Material Docente en abierto
de la Universidad Rey Juan Carlos

Apuntes: Manual para los proyectos de prácticas con NetGUI para la asignatura: Sistemas Telemáticos

2º Ingeniería Telemática,
2º Ingeniería en Tecnologías de la Telecomunicación,
2º Ingeniería en Sistemas de Telecomunicación

Autores: Eva M. Castro Barbero, José Centeno González, Pedro de las Heras Quirós
{eva.castro, jose.centeno, pedro.delasheras}@urjc.es

Material disponible en BURJC Digital: https://burjcdigital.urjc.es

Curso 2023/24

- Tema 1: NetGUI: Configuración de Switches, Cachés de ARP, IP Aliasing, Proxy ARP, VLANs
- Tema 2: NetGUI: Configuración de STP en switches
- Tema 3: NetGUI: Configuración de OSPF en Quagga
- Tema 4: NetGUI: Configuración de BGP en Quagga
- Tema 5: Herramientas para el análisis de conexiones TCP
- Tema 6: Firewalls en Linux con iptables. Configuración de un router NAT

Tema 1: NetGUI: Configuración de Switches, Cachés de ARP, IP Aliasing, VLANs

Sistemas Telemáticos

2º GIT – 2º GITT – 2º GIST

Eva M. Castro Barbero (eva.castro@urjc.es)
José Centeno González (jose.centeno@urjc.es)
Pedro de las Heras Quirós (pedro.delasheras@urjc.es)

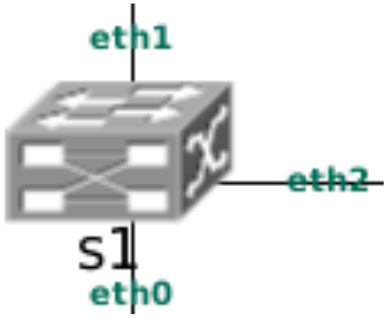
Diciembre 2023

Contenidos

Contenidos

Bridges/Switches en NetGUI

- La interfaz de NetGUI permite dibujar *bridges/switches* los cuáles están representados a través del siguiente icono:



- Estos *bridges/switches* se configuran a través del comando `brctl` que pertenece al paquete `bridge-utils` en Linux.
- Por defecto el protocolo STP está desactivado en los *switches*.

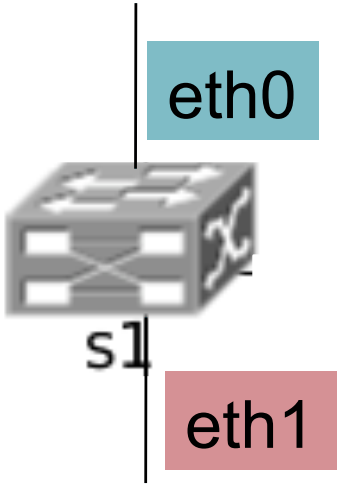
Consultar información sobre el *bridge* (I)

- **Mostrar la configuración del bridge**:

  `brctl show`

En s1:

```
s1:~# brctl show
bridge name          bridge id          STP enabled     interfaces
    s1           8000.3e323176a0de          no              eth0
                                                            eth1
```



Consultar información sobre el *bridge* (II)

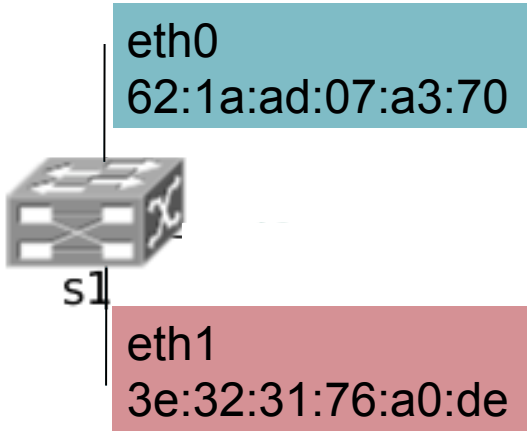
- **Mostrar la tabla de MACs aprendidas**:

```
brctl showmacs <nombreSwitch>
```

En s1:

```
s1:~# brctl showmacs s1
```

port no	mac addr	is local?	ageing timer
2	0a:29:6e:9a:3e:d4	no	11.77
2	3e:32:31:76:a0:de	yes	0.00
1	62:1a:ad:07:a3:70	yes	0.00
1	aa:da:5c:68:ed:27	no	11.77



- El `port no` enumera las interfaces empezando por 1 para `eth0`, 2 para `eth1` y así sucesivamente.
- Las líneas en amarillo contienen las MACs realmente aprendidas, las otras líneas muestran las interfaces locales del *bridge*.
- La columna `ageing timer` indica el tiempo (en segundos) que ha pasado desde que se aprendió o refrescó cada entrada. La entrada se elimina al llegar al valor máximo permitido (por defecto, 300 seg).

  NOTA: las direcciones de las interfaces locales no caducan nunca.

Borrar la tabla de MACs aprendidas por el *bridge*

- **Para eliminar las MACs aprendidas** por el *bridge* se deshabilita el funcionamiento del *bridge* con el comando:

```
ifconfig <nombre_br> down
```

En s1:

```
s1:~# ifconfig s1 down
```

- Al habilitar de nuevo el *bridge*, éste no tendrá ninguna MAC aprendida, salvo las de sus interfaces locales:

```
s1:~# ifconfig s1 up
s1:~# brctl showmacs s1
port no     mac addr      is local?  ageing timer
   2    3e:32:31:76:a0:de   yes         0.00
   1    62:1a:ad:07:a3:70   yes         0.00
```

- **Modificar el tiempo de caducidad de las entradas en la tabla de MACs aprendidas** (por defecto 300 seg):

```
brctl setageing <nombre_br> <tiempoEnSeg
```

En s1:

```
s1:~# brctl setageing s1 1
```

Contenidos

Caché de ARP en pcs y routers

- Para consultar la caché de ARP en una máquina se utiliza el comando `arp`:

```
pc2:~# arp -a
  ? (10.0.0.1) at 0A:29:92:55:93:70 [ether] on eth0
  ? (10.0.0.2) at 0B:39:12:54:83:71 [ether] on eth0
```

- Para borrar una entrada de la caché de ARP se utiliza la opción `-d` del comando `arp`:

```
pc2:~# arp -d 10.0.0.2
```

Si se consulta la caché de ARP ahora:

```
pc2:~# arp -a
  ? (10.0.0.1) at 0A:29:92:55:93:70 [ether] on eth0
  ? (10.0.0.2) at <incomplete> on eth0
```

Contenidos

Configuración de un *router* para que haga Proxy ARP

- Proxy ARP se utiliza en un *router* para que responda a solicitudes de ARP que preguntan una dirección MAC que no se corresponde con ninguna de las interfaces de red del *router*.
- Configuración:
  1. Para activar Proxy ARP en la interfaz `eth0` de un *router*:

     ```
     echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
     ```

  2. Para que el *router* responda con su dirección MAC a una solicitud de ARP preguntando por una cierta dirección:

     ```
     arp -i <interfaz_resp> -Ds <dirIP> <interfaz_MAC> netmask <máscara>
     ```

     - `<interfaz_resp>`: Interfaz del *router* que se utilizará para enviar la respuesta de ARP.
     - `<dirIP>`: Dirección IP de la solicitud de ARP para la que el *router* debe responder.
     - `<interfaz_MAC>`: La dirección MAC que irá en la respuesta de ARP será la dirección MAC de la interfaz `<interfaz_MAC>` del *router*
     - `<máscara>`: Máscara que permite hacer proxy ARP para un rango de direcciones IP definido por `<dirIP>` y `<máscara>`. Si la máscara es 255.255.255.255, se realizará el proxy ARP para una dirección IP concreta en vez de para un rango de direcciones IP.

