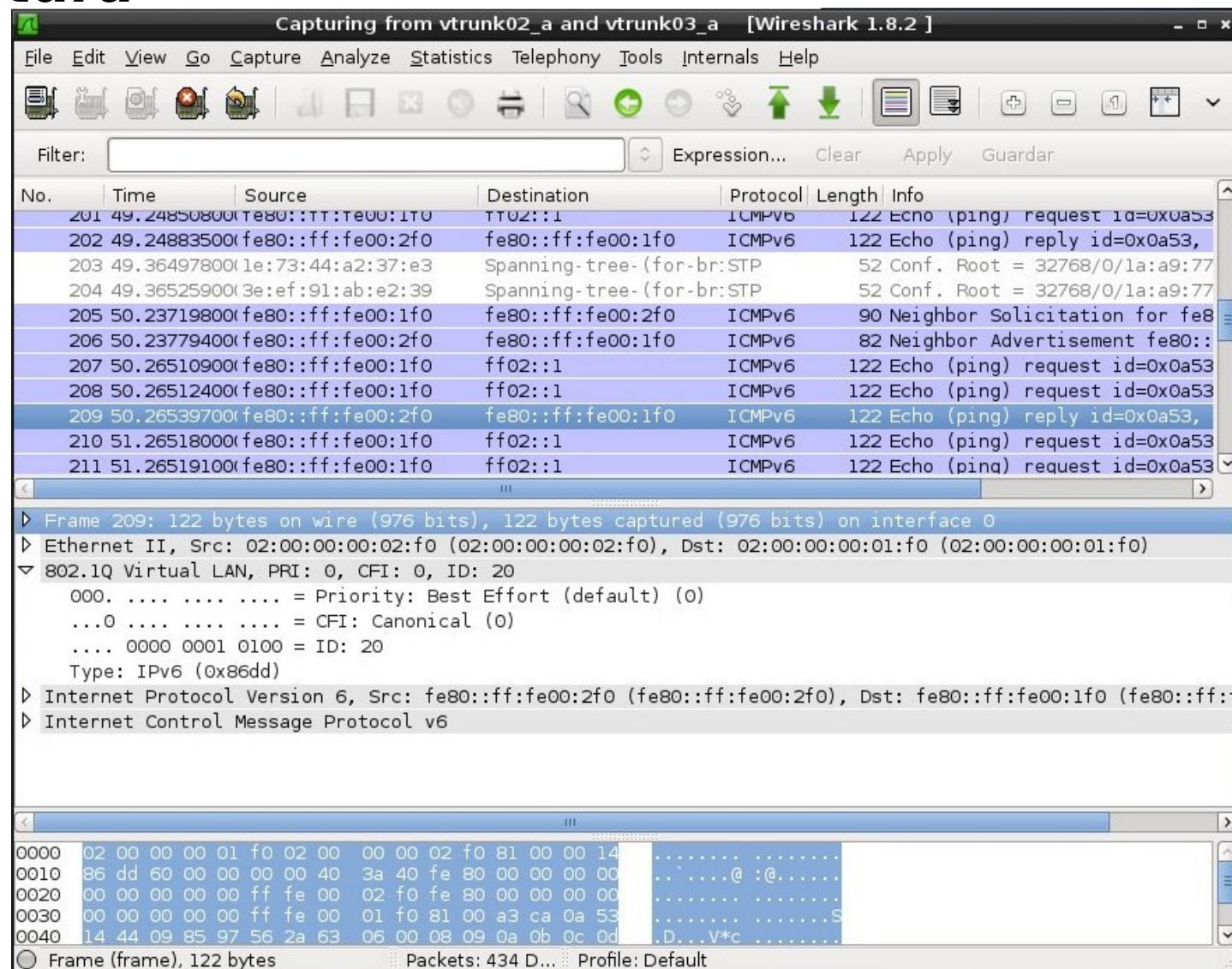
■ Estaciones:

- Pueden implementar o no 802.1Q



Ejemplo

■ Captura

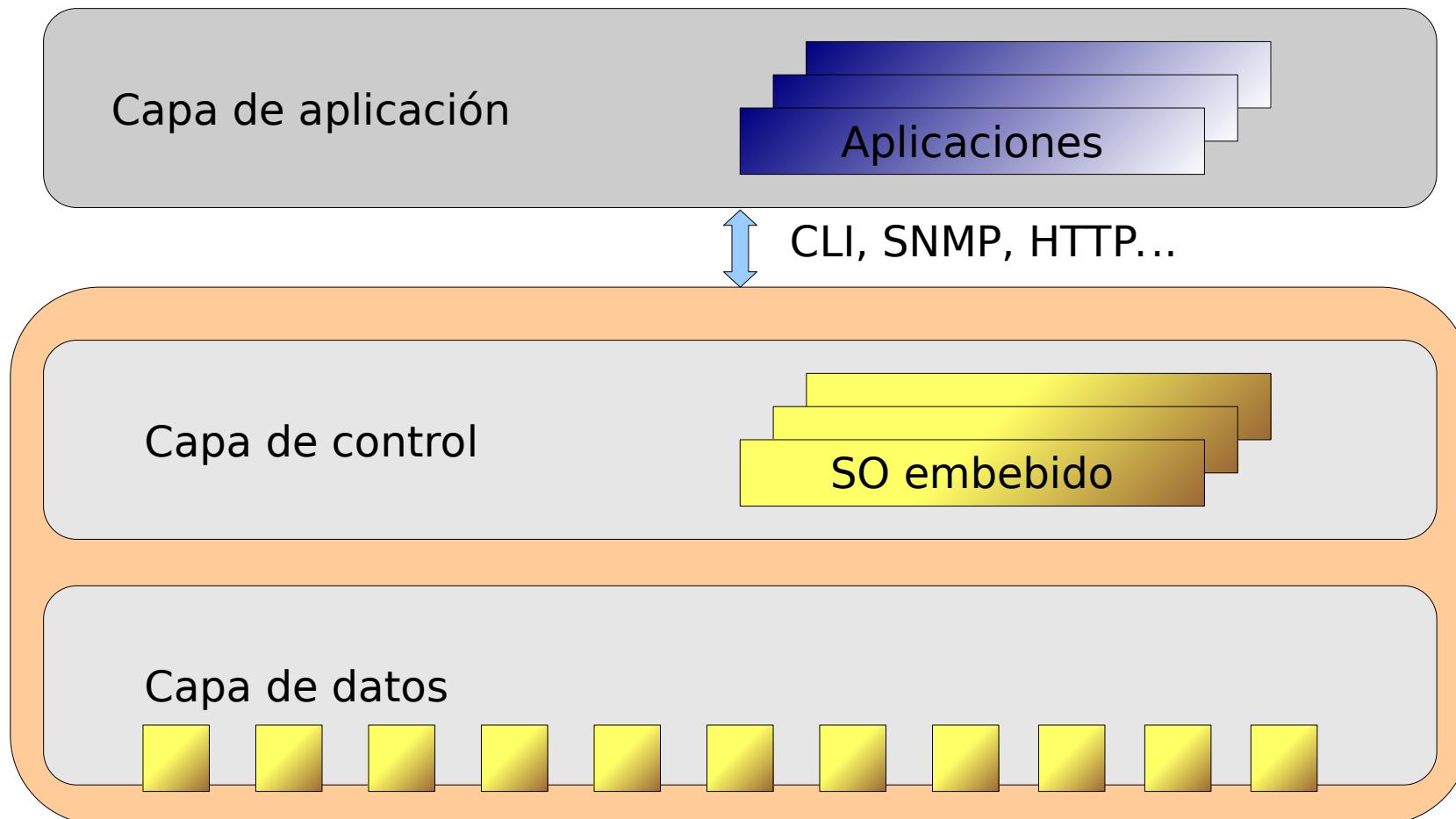


Redes Definidas por Software (SDN)

Software Defined Networks

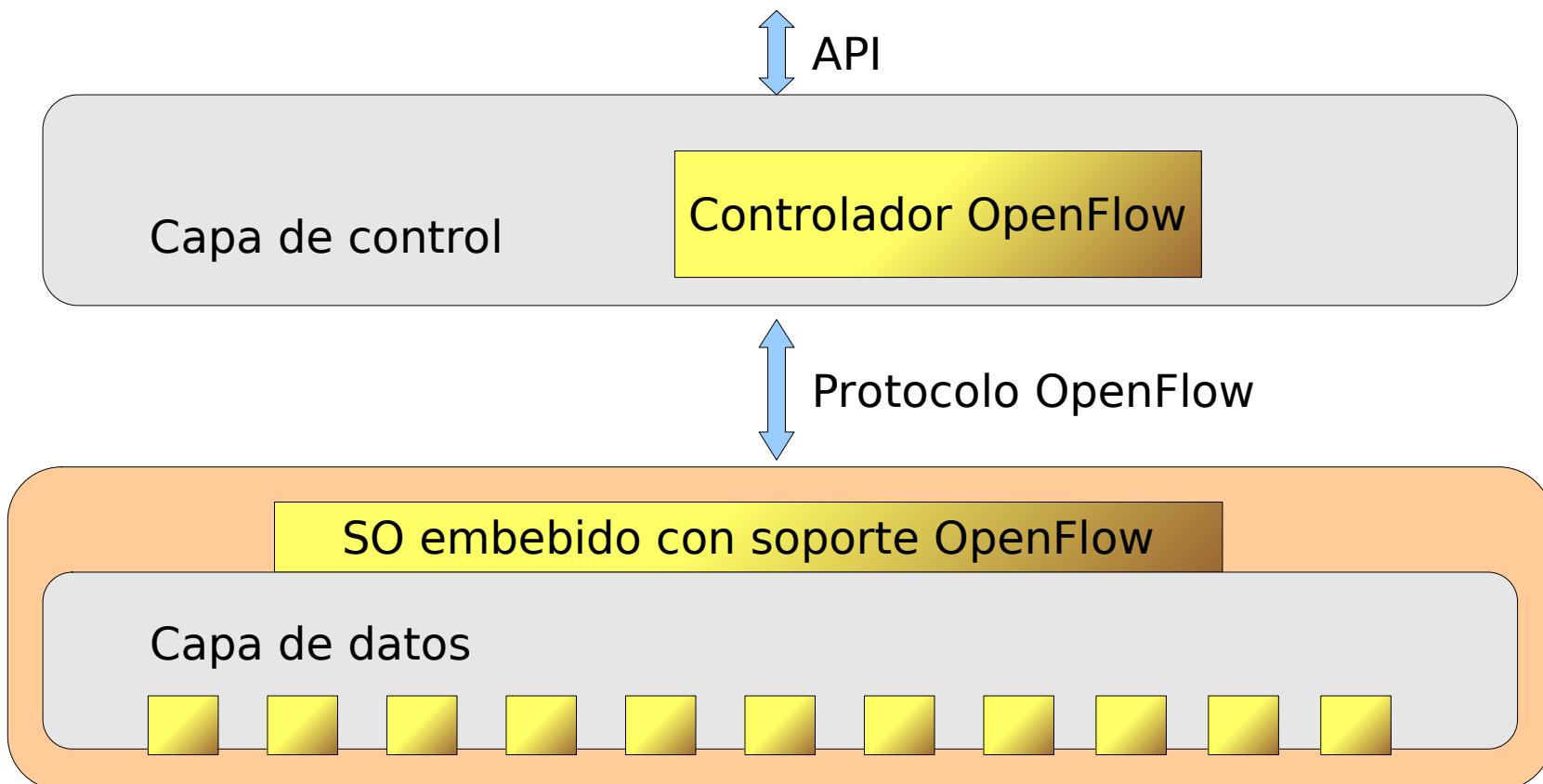
■ Switch Ethernet gestionable

- Se utilizan diversas API (muchas, propietarias), para configurar el funcionamiento del *switch*.