  3. Además puede ser necesario introducir en la tabla de encaminamiento del *router* una entrada adecuada para que, una vez le lleguen los datagramas IP gracias al proxy ARP, los reenvíe a su destino por la interfaz correcta.

Ejemplo

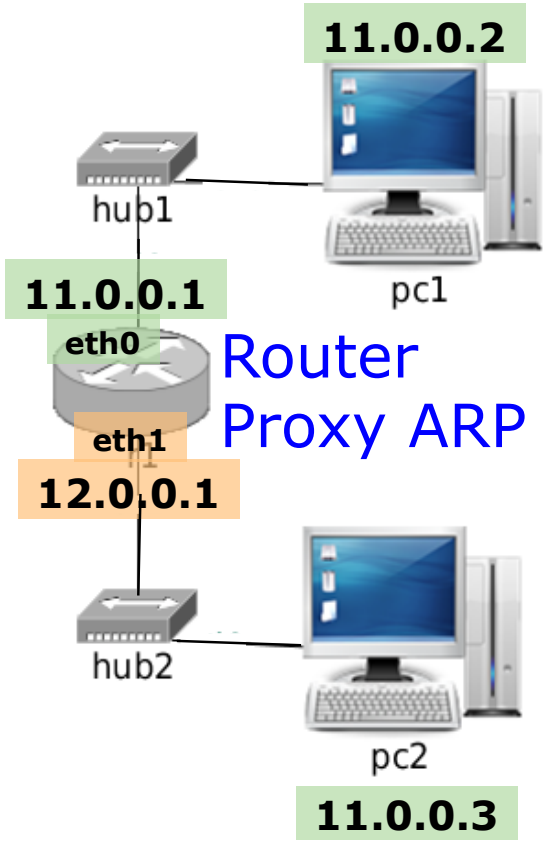
- Para que ser reciba tráfico en el sentido pc1→pc2:
  - El *router* debe responder con la MAC de `eth0` cuando pc1 mande una petición de ARP solicitando la dirección MAC de 11.0.0.3
  - En la tabla de encaminamiento del *router* hay que añadir una entrada específica para poder encaminar hasta 11.0.0.3 por la interfaz adecuada

```
r1:~# echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
r1:~# arp -i eth0 -Ds 11.0.0.3 eth0 netmask 255.255.255.255
r1:~# route add -host 11.0.0.3 dev eth1
```

- Para que se reciba el tráfico en el sentido pc2→pc1 habría que realizar una configuración análoga a la anterior.

```
r1:~# echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
r1:~# arp -i eth1 -Ds 11.0.0.2 eth1 netmask 255.255.255.255
```

(Nótese que en este caso no es necesario añadir ninguna ruta, ya que el *router* ya tendrá una ruta para la red 11.0.0.0/24 a través de `eth0`)



Contenidos

Configuración con la orden ip

- Ejemplo de asignación de las direcciones 11.0.0.1/24 y 12.0.0.1/24 a la interfaz eth0 de r1:

```
r1:~# ip link set eth0 up
r1:~# ip address add dev eth0 11.0.0.1/24 broadcast +
r1:~# ip address add dev eth0 12.0.0.1/24 broadcast +
r1:~# ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
  link/ether de:34:80:65:19:5a brd ff:ff:ff:ff:ff:ff
  inet 11.0.0.1/24 brd 11.0.0.255 scope global eth0
  inet 12.0.0.1/24 brd 12.0.0.255 scope global eth0
  inet6 fe80::dc34:80ff:fe65:195a/64 scope link
    valid_lft forever preferred_lft forever
```

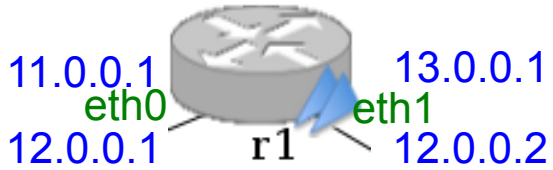
Configuración con la orden `ifconfig`

- Para utilizar IP aliasing con `ifconfig` es necesario referirse a las interfaces "virtuales" `eth0:0`, `eth0:1`... como aquellas que tendrán las direcciones IP adicionales a la primera que se asigne a `eth0`.
- Ejemplo de asignación de las direcciones 11.0.0.1/24 y 12.0.0.1/24 a la interfaz `eth0` de `r1`:

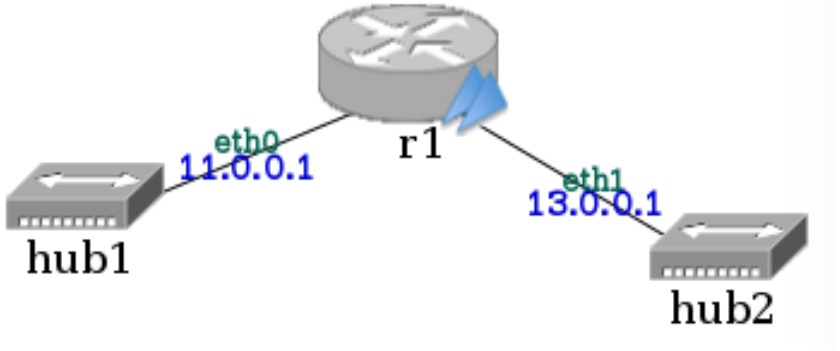
```
r1:~# ifconfig eth0 11.0.0.1 netmask 255.255.255.0
r1:~# ifconfig eth0:0 12.0.0.1 netmask 255.255.255.0
r1:~# ifconfig
amarillo
eth0      Link encap:Ethernet  HWaddr 26:33:2A:36:35:4A
          inet addr: 11.0.0.1  Bcast:11.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::2433:2aff:fe36:354a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:468 (468.0 b)
          Interrupt:5

eth0:0    Link encap:Ethernet  HWaddr 26:33:2A:36:35:4A
          inet addr: 12.0.0.1  Bcast:12.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

Ejemplo de configuración con IP aliasing (I)

- Se desean configurar las siguientes direcciones IP en `r1`:

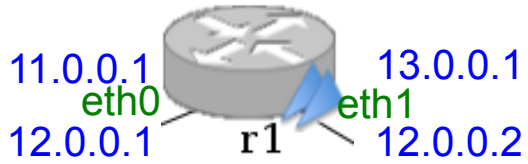


En NetGUI sólo se muestra la primera dirección IP configurada



Ejemplo de configuración con IP aliasing (II)

- Una vez añadidas las direcciones, si se consulta la tabla de encaminamiento en `r1`:




```
r1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
11.0.0.0        *               255.255.255.0   U     0      0        0 eth0
12.0.0.0        *               255.255.255.0   U     0      0        0 eth0
12.0.0.0        *               255.255.255.0   U     0      0        0 eth1
13.0.0.0        *               255.255.255.0   U     0      0        0 eth1
r1:~#
```

Ejemplo de configuración con IP aliasing (III)

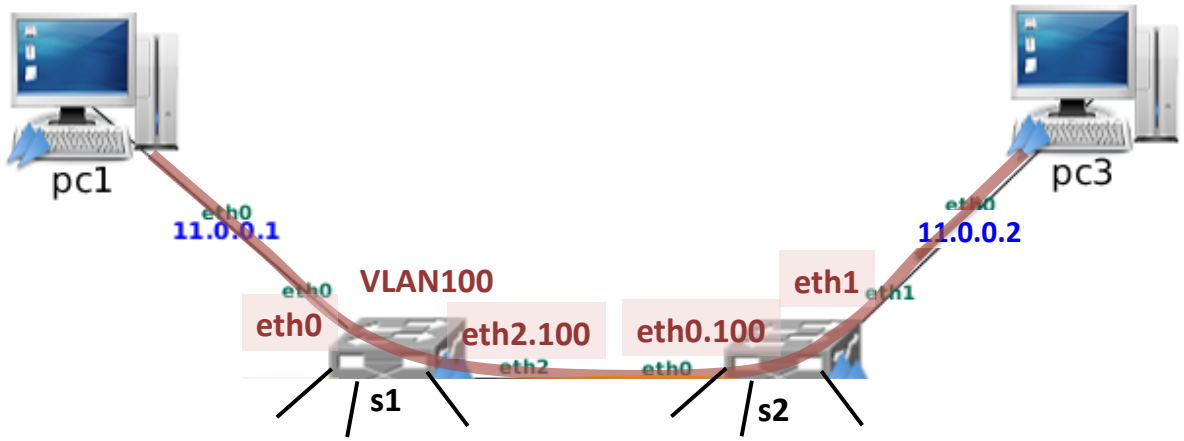
- Suponiendo que hay una sola máquina de la subred 12.0.0.0/24 conectada a la interfaz `eth0` de `r1`, p. ej. la máquina 12.0.0.100 y que en la interfaz `eth1` de `r1` hay varias máquinas de la subred 12.0.0.0/24, es conveniente dejar la tabla de encaminamiento de la siguiente forma:

```
r1:~# route del -net 12.0.0.0 netmask 255.255.255.0 dev eth0
r1:~# route add -host 12.0.0.100 dev eth0
r1:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
12.0.0.100      *               255.255.255.255 UH    0      0        0 eth0
11.0.0.0        *               255.255.255.0   U     0      0        0 eth0
12.0.0.0        *               255.255.255.0   U     0      0        0 eth1
13.0.0.0        *               255.255.255.0   U     0      0        0 eth1
r1:~#
```

Contenidos

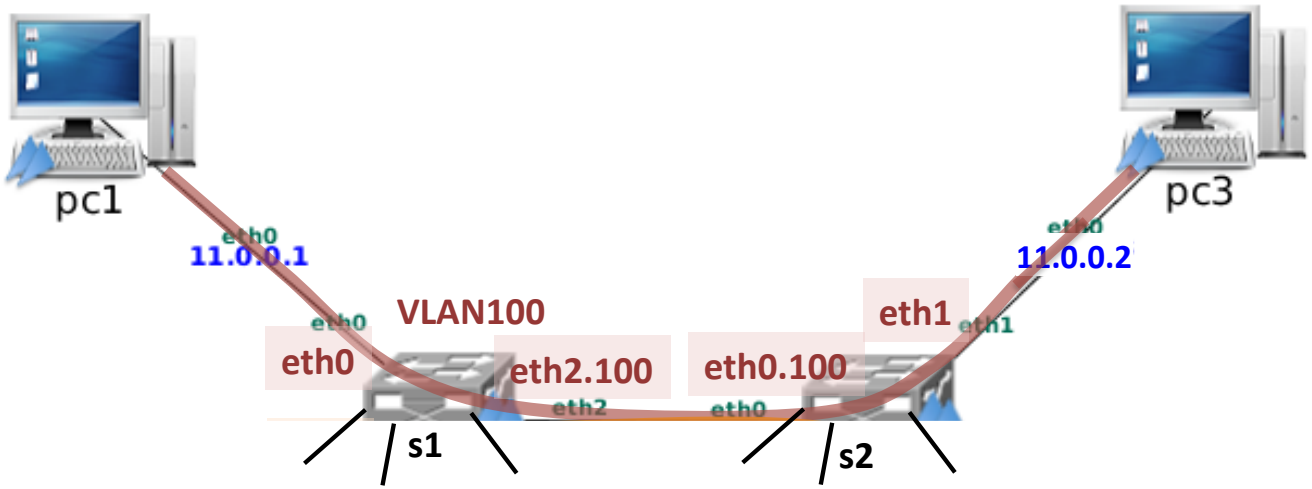
Identificar interfaces VLANs

- Para configurar una VLAN es necesario determinar qué interfaces físicas del *switch* van a pertenecer a esa VLAN y si esas interfaces son:
  - **Interfaces sin ID VLAN**: `eth0`, `eth1`, etc. A través de este tipo de interfaces se envían/reciben tramas sin la etiqueta VLAN. Normalmente a estas interfaces están conectados dispositivos finales.
  - **Interfaces con ID VLAN**: se definen con el nombre de la interfaz seguido del identificador de VLAN. Por ejemplo, para la VLAN 100: `eth2.100`, `eth0.100`, etc. Las tramas que se reciben/envían en estas interfaces llevan etiqueta VLAN y dicha etiqueta no se modifica. Estas interfaces normalmente conectan *switches* y a través de ellas viajan diferentes VLANs.



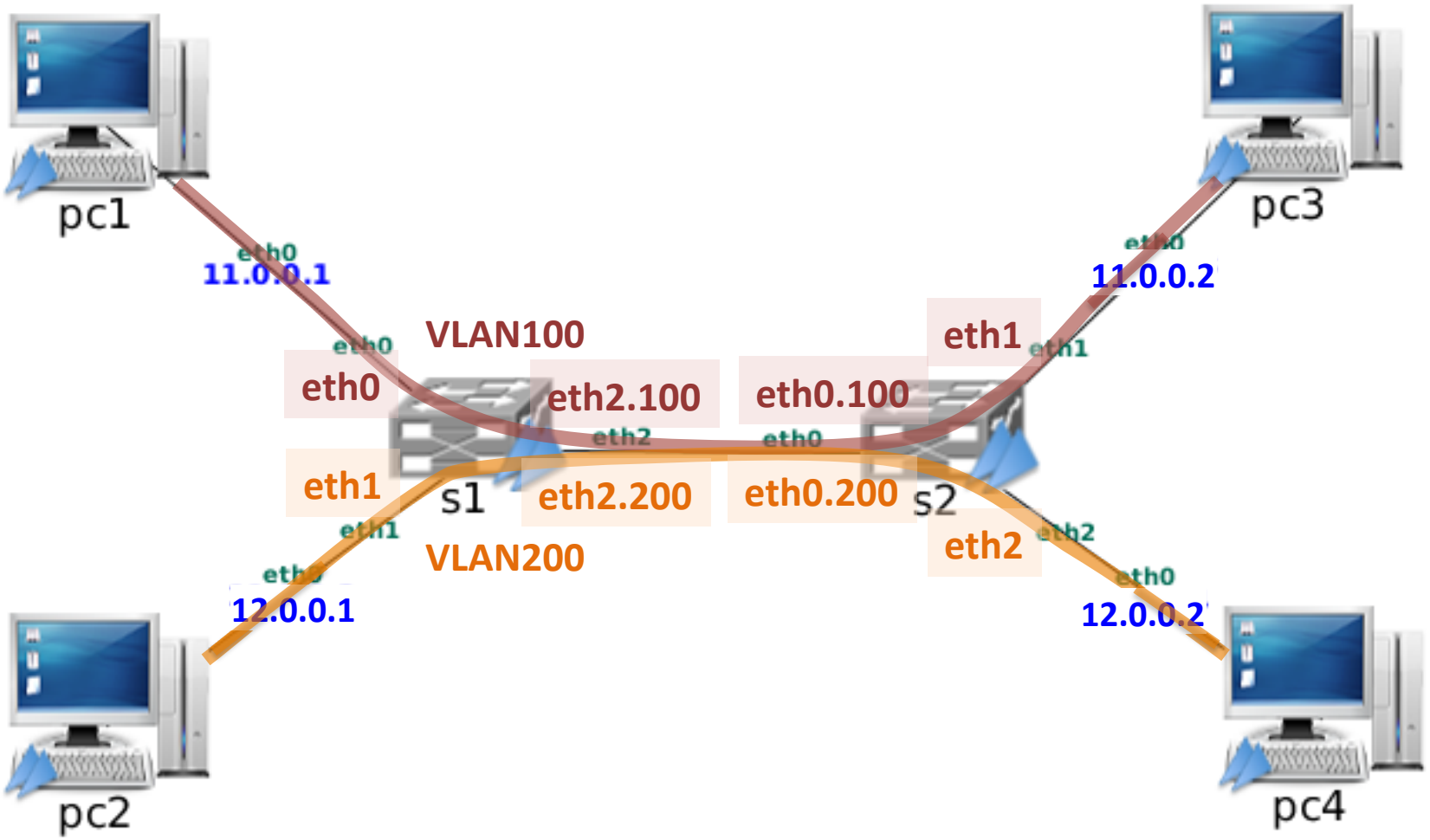
Reenvío entre interfaces de la misma VLANs

- Una vez identificadas las interfaces es necesario configurar el reenvío entre las interfaces de una determinada VLAN. Por ejemplo, en VLAN100:
  - En s1 configurar un *switch* que reenvíe tráfico entre las interfaces `eth0` y `eth2.100`.
  - En s2 configurar un *switch* que reenvíe tráfico entre las interfaces `eth0.100` y `eth1` .



Ejemplo de configuración de 2 VLANs

- En la figura se muestran 2 VLANs: VLAN100 y VLAN200:



Configuración de VLANs en los switches NetGUI

- **PASO 0**: Antes de comenzar la configuración de las VLANs en un *switch* de NetGUI es necesario eliminar la configuración por defecto del *switch*.
- Para configurar VLANs en un *switch* Linux se van a realizar los siguientes pasos:
  1. **PASO 1**: Crear las interfaces con ID VLAN, interfaces trunk del *switch*: `vconfig`.
  2. **PASO 2**: Activar las interfaces con ID VLAN que se corresponden con interfaces de tipo *trunk*: `ifconfig`
  3. **PASO 3**: Crear el *switch* virtual: `brctl`
  4. **PASO 4**: Configurar la función de reenvío de tramas Ethernet entre interfaces de una misma VLAN dentro del *switch* virtual: `brctl`
- En los pcs y *routers* no se configurarán VLANs, para ellos será transparente el uso de VLANs.

PASO 0: Eliminar la configuración por defecto en el switch

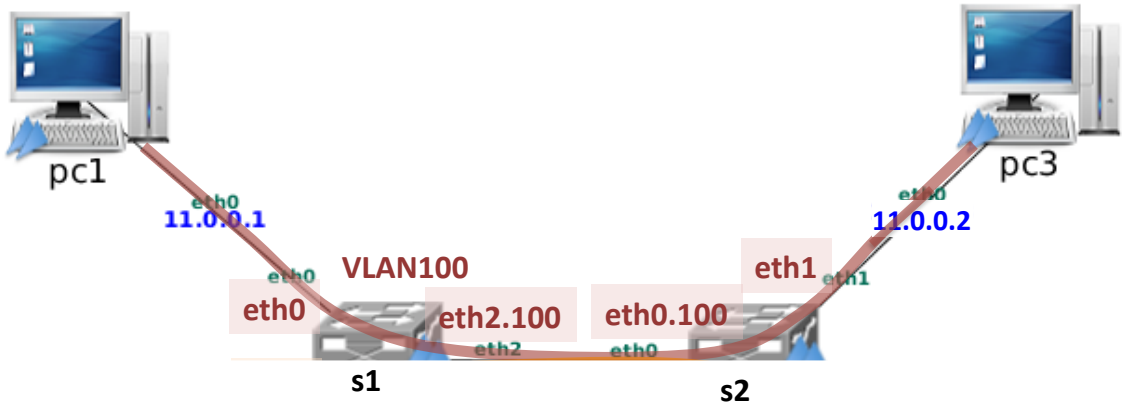
- Borrar la configuración del *switch* definido sin VLANs. Si el *switch* se llama `s1`, es necesario ejecutar lo siguiente:

```
s1:~# ifconfig s1 down
s1:~# brctl delbr s1
```

- En la figura anterior que muestra la VLAN100 y VLAN200 sería necesario borrar la configuración por defecto de los *switches*. `s1` y `s2`.

Configuración de VLAN 100 (I)

PASO 1: Crear las interfaces VLAN, las interfaces trunk, de la VLAN 100



1. **Crear las interfaces con ID VLAN, las interfaces trunk, de la VLAN 100:** `s1(eth2)` **y** `s2(eth0)`.

   `vconfig add <interfaz> <idVLAN>`

   Al ejecutar esta orden, se crea una interfaz virtual con el nombre `<interfaz>.<idVLAN>`.
   Por ejemplo, para especificar VLAN100 en `s1(eth2)`:

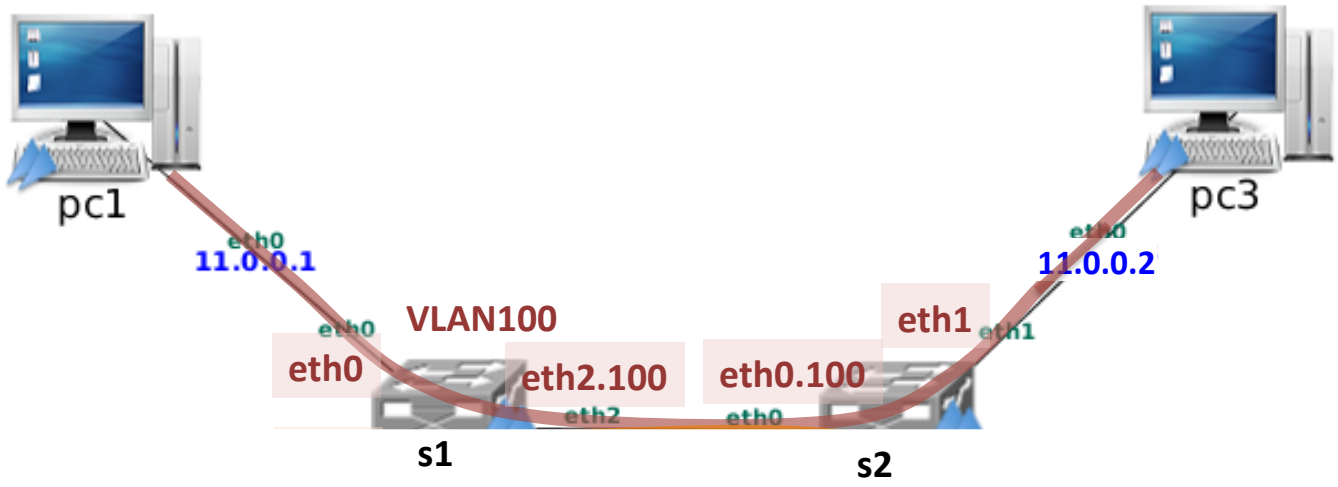
   ```
   s1:~# vconfig add eth2 100
   ```

   Para la configuración de VLAN100 habrá que usar `vconfig` en `s1(eth2)` y `s2(eth0)` creando las interfaces:

   - En `s1`: `eth2.100`
   - En `s2`: `eth0.100`

Configuración de VLAN 100 (II)

PASO 2: Activar las interfaces VLAN que se corresponden con interfaces *trunk*



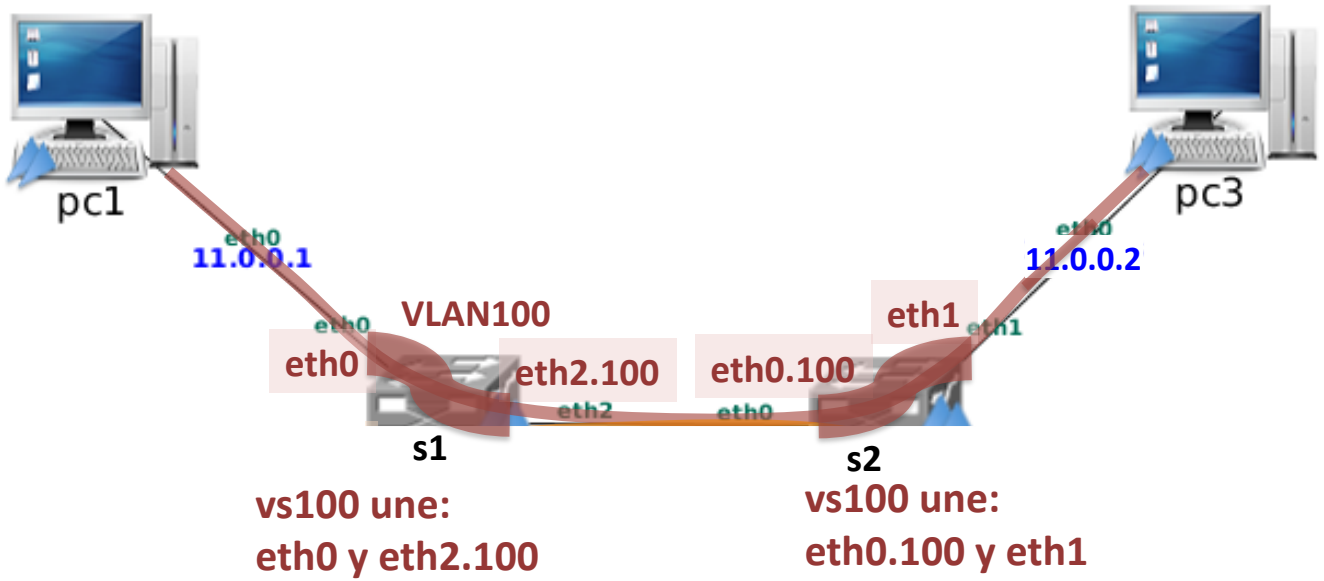
2. Una vez creadas las interfaces con ID VLAN, es necesario **activar todas las interfaces VLAN asociadas a una interfaz trunk**. En el ejemplo, sería necesario activar en s1 `eth2.100` y en s2 `eth0.100`. Por ejemplo, para activar la interfaz `eth2.100` de `s1` es necesario:

```
s1:~# ifconfig eth2.100 up
```

Al ejecutar `ifconfig` en `s1` se observará que se ha creado la interfaz VLAN `eth2.100` asociada a la interfaz `eth2`.

Configuración de VLAN 100 (III)

PASO 3: Switch virtual para una VLAN (I)



3. Crear *switches* virtuales para cada VLAN.

   En el dispositivo *switch* se crearán tantos *swicthes* virtuales superpuestos como VLANs diferentes estén definidas en dicho dispositivo. Estos *swtiches* virtuales se crean con el comando `brctl`:

   ```
   brctl addbr <nombreSwitchVirtual>
   ```

   Cada uno de esos *switches* virtuales estará encargado de hacer el reenvío de tramas de su VLAN.

Configuración de VLAN 100 (IV)

PASO 3: Switch virtual para una VLAN (II)

En s1 y s2 habrá que crear un *switch* virtual para el reenvío de tramas de VLAN100. Por ejemplo:

```
s1:~# brctl addbr vs100
```

```
s2:~# brctl addbr vs100
```

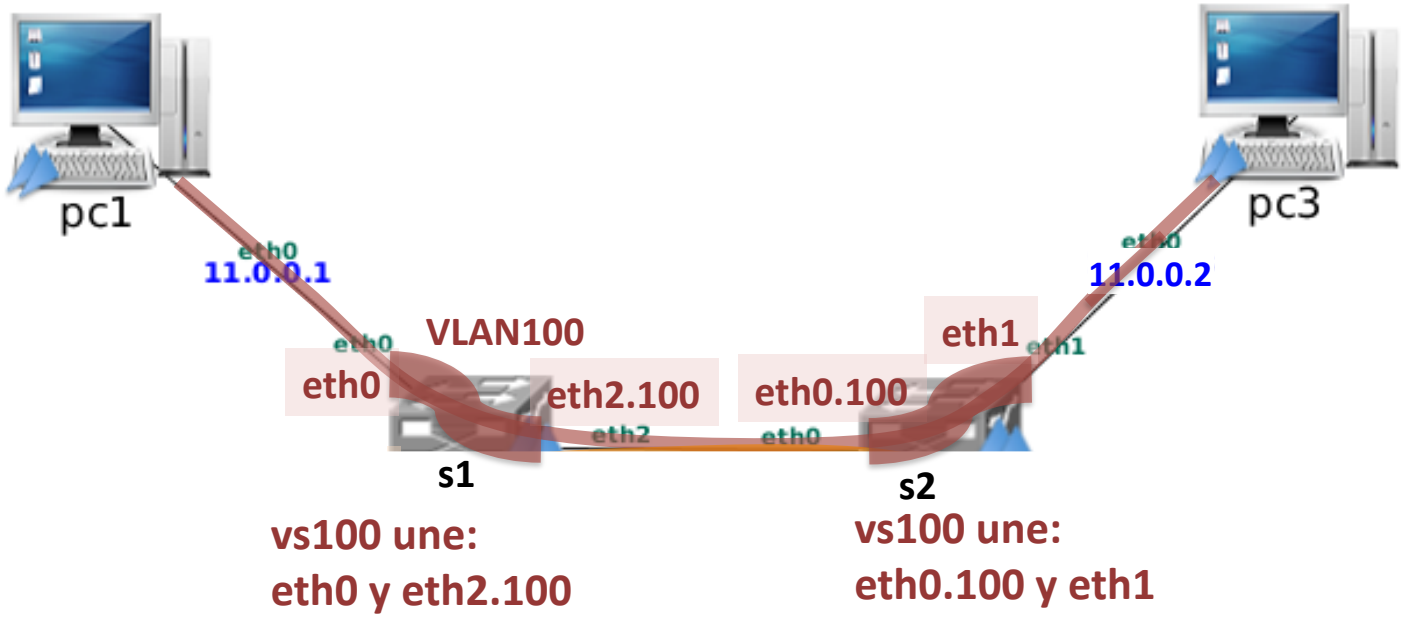
Los nombres de los *switches* pueden ser cualesquiera, elegimos el mismo para que indiquen que tanto el vs100 de s1 como el vs100 de s2 se utilizarán para la VLAN100.

Adicionalmente, cuando se configure VLAN200 será necesario crear otro *switch* virtual en s1, por ejemplo vs200, que deberá funcionar simultáneamente con vs100. Cada uno de ellos estará encargado de realizar el reenvío de tramas en VLAN100 y VLAN200 respectivamente.

E igualmente en s2.

Configuración de VLAN 100 (V)

PASO 4: Interfaces para el reenvío en una VLAN



4. Especificar las interfaces sue el *switch* virtual va a utilizar para realizar el reenvío de tráfico en cada *switch*:

```
brctl addif <nombreSwitchVirtual> <interfaz>
```

Reenvío de VLAN100 en s1:

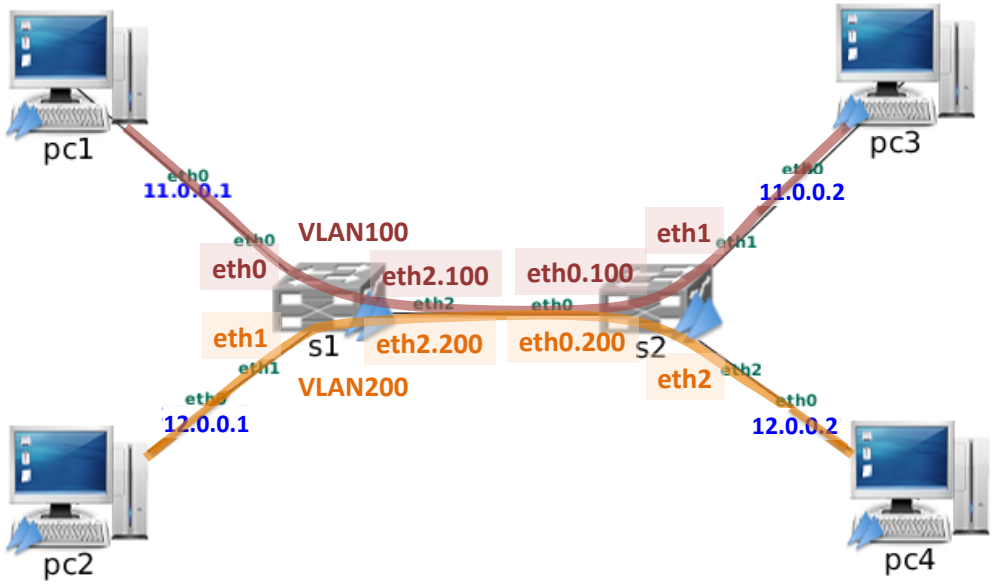
```
s1:~# brctl addif vs100 eth0
s1:~# brctl addif vs100 eth2.100
s1:~# ifconfig vs100 up
```

Reenvío de VLAN100 en s2:

```
s2:~# brctl addif vs100 eth0.100
s2:~# brctl addif vs100 eth1
s2:~# ifconfig vs100 up
```

Configuración de VLAN 200

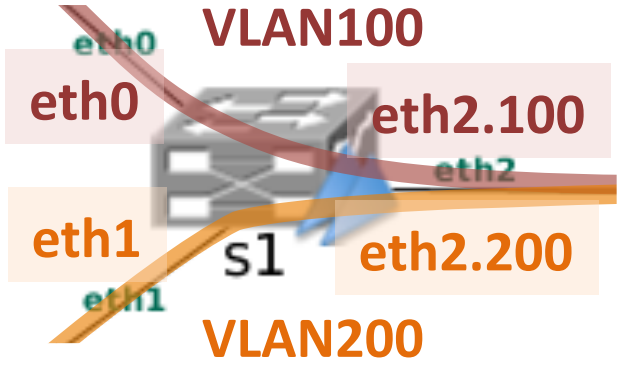
- Una vez configurada VLAN100, para configurar VLAN200:
  1. Crear las interfaces con ID VLAN de VLAN200: en s1 `eth2.200` y en s2 `eth0.200`
  2. Activar en s1 la interfaz `eth2.200` y en s2 `eth0.200`.
  3. Crear los *switches* virtuales para VLAN200 en `s1` y `s2`, por ejemplo: `vslan200` para `s1` y `vslan200` para `s2`.
  4. Configurar la función de reenvío de tramas Ethernet entre las interfaces de `s1` y `s2`



Comprobar la configuración del reenvío en un switch

- Una vez configurado el reenvío en un *switch*, con el comando `brctl show` puede comprobarse la configuración aplicada. Así en `s1`:

```
s1:~# brctl show
bridge name     bridge id               STP enabled     interfaces
vs100           8000.1a65e4986698       no              eth0
                                                        eth2.100
vs200           8000.1a65e4986698       no              eth1
                                                        eth2.200
```



Tema 2: NetGUI: Spanning Tree Protocolo (STP)

Sistemas Telemáticos

2º GIT – 2º GITT – 2º GIST

Eva M. Castro Barbero (eva.castro@urjc.es)
José Centeno González (jose.centeno@urjc.es)
Pedro de las Heras Quirós (pedro.delasheras@urjc.es)

Diciembre 2023

Contenidos

Activar/Desactivar STP

- Por defecto el protocolo STP está desactivado en los *switches* de NetGUI:

```
s1:~# brctl show
bridge name     bridge id               STP enabled     interfaces
s1              1000.3233cd147f77       no              eth0
                                                        eth1
```

- **Activar STP**:

```
brctl stp <nombre_br> on
```

En s1:

```
s1:~# brctl stp s1 on
```

- **Desactivar STP**:

```
brctl stp <nombre_br> off
```

En s1:

```
s1:~# brctl stp s1 off
```

Contenidos

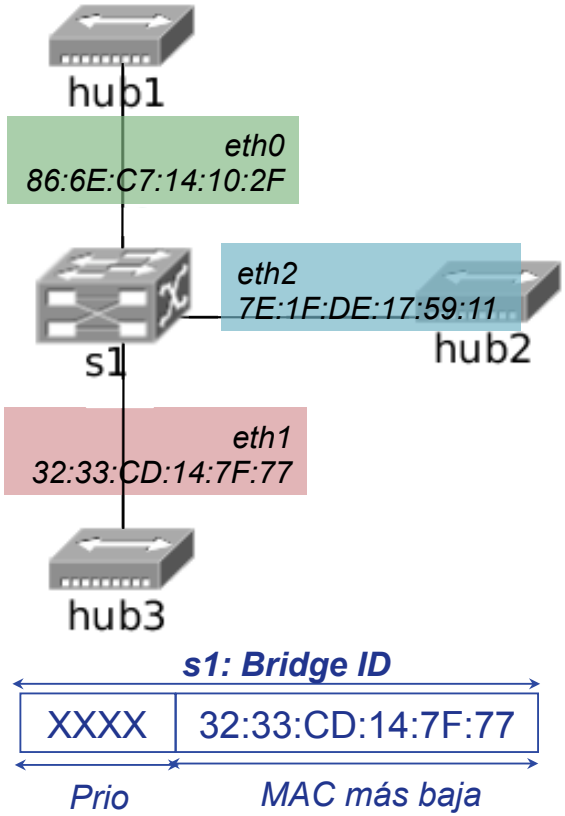
Identificador de un switch

- El *Bridge ID* se construye concatenando una **prioridad** con la **dirección MAC más baja** de las que tiene el *bridge*.
- La prioridad puede fijarse por el administrador. Por defecto es 0x8000.
- **Para asignar una prioridad** a un *switch*:

```
brctl setbridgeprio <nombre_br> <prioridad>
```

En s1:

```
s1:~# brctl setbridgeprio s1 0x1000
```

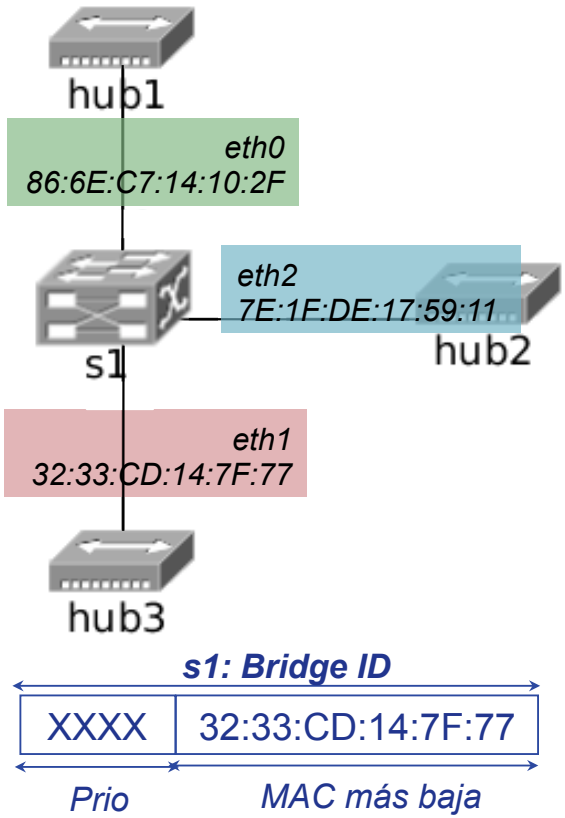


- Consultamos la información del *switch*:

```
s1:~# brctl show
bridge name     bridge id               STP enabled     interfaces
s1              1000.3233cd147f77       yes             eth0
                                                        eth1
                                                        eth2
```

Identificador de un puerto de un switch

- Los puertos de un switch están identificados en la figura de NetGUI con el nombre: eth0, eth1, eth2, etc.
- STP numera estos puertos comenzando por el 8001. Es decir la interfaz eth0 se corresponde con el número de puerto 8001, la interfaz eth1 con el número 8002, la interfaz eth2 con el número 8003, etc



Contenidos

Mostrar el estado del protocolo STP de un *switch* (I)

- El resultado de ejecutar `brctl showstp <nombre_br>` proporciona información que se puede clasificar en:
  - **sección general** del *bridge*
  - **secciones particulares de cada una de las interfaces** del *bridge*.
- En el ejemplo anterior, se ha coloreado con amarillo cada una de estas secciones: `s1`, `eth0(1)`, `eth1(2)` ...
- Mostrar el estado del protocolo STP en un *bridge* proporciona mucha información del funcionamiento de STP. Nosotros nos centraremos en los campos resaltados en verde.

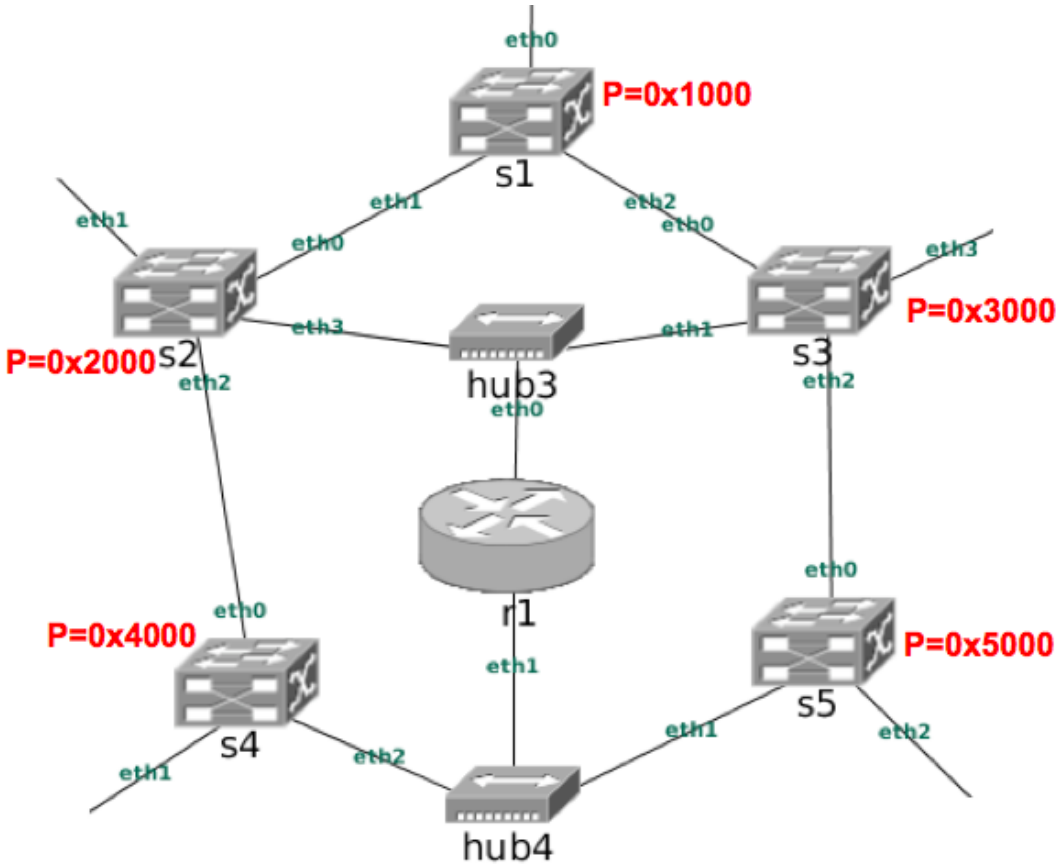
Mostrar el estado del protocolo STP de un *switch* (II)

- **Sección general** del *bridge*:
  - `bridge id`: identificador del *bridge* del que se muestra la información
  - `designated root`: identificador del nodo raíz (de los conocidos por este *bridge*)
  - `root port`: puerto raíz (RP) del *bridge*, 0 si no tiene ninguno (si este nodo es el nodo raíz)
  - `path cost`: coste para alcanzar el nodo raíz desde este *bridge*
- **Sección particular de cada interfaz**:
  - `port id`: identificador de este puerto en este *bridge*
  - `designated root`: nodo raíz anunciado por el *bridge* designado
  - `designated bridge`: *bridge* designado (nodo más cercano al raíz) de entre los que hay conectados a esta interfaz
  - `designated port`: id del puerto del *bridge* designado que es el puerto designado (DP)
  - `designated cost`: coste desde el *bridge* designado al nodo raíz.
  - `state`: estado de ese puerto, los más relevantes son `blocking` o `forwarding`)
  - `path cost`: coste de ese enlace. Los enlaces de 10Mbps (enlaces de NetGUI) tienen coste 100.

Contenidos

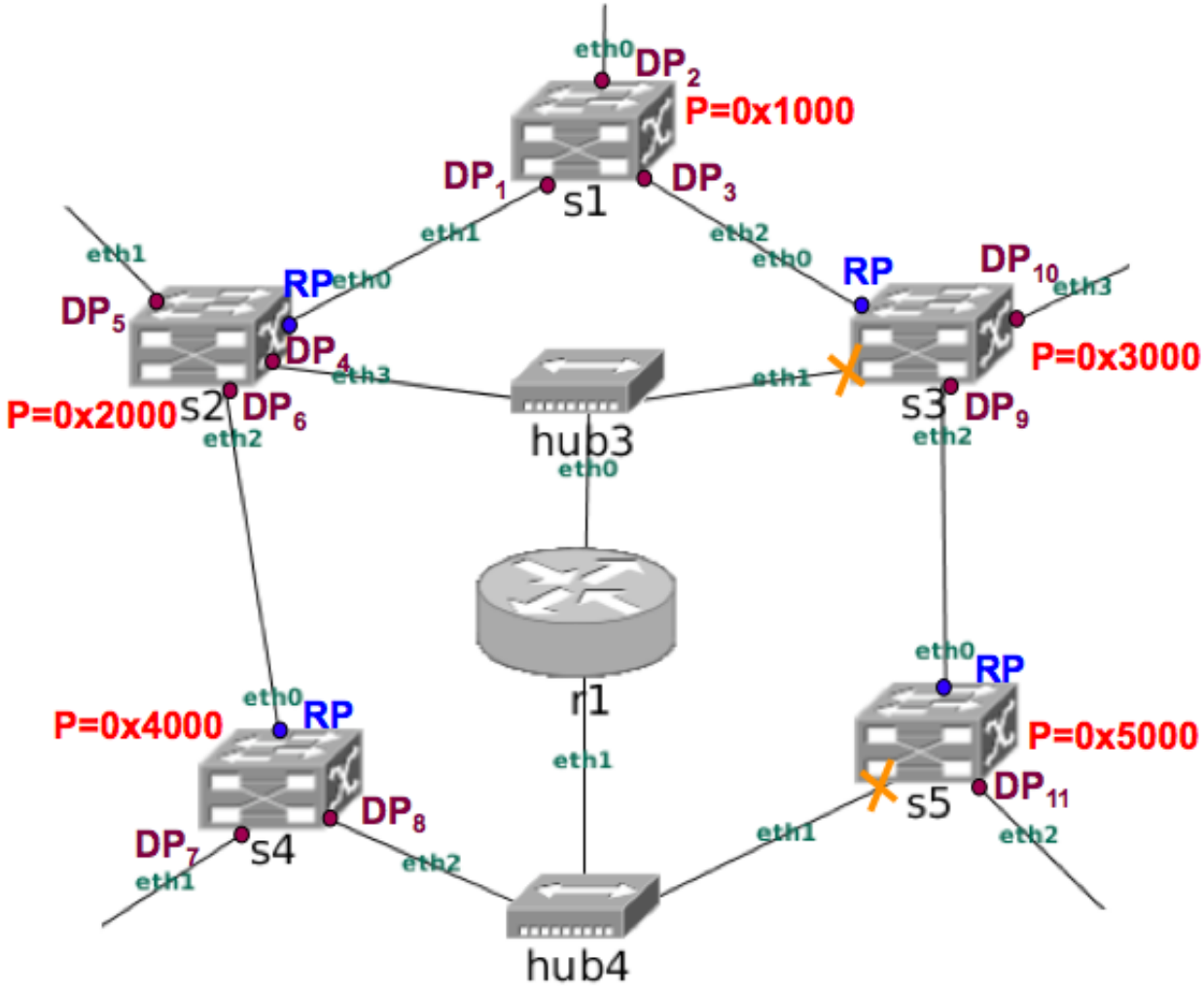
Ejemplo del estado de STP en un *switch* (I)

- En la figura se muestra la conexión de 5 *switches*: `s1`, `s2`, `s3`, `s4` y `s5`.
- Las prioridades configuradas en los *switches* se muestran en rojo.
- Según dichas prioridades, el *switch* raíz del árbol STP es `s1`.



Ejemplo del estado de STP en un *bridge* (II)

- En la figura se muestra la designación de los puertos de cada *switch* como **RP** (Root Port), como **DP** (Designated Port) o como puertos bloqueados **X**. El switch s1 es el switch raíz.



Ejemplo del estado de STP en un *bridge* (III)

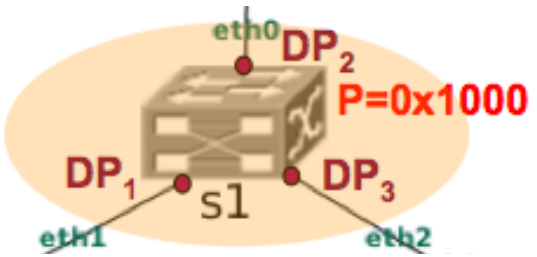
```
s1:~# brctl showstp s1
s1
 bridge id              1000.622ef983c638
 designated root        1000.622ef983c638
 root port                 0                    path cost                  0
 max age                  20.00                 bridge max age            20.00
 hello time                2.00                 bridge hello time          2.00
 forward delay            15.00                 bridge forward delay      15.00
 ageing time             300.00
 hello timer               0.94                 tcn timer                  0.00
 topology change timer     0.00                 gc timer                   3.94
 flags