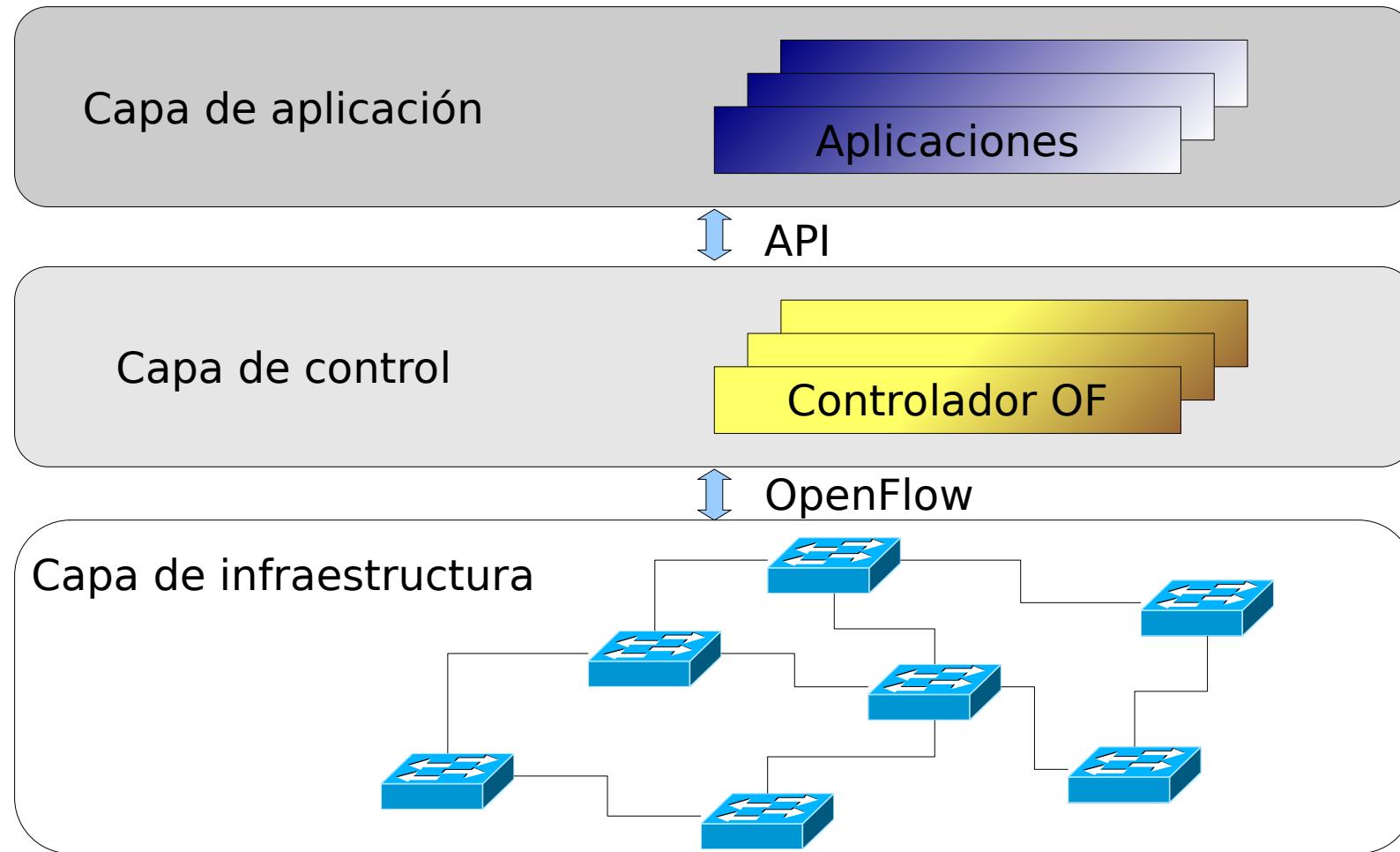
■ Software Defined Networks

- Consisten en separar la capa de control de la capa de datos (encaminamiento), para hacer gestionable esta última de manera versátil y estándar.



■ Software Defined Networks

- Un solo controlador puede gestionar múltiples dispositivos.



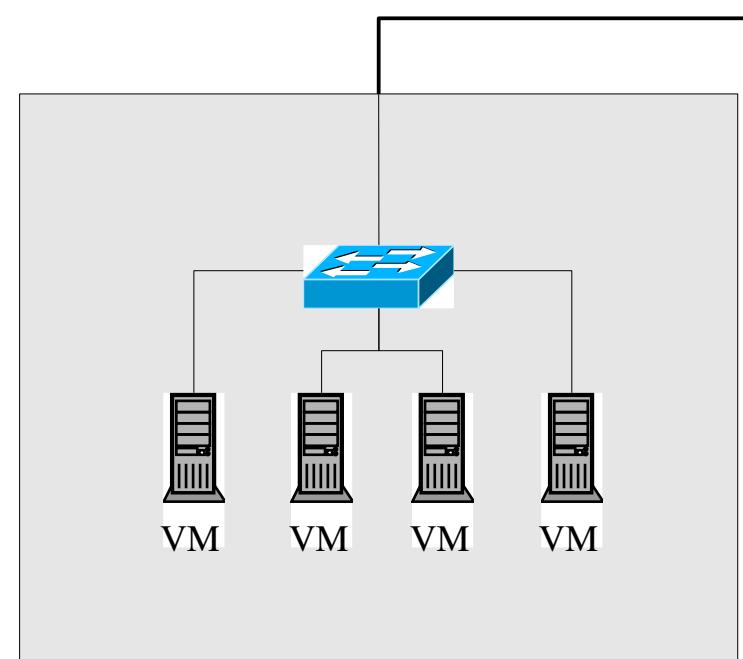
SDN

Open VSwitch

Open vSwitch

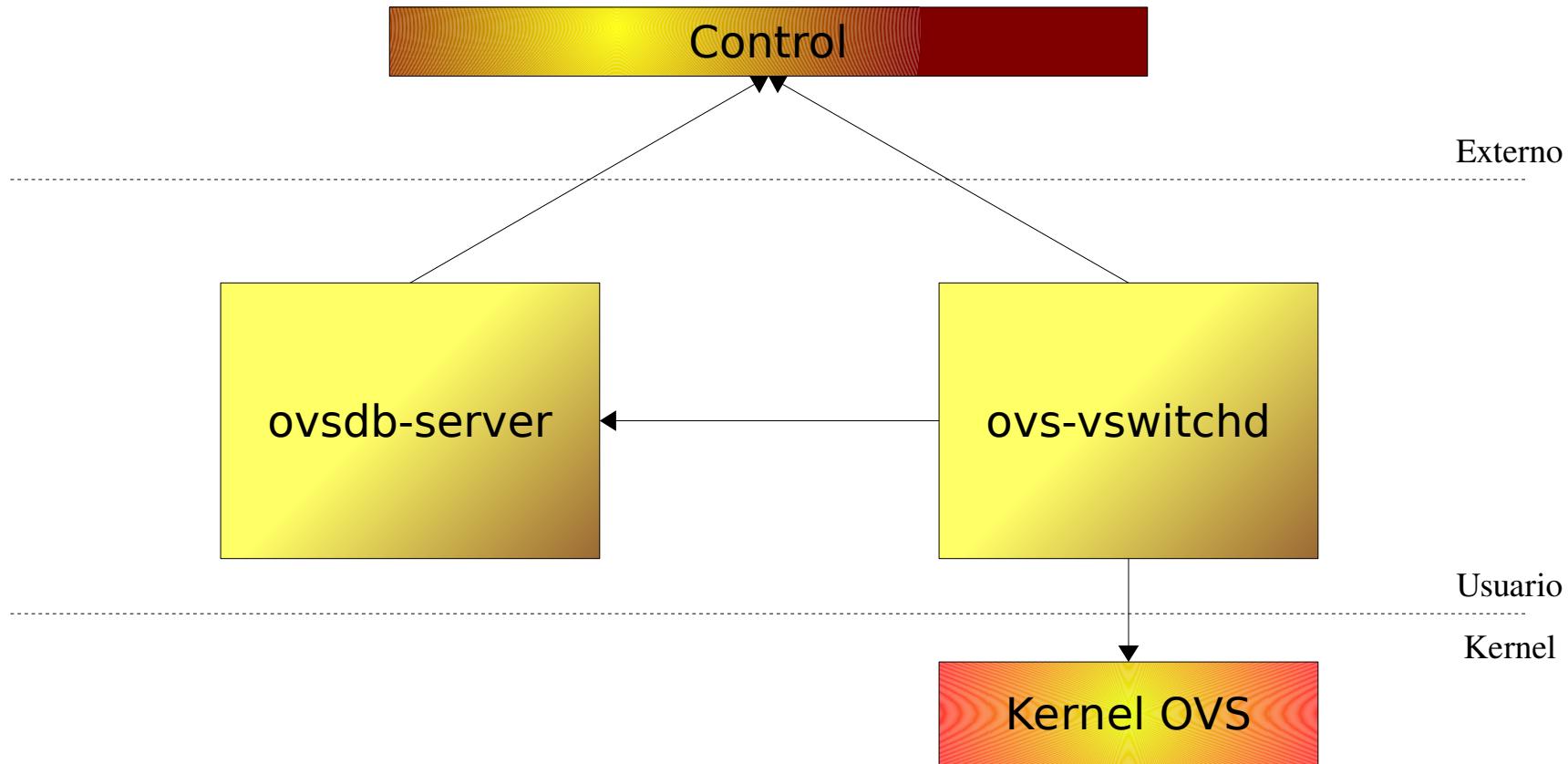
■ Introducción

- Software abierto que implementa un *switch* virtual en entornos de servidores virtualizados.
- Reenvía tráfico entre VM que se ejecutan en la misma máquina física.
 - También controla el acceso de las VM a la red física.
- Puede permitir el manejo de *switches* físicos con la misma arquitectura.
- Documentación:
 - RFC7047 (informativo).
 - <http://openvswitch.org/>



Open vSwitch

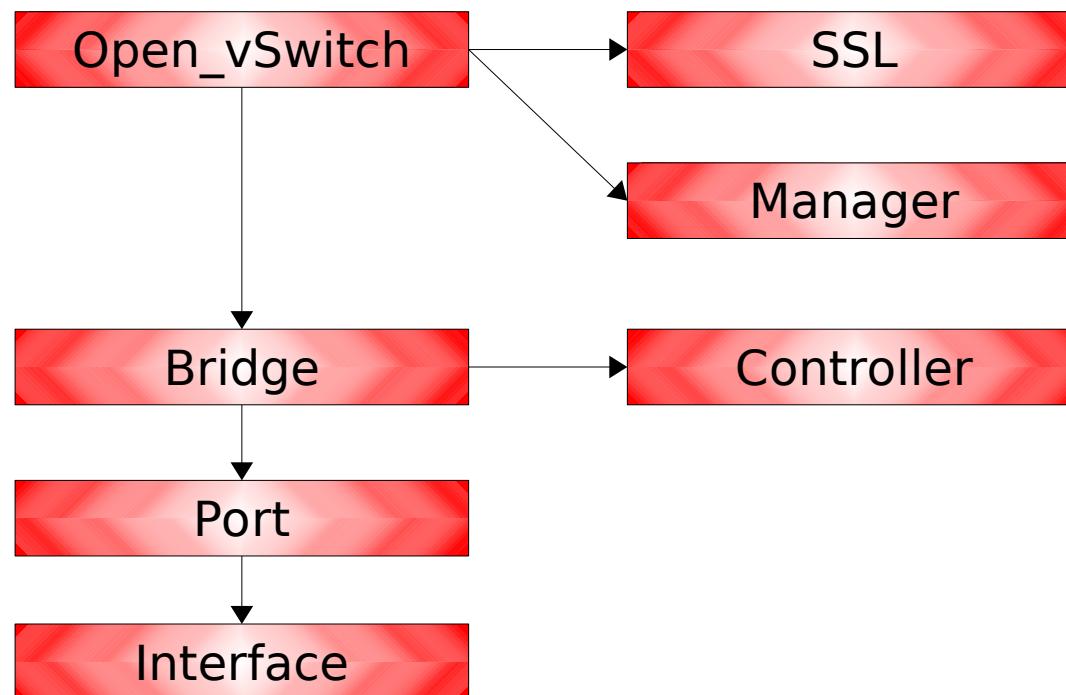
■ Arquitectura



Open vSwitch

■ ovSdb-server

- Mantiene la base de datos de Open vSwitch.
 - Puentes, interfaces, túneles
 - Direcciones de los controladores
- Configuración en memoria no volátil.



Open vSwitch

■ ovs-vswitchd

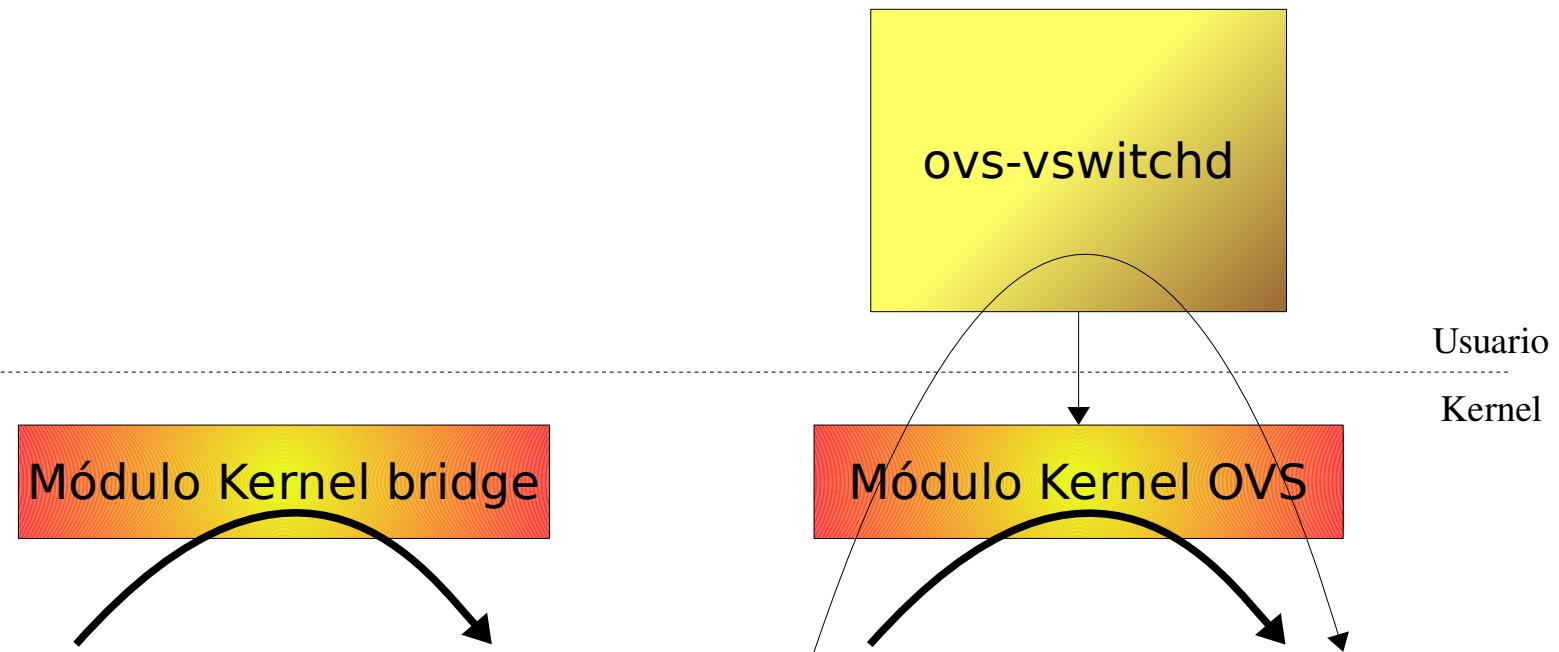
- Es el gestor de Open vSwitch
 - Mantiene actualizada la información de la base de datos.
 - Configura el módulo OVS del kernel.
 - Gestiona los interfaces virtuales.
- Puede soportar múltiples *bridges* independientes (datapaths).
- Permite clasificación de paquetes.
- Soporta definiciones de QoS.
- Admite *mirroring*, *trunking* y *VLAN*.
- A partir de la versión 1.9, también admite *patching*.



Open vSwitch

■ Implementación

- Con el módulo bridge de Linux, todo el tratamiento se hace en espacio de kernel.
- Con Open vSwitch, el primer paquete del flujo se trata en espacio de usuario por parte del ovs-vswitchd, que instruye al módulo OVS del kernel para que gestione los paquetes siguientes.



Open vSwitch

■ Utilidades

- **ovs-vsctl**: configura ovs-vswitchd.

- La configuración se almacena en la base de datos.

- Crear un puente (bridge):

```
ovs-vsctl add-br <bridge>
```

- Añadir un puerto a un bridge:

```
ovs-vsctl add-port <bridge> <port>
```

- Listar bridges:

```
ovs-vsctl list-br
```

- Listar puertos de un bridge:

```
ovs-vsctl list-ports <bridge>
```

- Listar información de un puerto (o de todos):

```
ovs-vsctl list port [<puerto>]
```

- Mostrar información de un interfaz (o todos):

```
ovs-vsctl list interface [<interface>]
```

- Listar información de la base de datos:

```
ovs-vsctl list <table>
```

■ Ejemplos

- Crear un bridge:

```
# ovs-vsctl add-br br0
```

- Añadir un puerto a un bridge:

```
# ovs-vsctl add-port br0 eth0
```

- O bien:

```
# ovs-vsctl add-br br0 -- add-port br0 eth0
```

- Eliminar un puerto de un bridge:

```
# ovs-vsctl del-port br0 eth0
```

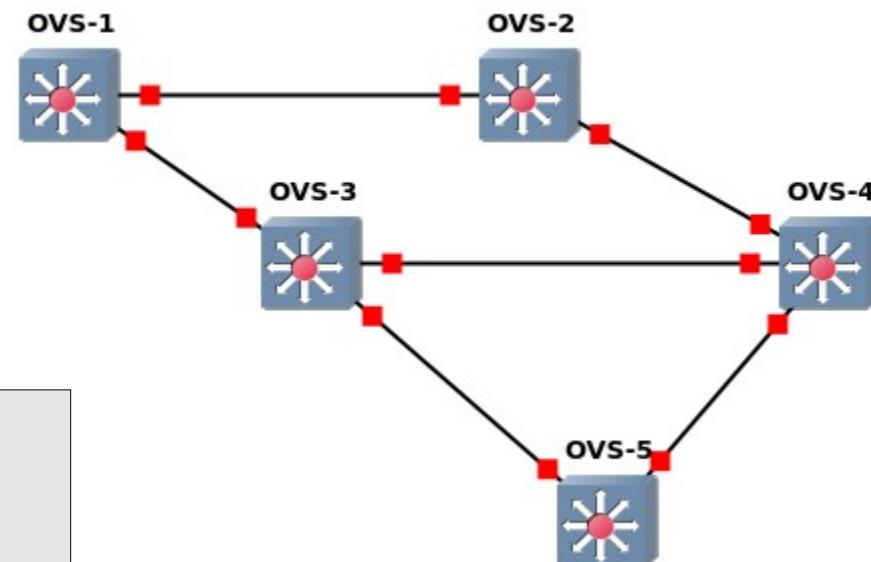
- Borrar un bridge:

```
# ovs-vsctl del-br br0
```

Patching con Open vSwitch

■ STP (Spanning Tree Protocol, 802.1D):

- Cuando se construyen topologías complejas, pueden aparecer bucles.
- Para evitar problemas de saturación (p.e. multicast storm), el protocolo STP “poda” algunas ramas para convertir el grafo en un árbol de recubrimiento mínimo.
- Se rompen los bucles, pero se mantienen los enlaces redundantes en caso de fallo.



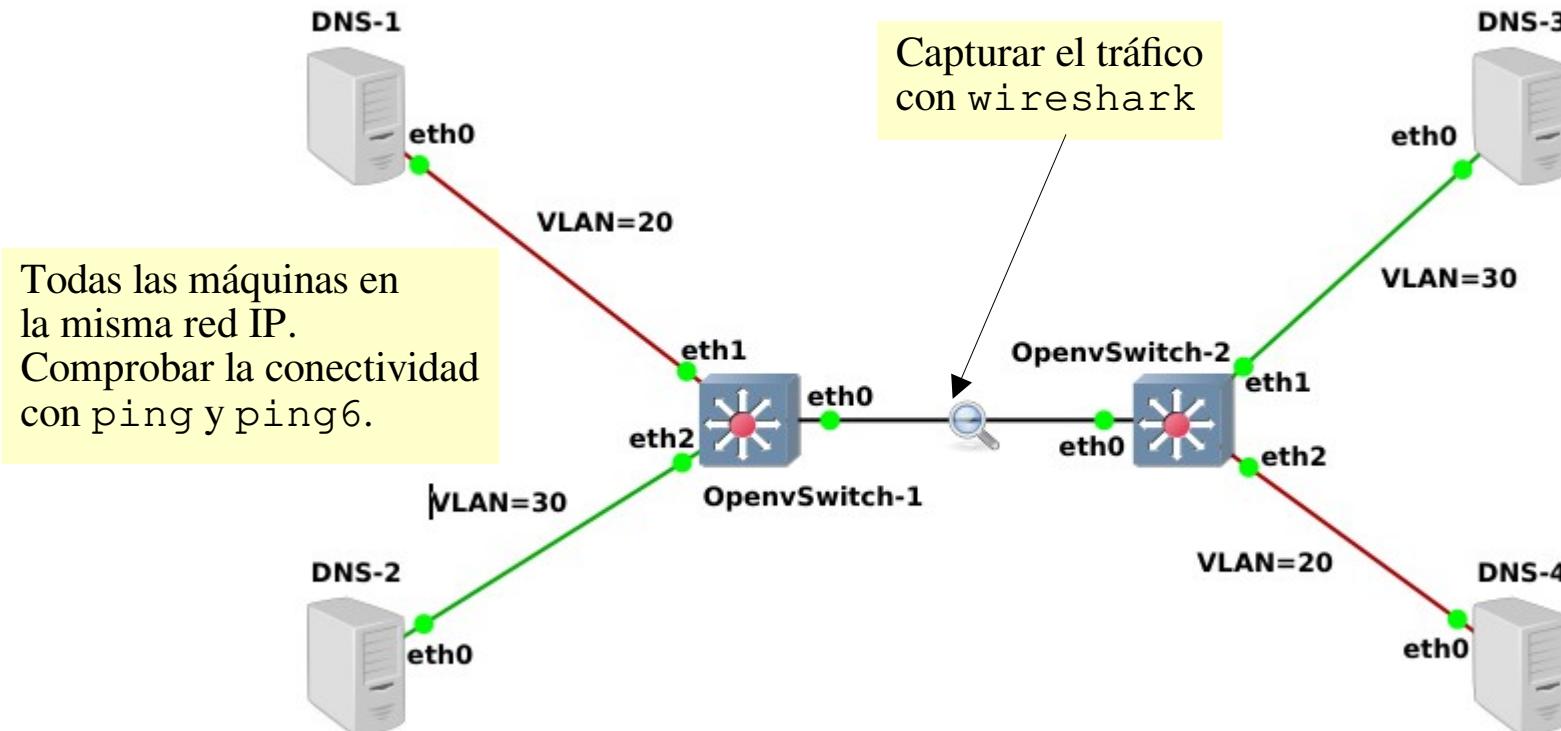
```
# crear el bridge:  
ovs-vsctl add-br br0  
# activar STP:  
ovs-vsctl set bridge br0 stp_enable=true
```

SDN

802.1Q

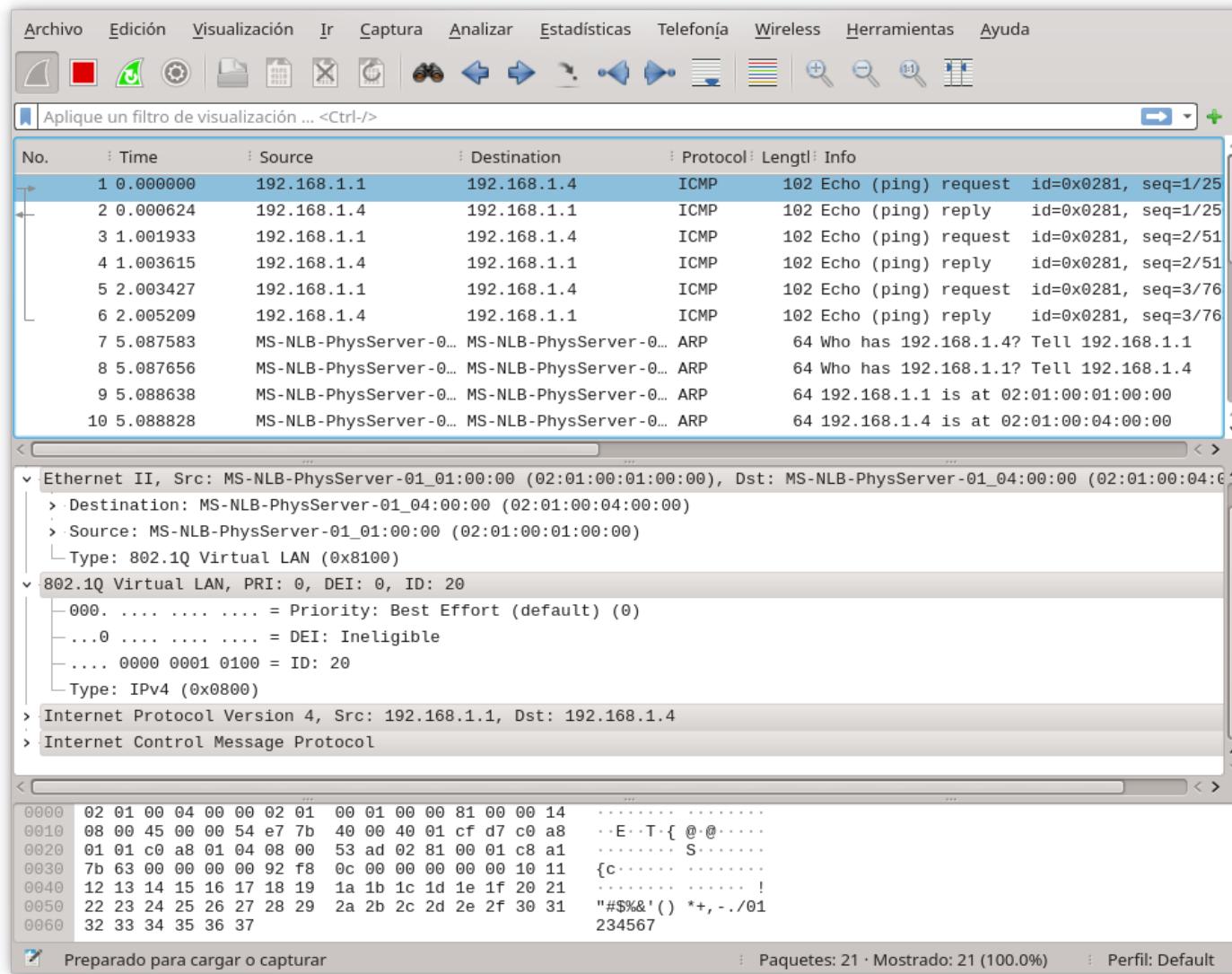
VLAN con Open vSwitch

■ Open vSwitch permite Utilizar 802.1Q



Ejemplo

Captura



SDN

OpenFlow

■ ovs-ofctl

- La verdadera utilidad de Open vSwitch es la posibilidad de controlar, de manera dinámica, los flujos de datos que atraviesan el *switch*.
- Un flujo es un conjunto de paquetes que cumplen determinado criterio. Los criterios se definen mediante “reglas”.
- Los distintos flujos se agrupan en tablas (0 a 253) y además se ordenan por prioridad.
- Cada regla tiene una sección “*match*” y una sección “*actions*”.
 - *match*: in_port, dl_src, dl_dst, dl_vlan, dl_type, ip, arp, icmp, tcp, udp, ipv6, icmp6, tcp6, udp6...
 - *actions*: normal, flood, all, drop, resubmit, <puerto>, mod_vlan_vid, strip_vlan, mod_dl_src, mod_dl_dst, mod_nw_src, goto_table...

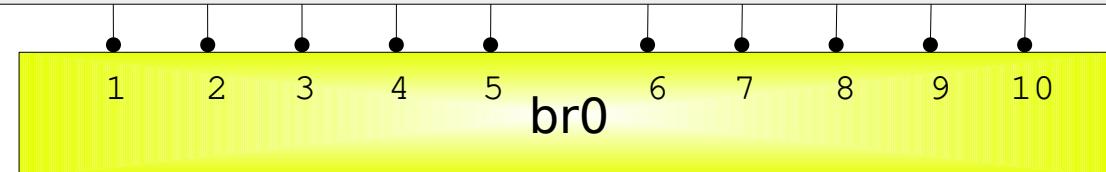
Flujos

■ ovs-ofctl

- Ejemplo. Obtener número de puerto OpenFlow (útil para definir los flujos):

```
ovs-vsctl add-port br0 eth0 -- set Interface eth0 ofport_request=1  
ovs-vsctl get Interface eth0 ofport  
1
```

```
ovs-ofctl show br0  
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000d601753ebd4f  
n_tables:254, n_buffers:256  
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS  
ARP_MATCH_IP  
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src  
mod_dl_dst mod_nw_src mod_nw_dst mod_nw_tos mod_tp_src mod_tp_dst  
1(eth0): addr:be:a8:1f:31:bd:b3  
    config: 0  
    state: 0  
    current: 10MB-FD COPPER  
    speed: 10 Mbps now, 0 Mbps max  
2(eth1): addr:9e:b7:fd:c0:b7:1c  
    config: 0  
    state: 0  
    current: 10MB-FD COPPER  
    speed: 10 Mbps now, 0 Mbps max
```

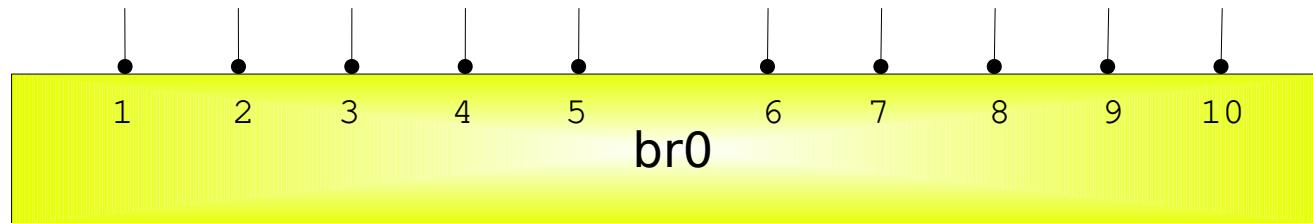


Flujos

■ ovs-ofctl

- Ejemplo. Mostrar los flujos definidos actualmente.

```
ovs-ofctl dump-flows br0
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=82.214s,
    table=0, n_packets=0, n_bytes=0, idle_age=82,
    priority=0 actions=NORMAL
cookie=0x0, duration=5277.696s,
    table=0, n_packets=9649, n_bytes=945602, idle_age=0,
    priority=50, in_port=1 actions=resubmit(,101)
```

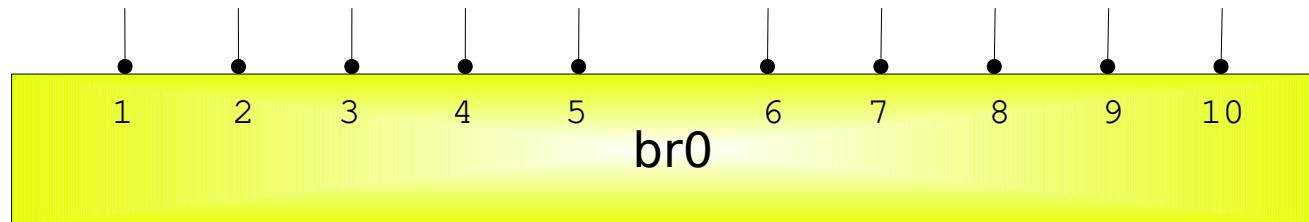


Flujos

■ ovs-ofctl

- Ejemplo. Añadir flujos.

```
ovs-ofctl add-flow br0 \
    "table=0, priority=210 \
     ipv6_src=fe80::ff:fe00:1f0/128 icmp6 icmp_type=134 actions=normal"
ovs-ofctl add-flow br0 \
    "table=0, priority=200 \
     icmp6 icmp_type=134 actions=drop"
ovs-ofctl add-flow br0 \
    "table=0, priority=100 dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 \
     actions=flood"
```

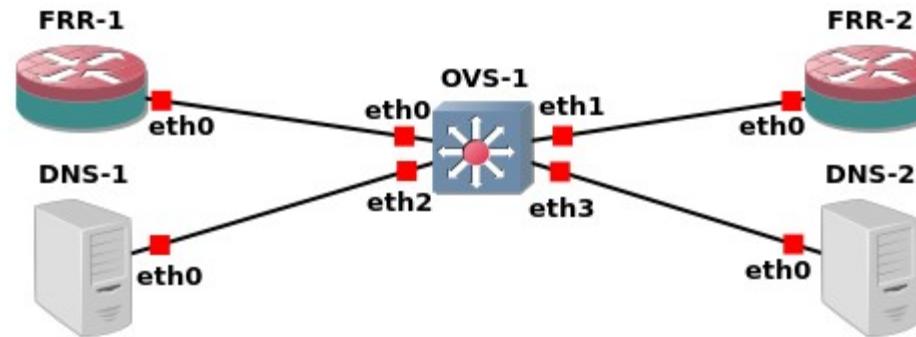


Flujos

■ ovs-ofctl

■ Práctica.

- La máquina FRR-1 realiza un anuncio de prefijos, y es la única autorizada.
- La máquina FRR-2 también intentará hacerlo, pero OpenFlow lo impedirá.

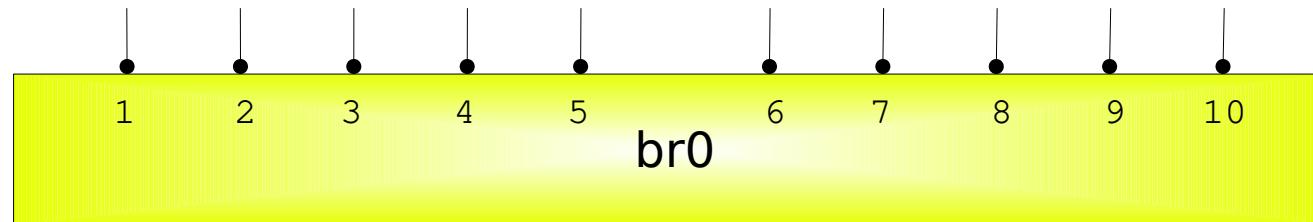


Flujos

■ ovs-ofctl

- Ejemplo. Borrar flujos.

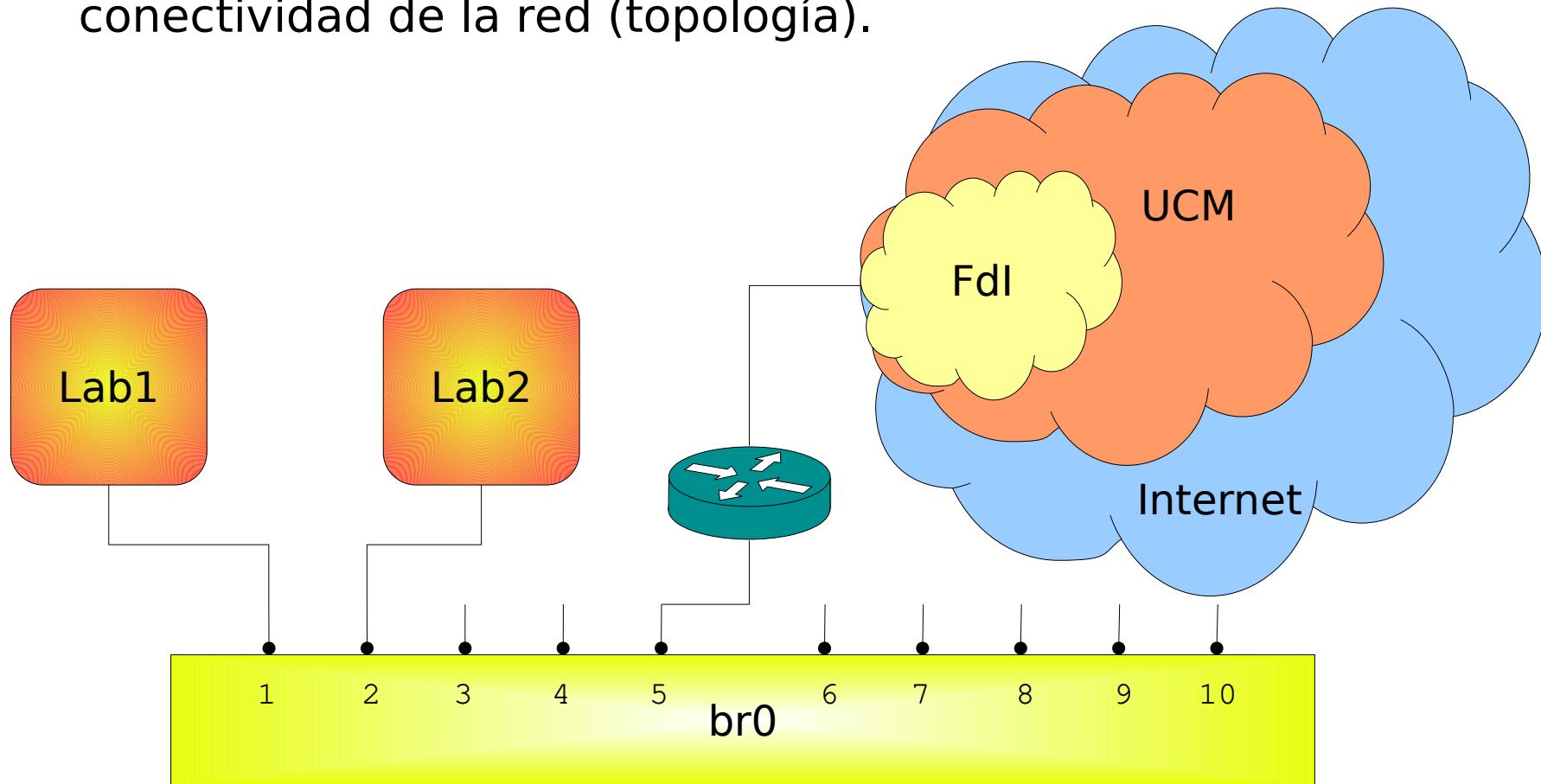
```
ovs-ofctl del-flows br0 "ipv6_src=fe80::ff:fe00:1f0/128 icmp6 icmp_type=134"  
ovs-ofctl del-flows br0
```



Flujos

■ ovs-ofctl

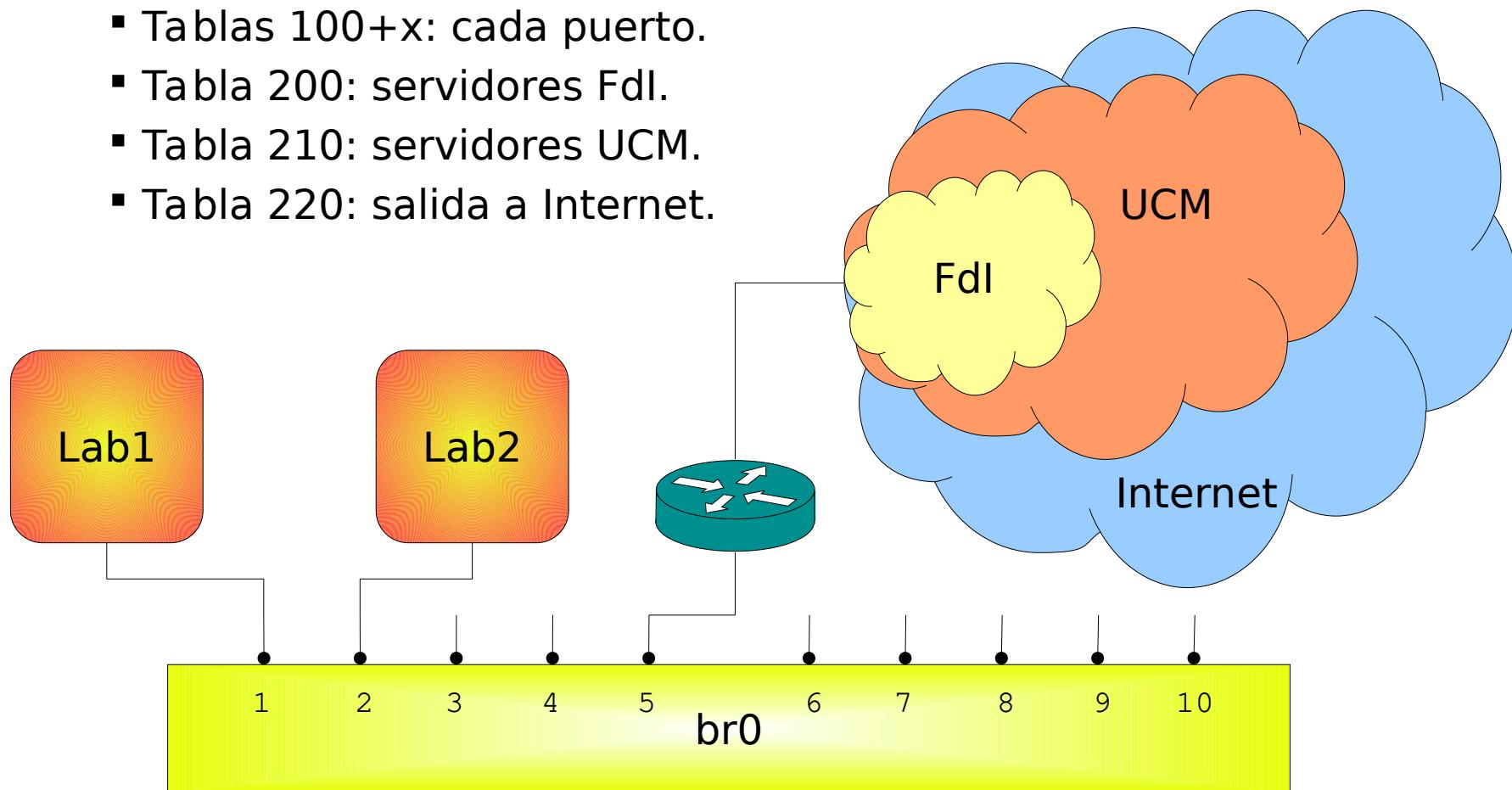
- Es muy útil para controlar de manera dinámica la conectividad de la red (topología).



Flujos

■ ovs-ofctl

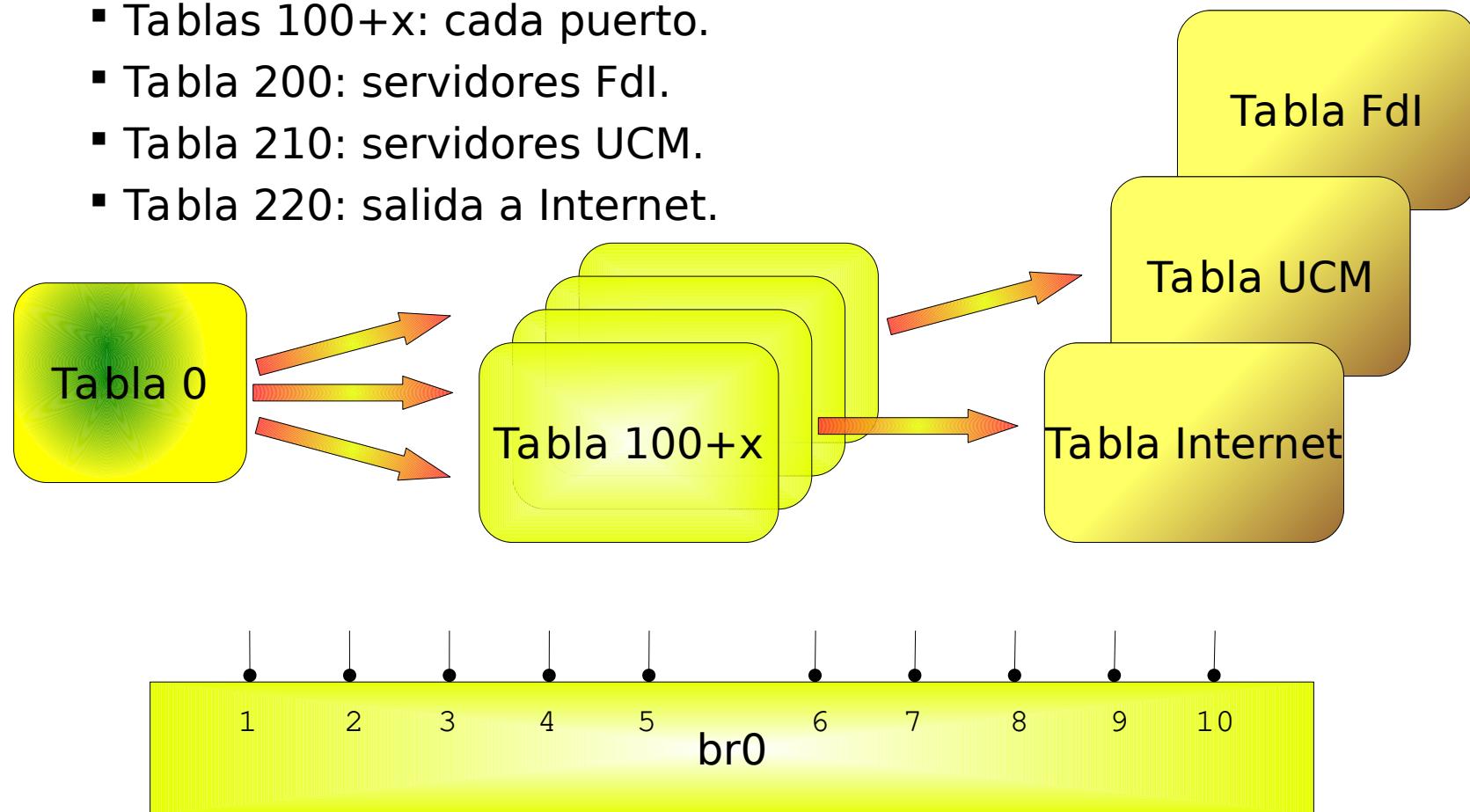
- Las tablas permiten organizar de manera eficiente los flujos.
 - Tabla 0: clasificador.
 - Tablas 100+x: cada puerto.
 - Tabla 200: servidores Fdl.
 - Tabla 210: servidores UCM.
 - Tabla 220: salida a Internet.



Flujos

■ ovs-ofctl

- Las tablas permiten organizar de manera eficiente los flujos.
 - Tabla 0: clasificador.
 - Tablas 100+x: cada puerto.
 - Tabla 200: servidores Fdl.
 - Tabla 210: servidores UCM.
 - Tabla 220: salida a Internet.

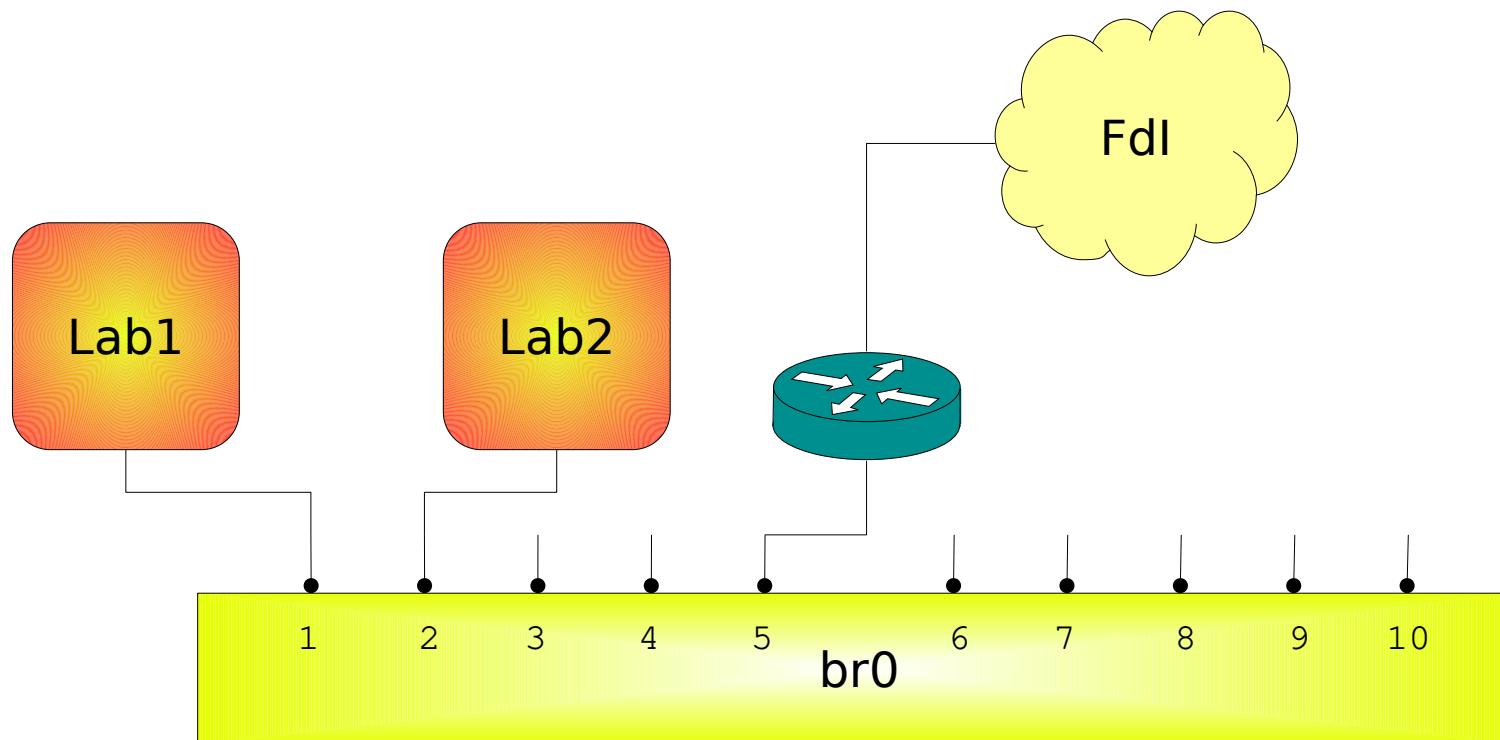


Flujos

■ ovs-ofctl

- Ejemplo. Clasificar los puertos.

```
ovs-ofctl del-flows br0
ovs-ofctl add-flow br0 "table=0 priority=1000 arp actions=normal"
ovs-ofctl add-flow br0 "table=0 priority=100 in_port=1 actions=resubmit(,101)"
ovs-ofctl add-flow br0 "table=0 priority=100 in_port=2 actions=resubmit(,102)"
ovs-ofctl add-flow br0 "table=0 priority=100 in_port=3 actions=resubmit(,103)"
...
...
```

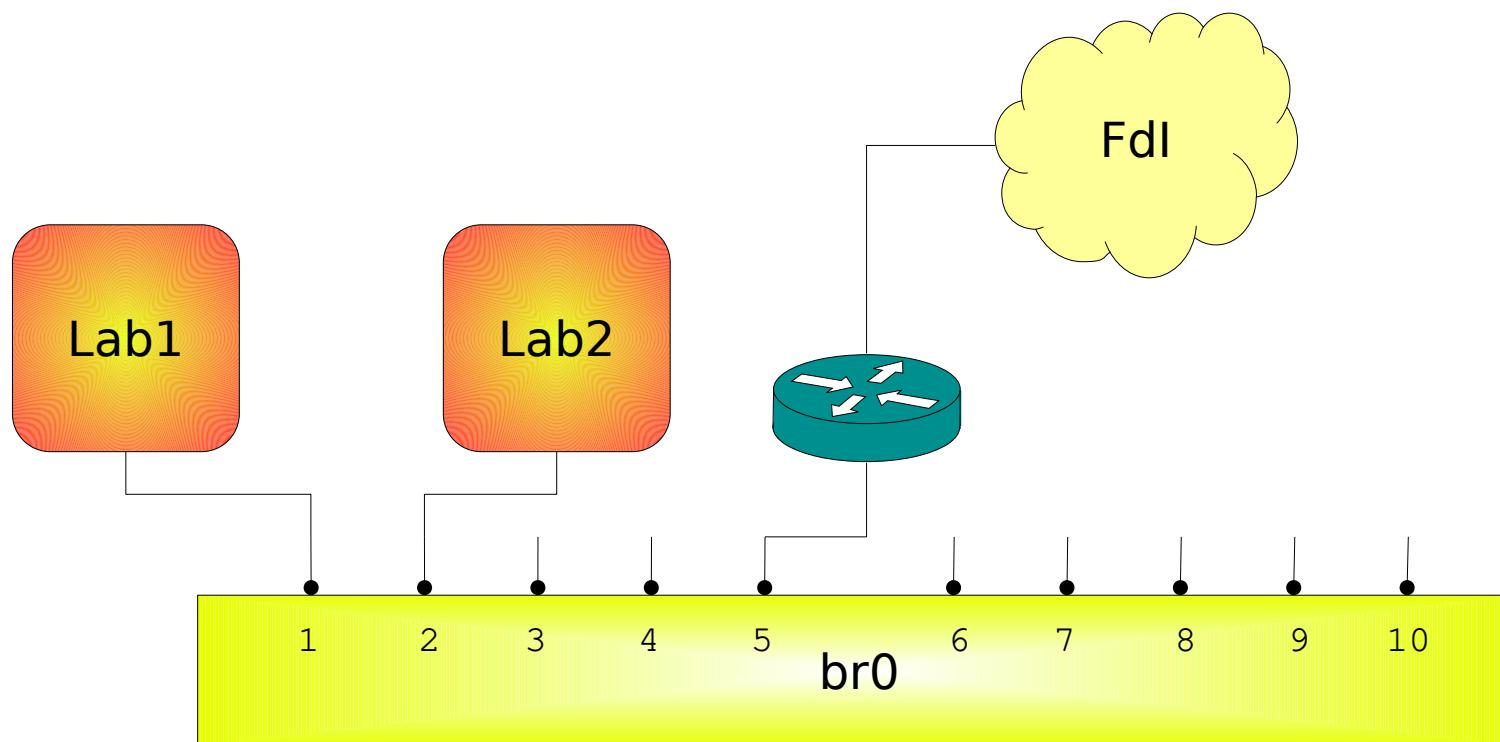


Flujos

■ ovs-ofctl

- Ejemplo. Conectividad de los puertos.

```
ovs-ofctl add-flow br0 "table=101 priority=0 actions=normal"
ovs-ofctl add-flow br0 "table=102 priority=0 actions=normal"
ovs-ofctl add-flow br0 "table=103 priority=0 actions=normal"
ovs-ofctl add-flow br0 "table=104 priority=0 actions=normal"
...
...
```

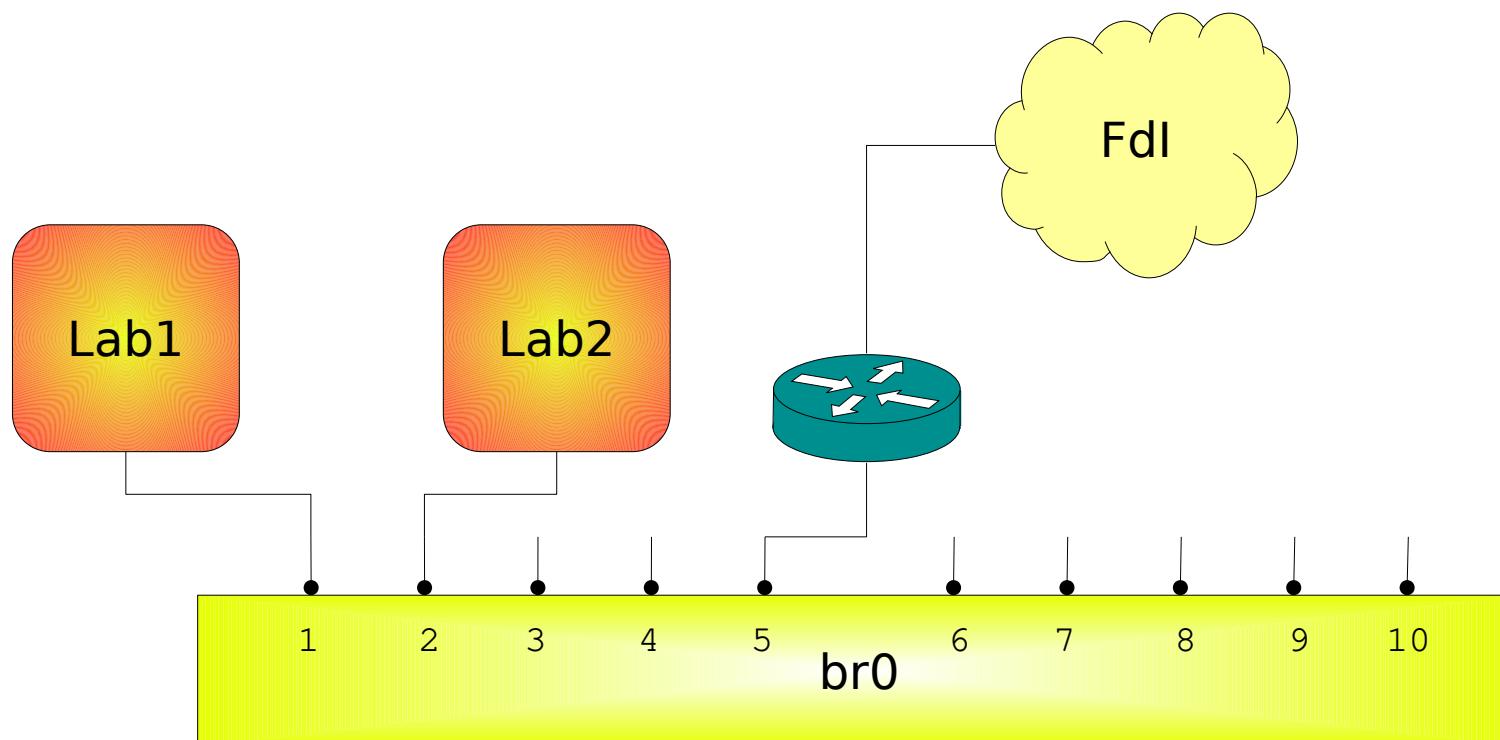


Flujos

■ ovs-ofctl

■ Ejemplo. Servidores FdI y UCM

```
ovs-ofctl add-flow br0 \
  "table=200 priority=0 ip,nw_dst=147.96.80.0/23 actions=normal"
ovs-ofctl add-flow br0 \
  "table=210 priority=0 ip nw_dst=147.96.0.0/16 actions=normal"
```



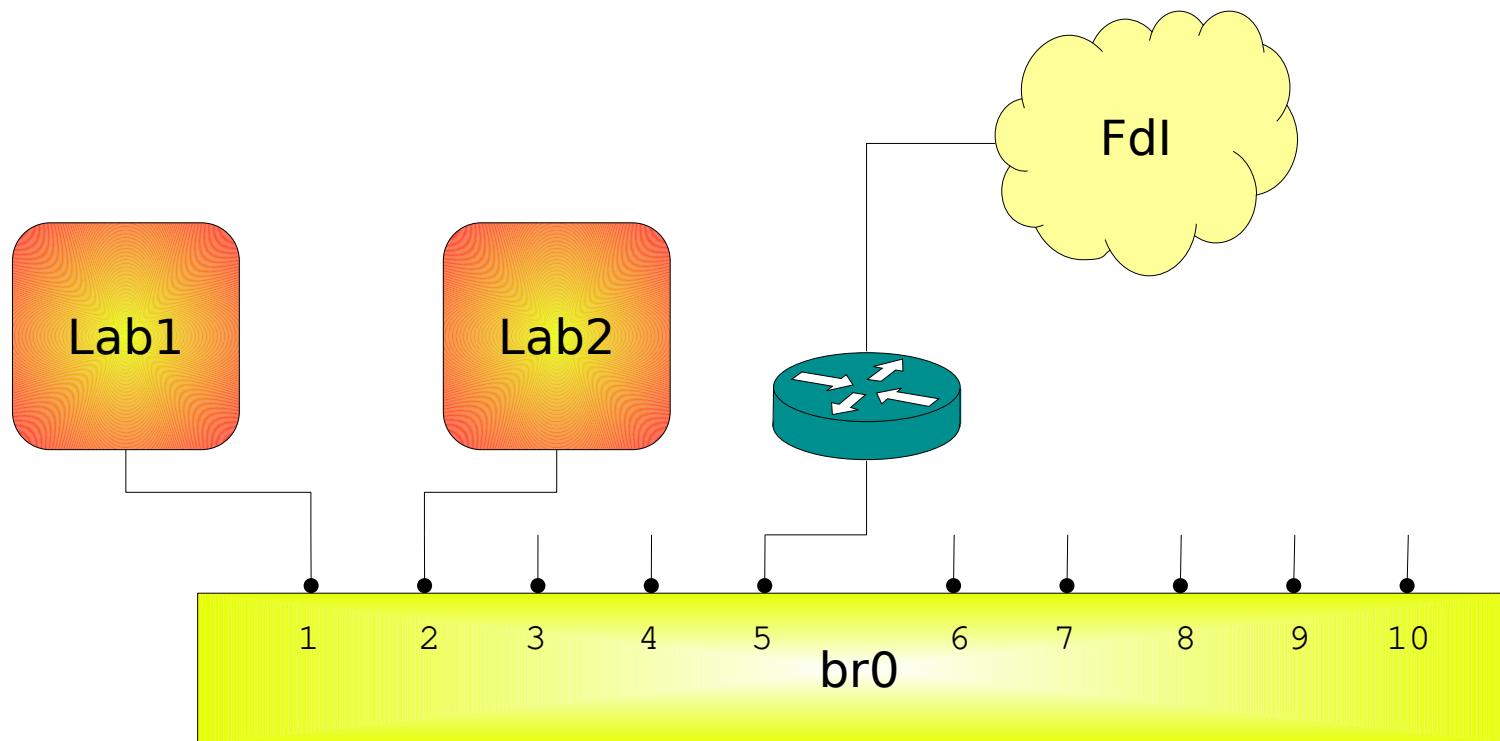
Flujos

■ ovs-ofctl

- Ejemplo. Lab1 sólo conexión a servidores Fdl.

```
ovs-ofctl add-flow br0 "table=101 priority=100 resubmit(,200)"
```

```
ovs-ofctl del-flows --strict br0 "table=101 priority=100"
```



Flujos

■ ovs-ofctl

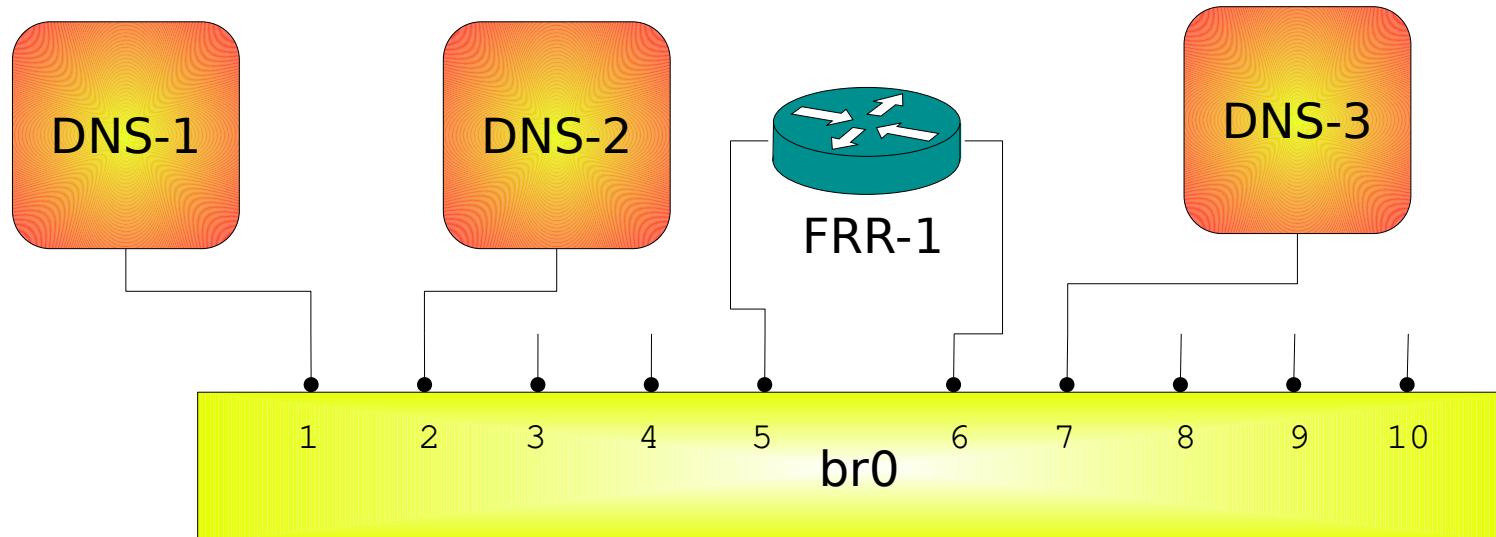
■ Práctica

Clients: DNS-1, DNS-2
Router: FRR-1
Server: DNS-3

Red clients: 192.168.1.0/24
Red server: 172.16.1.0/24

¿Y si pertenecieran a la misma red?

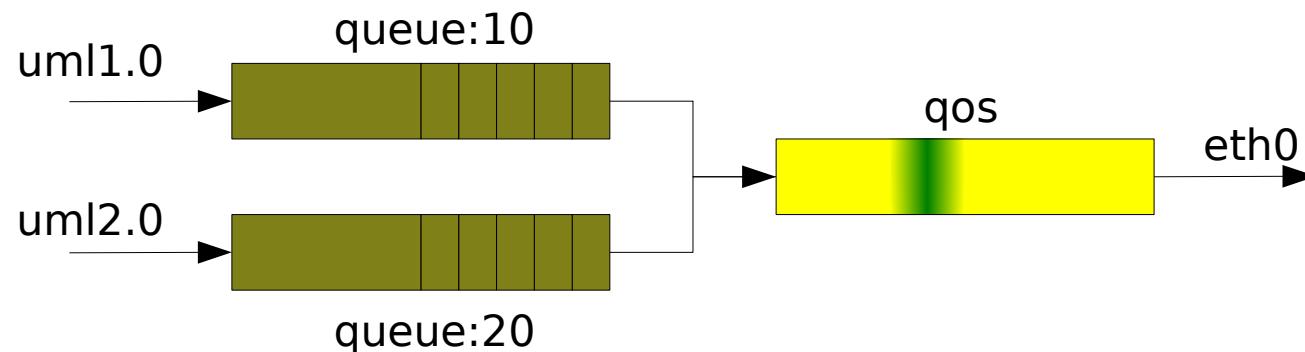
Desde DNS-1 se podrá acceder a DNS-3, pero no a DNS-2.
Desde DNS-2 se podrá acceder a DNS-3.



SDN

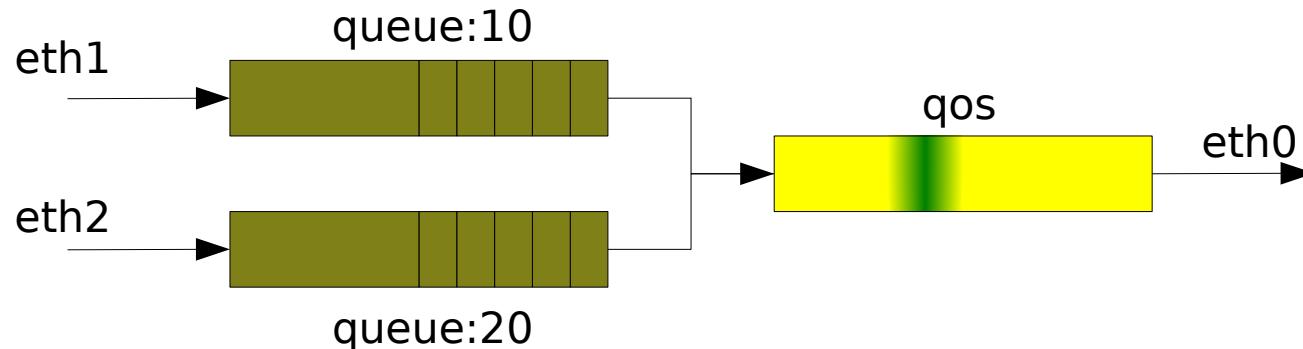
QoS

- Es posible limitar el ancho de banda mínimo y máximo asignado a cada puerto, la prioridad, tamaño de ráfaga...
 - Mediante la definición de colas en ovs-vsctl
 - Flujos OpenFlow en ovs-ofctl que utilicen las colas.
 - Definir QoS en ovs-vsctl que hagan uso de las colas.
- Ejemplo 1:
 - Interfaz físico eth0 a 1Gbps.
 - Interfaces virtuales uml1.0 y uml2.0 limitados a 10Mbps y 20Mbps, respectivamente.
- Ejemplo 2:
 - Se desea limitar el tráfico “broadcast” o “multicast”.



QoS

```
ovs-vsctl -- \
    add-br br0 -- \
    add-port br0 eth0 -- \
    add-port br0 eth1 -- set interface eth1 ofport_request=5 -- \
    add-port br0 eth2 -- set interface eth2 ofport_request=6
ovs-vsctl create queue other-config:max-rate=10000000
8ad26ce4-ee2f-4fb1-949a-3ea41e8bb6ed
ovs-vsctl create queue other-config:max-rate=20000000
15e25965-db81-474d-b609-0a435b96bce1
ovs-vsctl create qos type=linux-htb \
    other-config:max-rate=1000000000 \
    queues:10=8ad26ce4-ee2f-4fb1-949a-3ea41e8bb6ed \
    queues:20=15e25965-db81-474d-b609-0a435b96bce1
97ff4d44-de16-4ac9-8989-5c6ceff774c1
ovs-vsctl set port eth0 qos=97ff4d44-de16-4ac9-8989-5c6ceff774c1
```



- Para no tener que anotar los identificadores de las colas, se puede hacer todo en una sola orden:

```
ovs-vsctl -- \
    add-br br0 -- \
    add-port br0 eth0 -- \
    add-port br0 eth1 -- set interface eth1 ofport_request=5 -- \
    add-port br0 eth2 -- set interface eth2 ofport_request=6 -- \
    set port eth0 qos=@newqos -- \
    --id=@newqos create qos type=linux-htb \
        other-config:max-rate=100000000 \
        queues:10=@eth10queue \
        queues:20=@eth20queue -- \
    --id=@eth10queue create queue other-config:max-rate=10000000 -- \
    --id=@eth20queue create queue other-config:max-rate=20000000
```

```
ovs-vsctl list queue
_uuid                  : 8ad26ce4-ee2f-4fb1-949a-3ea41e8bb6ed
dscp                   : []
external_ids           : {}
other_config           : {max-rate="10000000"}

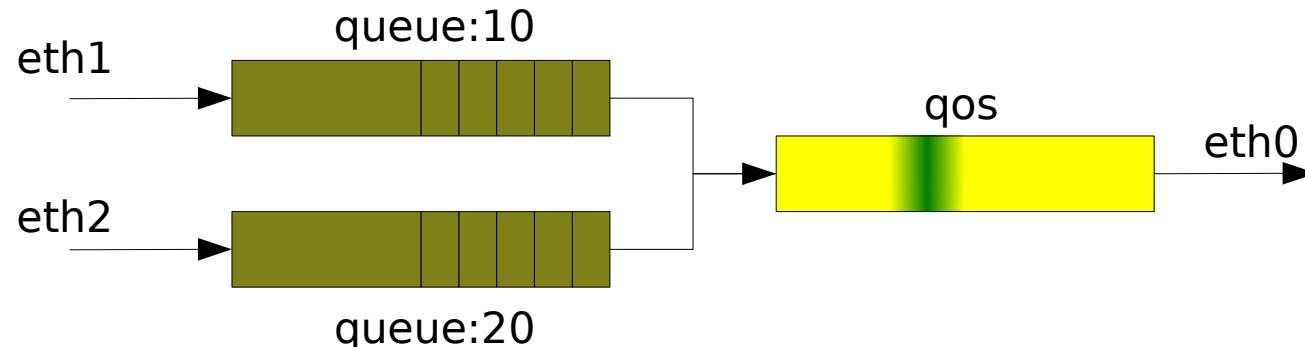
_uuid                  : 15e25965-db81-474d-b609-0a435b96bce1
dscp                   : []
external_ids           : {}
other_config           : {max-rate="20000000"}
```

QoS

- Se deben definir los flujos para utilizar las colas recién creadas.

```
ovs-vsctl list qos
_uuid          : 97ff4d44-de16-4ac9-8989-5c6ceff774c1
_external_ids   : {}
_other_config    : {max-rate="10000000000"}
_queues         : {10=8ad26ce4-ee2f-4fb1-949a-3ea41e8bb6ed,
                  20=15e25965-db81-474d-b609-0a435b96bce1}
_type          : linux-htb
```

```
ovs-ofctl add-flow br0 in_port=5,actions=set_queue:10,normal
ovs-ofctl add-flow br0 in_port=6,actions=set_queue:20,normal
```

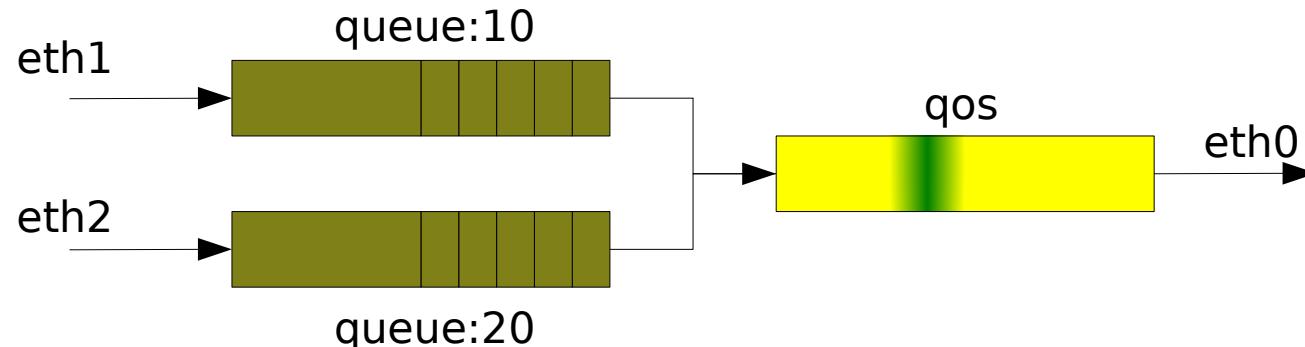


- Se pueden cambiar, en tiempo real, los parámetros de una cola:

```
ovs-vsctl set queue 8ad26ce4-ee2f-4fb1-949a-3ea41e8bb6ed \
    other_config:min-rate=1000000

ovs-vsctl list queue
_uuid                  : 8ad26ce4-ee2f-4fb1-949a-3ea41e8bb6ed
dscp                   : []
external_ids           : {}
other_config           : {max-rate="10000000", min-rate="1000000"}

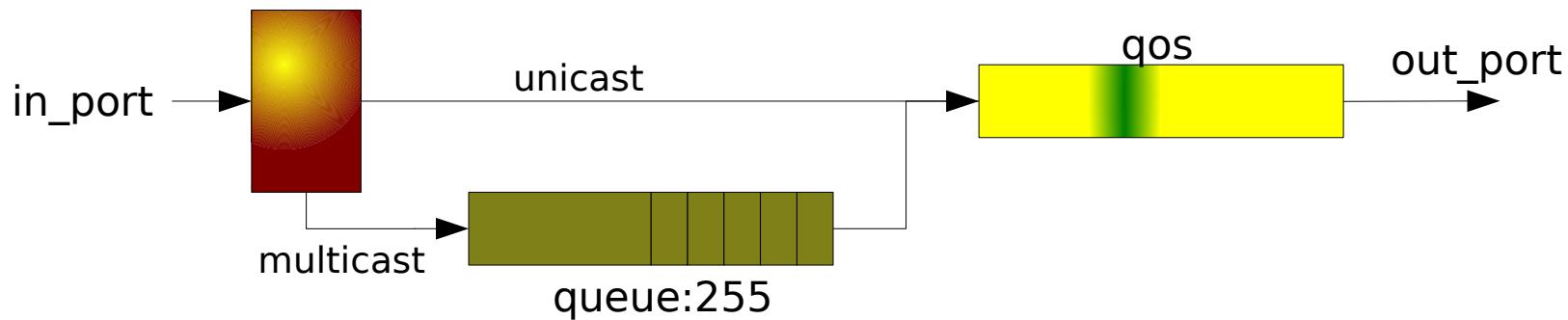
_uuid                  : 15e25965-db81-474d-b609-0a435b96bc1
dscp                   : []
external_ids           : {}
other_config           : {max-rate="20000000"}
```



- Ejemplo 2:
 - Limitar el tráfico multicast por todos los enlaces.

```
ovs-vsctl set port eth1 qos=@newqos -- \
    set port eth2 qos=@newqos -- \
    --id=@newqos create qos type=linux-htb queues:255=@brstqueue -- \
    --id=@brstqueue create queue other-config:max-rate=100000

ovs-ofctl add-flow br0 \
    "dl_dst=01:00:00:00:00:00/01:00:00:00:00:00 \\" \
    actions=set_queue:255,normal"
```



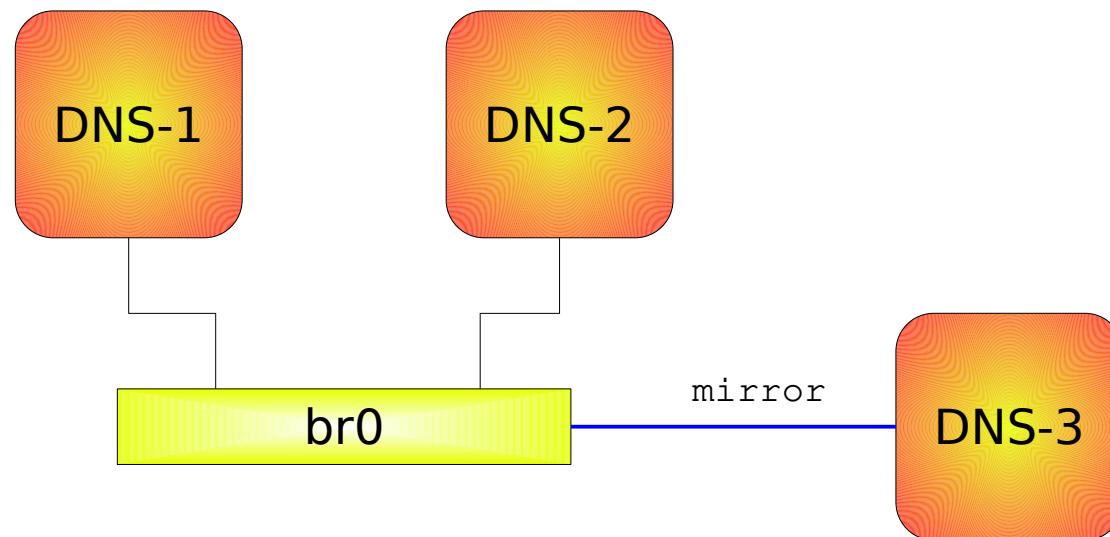
SDN

Mirroring

Mirroring

■ Mirroring o Port Span

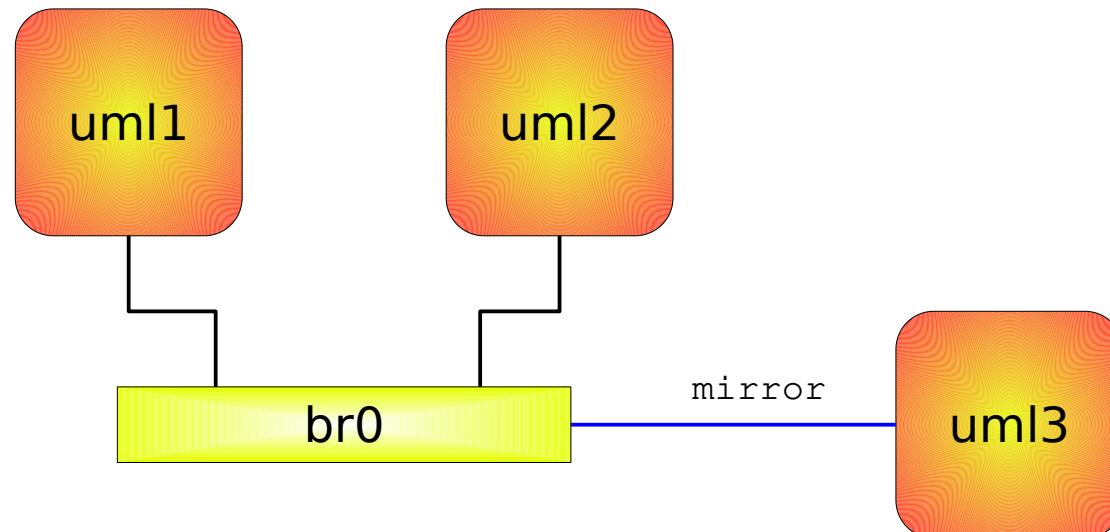
- Se copia el contenido de uno o más puertos sobre otro.
 - El puerto sobre el que se hace mirroring queda inhabilitado para tráfico normal.
- Útil para depuración, detección de situaciones anómalas en la red, IDS...



Mirroring

- Todos los puertos:

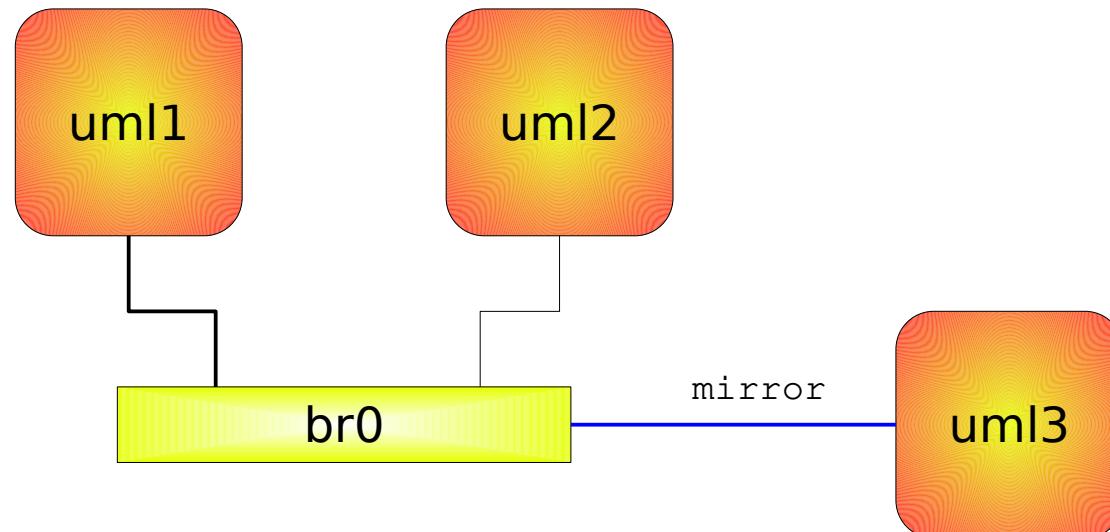
```
ovs-vsctl add-br br0
ovs-vsctl add-port br0 eth1
ovs-vsctl add-port br0 eth2
ovs-vsctl add-port br0 eth3 -- \
    --id=@p get port eth3 -- \
    --id=@m create mirror name=m0 select-all=true output-port=@p -- \
    set bridge br0 mirrors=@m
```



Mirroring

- Seleccionar sólo un puerto:

```
ovs-vsctl add-br br0
ovs-vsctl add-port br0 eth1
ovs-vsctl add-port br0 eth2
ovs-vsctl add-port br0 eth3
ovs-vsctl --id=@p1 get port eth1 -- \
--id=@p get port eth3 -- \
--id=@m create mirror name=m0 \
select_dst_port=@p1 select_src_port=@p1 output-port=@p -- \
set bridge br0 mirrors=@m
```



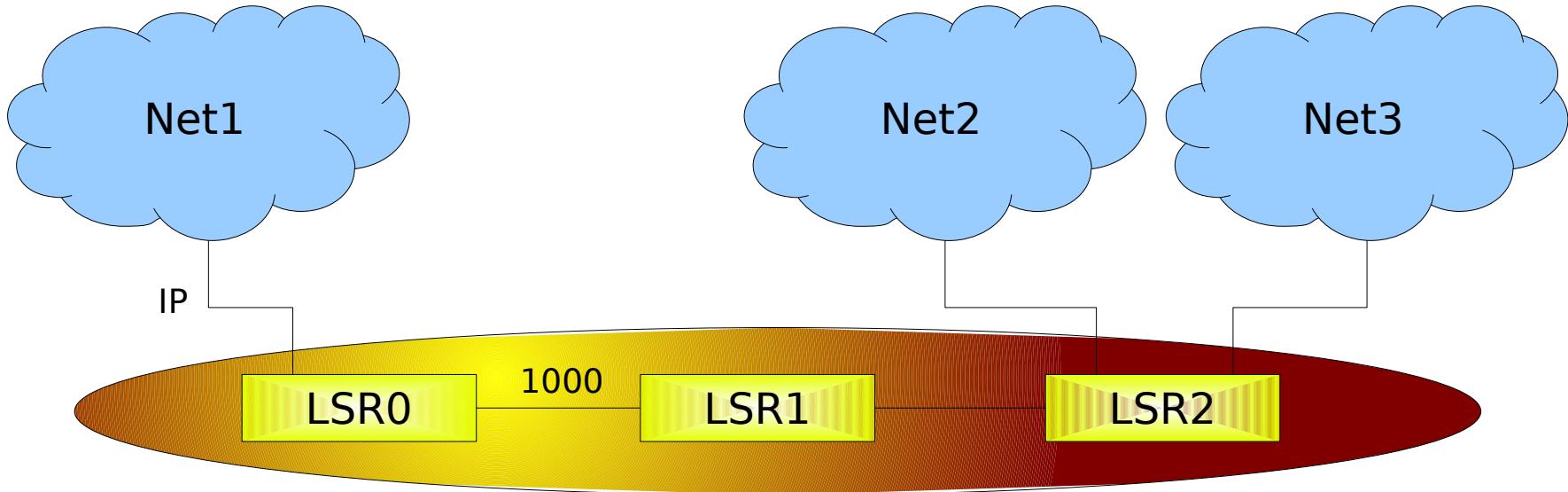
SDN

MPLS

■ Añadir una etiqueta MPLS

- Es necesario especificar el tipo Ethernet de MPLS: 0x8847 para MPLS unicast y 0x8848 para multicast.

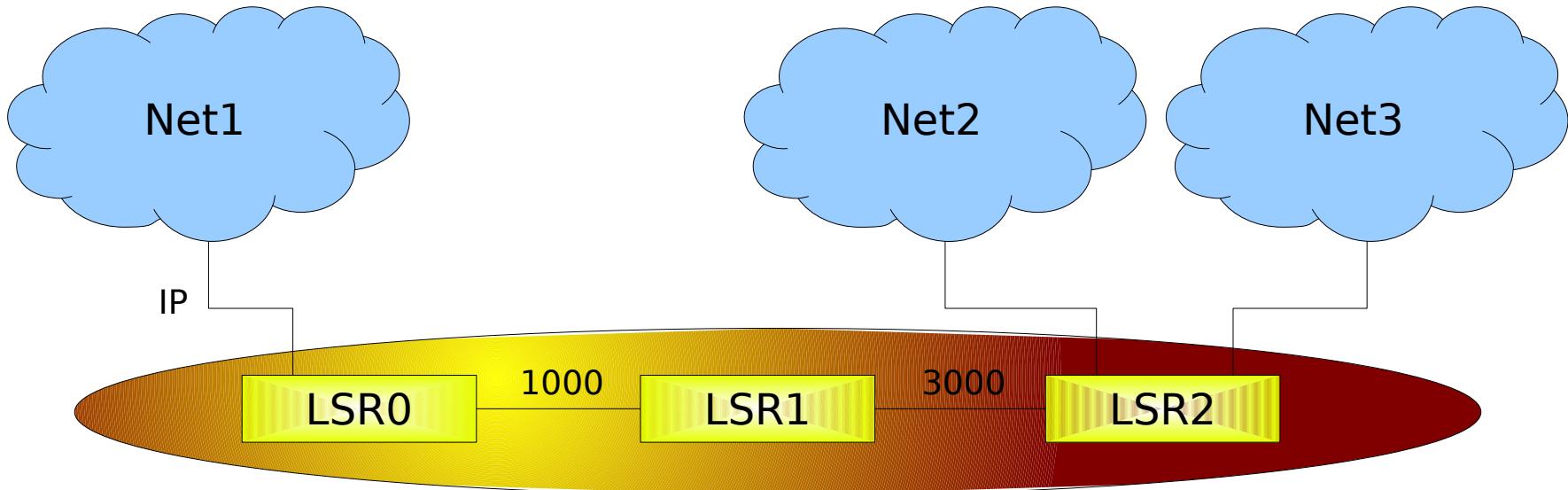
```
ovs-ofctl add-flow LSR0 'in_port=1, ip, nw_dst=192.168.0.0/16,  
actions=push_mpls:0x8847, set_field:1000->mpls_label, output:2'
```



■ Modificar etiqueta MPLS

- Ejemplo:

```
ovs-ofctl add-flow LSR1 'in_port=1, mpls, mpls_label=1000,  
actions=set_field:3000->mpls_label, output:2'
```



■ Eliminar etiqueta MPLS

- Es necesario especificar el tipo Ethernet.

```
ovs-ofctl add-flow LSR2 'in_port=1, mpls,  
mpls_label=3000, actions=pop_mpls:0x0800, output:3'
```