eth0 (1)
 port id                8001                    state                forwarding
 designated root        1000.622ef983c638       path cost                 100
 designated bridge      1000.622ef983c638       message age timer         0.00
 designated port        8001                    forward delay timer       0.00
 designated cost           0                    hold timer                0.00
 flags

eth1 (2)
 port id                8002                    state                forwarding
 designated root        1000.622ef983c638       path cost                 100
 designated bridge      1000.622ef983c638       message age timer         0.00
 designated port        8002                    forward delay timer       0.00
 designated cost           0                    hold timer                0.00
 flags

eth2 (3)
 port id                8003                    state                forwarding
 designated root        1000.622ef983c638       path cost                 100
 designated bridge      1000.622ef983c638       message age timer         0.00
 designated port        8003                    forward delay timer       0.00
 designated cost           0                    hold timer                0.00
 flags
```

s1 es el switch raíz (`designated root=1000.622ef983c638`).

El número de RP es cero (`root port=0`).

La distancia al nodo raíz es cero (`path cost=0`).

Todas sus interfaces son DP (`designated bridge=1000.622ef983c638`).



Ejemplo del estado de STP en un *bridge* (IV)

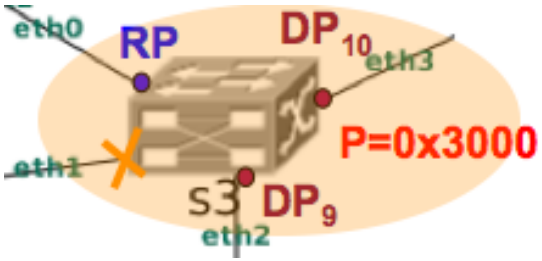
```
s3:~# brctl showstp s3
s3
 bridge id              3000.1e947ee92932
 designated root        1000.622ef983c638
 root port                 1                    path cost                100
 max age                  20.00                 bridge max age           20.00
 hello time                2.00                 bridge hello time         2.00
 forward delay            15.00                 bridge forward delay     15.00
 ageing time             300.00
 hello timer               0.00                 tcn timer                 0.00
 topology change timer     0.00                 gc timer                  9.02
 flags

eth0 (1)
 port id                8001                    state                forwarding
 designated root        1000.622ef983c638       path cost                100
 designated bridge      1000.622ef983c638       message age timer        19.83
 designated port        8003                    forward delay timer       0.00
 designated cost           0                    hold timer                0.00
 flags

eth1 (2)
 port id                8002                    state                blocking
 designated root        1000.622ef983c638       path cost                100
 designated bridge      2000.4ab1cf0c2caf        message age timer        19.82
 designated port        8004                    forward delay timer       0.00
 designated cost         100                    hold timer                0.00
 flags

eth2 (3)
 port id                8003                    state                forwarding
 designated root        1000.622ef983c638       path cost                100
 designated bridge      3000.1e947ee92932        message age timer         0.00
 designated port        8003                    forward delay timer       0.00
 designated cost         100                    hold timer                1.02
 flags

eth3 (4)
 port id                8004                    state                forwarding
 designated root        1000.622ef983c638       path cost                100
 designated bridge      3000.1e947ee92932        message age timer         0.00
 designated port        8004                    forward delay timer       0.00
 designated cost         100                    hold timer                1.02
 flags
```

La interfaz `eth0` de `s3` es **RP** (`root port=1`).

`s3` tiene una distancia al switch raíz de 100 (`path cost=100`), está directamente conectado al raíz.

`eth2` y `eth3` son **DP** (`designated bridge =3000.1e947ee92932`).

`eth1` está bloqueada (`blocking`).



Ejemplo del estado de STP en un *bridge* (V)

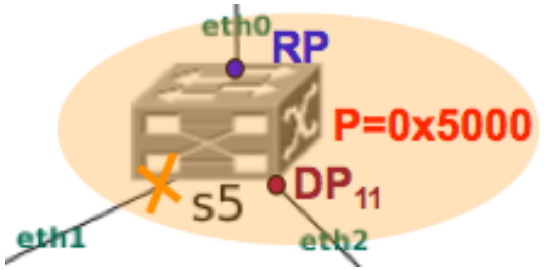
```
s5:~# brctl showstp s5
s5
 bridge id              5000.26c891976e1d
 designated root        1000.622ef983c638
 root port                 1                    path cost                200
 max age                  20.00                 bridge max age           20.00
 hello time                2.00                 bridge hello time         2.00
 forward delay            15.00                 bridge forward delay     15.00
 ageing time             300.00
 hello timer               0.00                 tcn timer                 0.00
 topology change timer     0.00                 gc timer                291.02
 flags

eth0 (1)
 port id                8001                    state                 forwarding
 designated root        1000.622ef983c638       path cost                100
 designated bridge      3000.1e947ee92932       message age timer          18.83
 designated port        8003                    forward delay timer         0.00
 designated cost         100                    hold timer                  0.00
 flags

eth1 (2)
 port id                8002                    state                 blocking
 designated root        1000.622ef983c638       path cost                100
 designated bridge      4000.0ef7e72c8276       message age timer          18.82
 designated port        8003                    forward delay timer         0.00
 designated cost         200                    hold timer                  0.00
 flags

eth2 (3)
 port id                8003                    state                 forwarding
 designated root        1000.622ef983c638       path cost                100
 designated bridge      5000.26c891976e1d       message age timer           0.00
 designated port        8003                    forward delay timer         0.00
 designated cost         200                    hold timer                  0.00
 flags
```

La interfaz eth0 de s5 es **RP** (root port=1).

s5 tiene una distancia al switch raíz de 200 (path cost=200)

eth2 es **DP** (designated bridge =5000.26c891976e1d).

eth1 está bloqueada (blocking).



Tema 3: NetGUI: Configuración de OSPF en Quagga

Sistemas Telemáticos

2º GIT – 2º GITT – 2º GIST

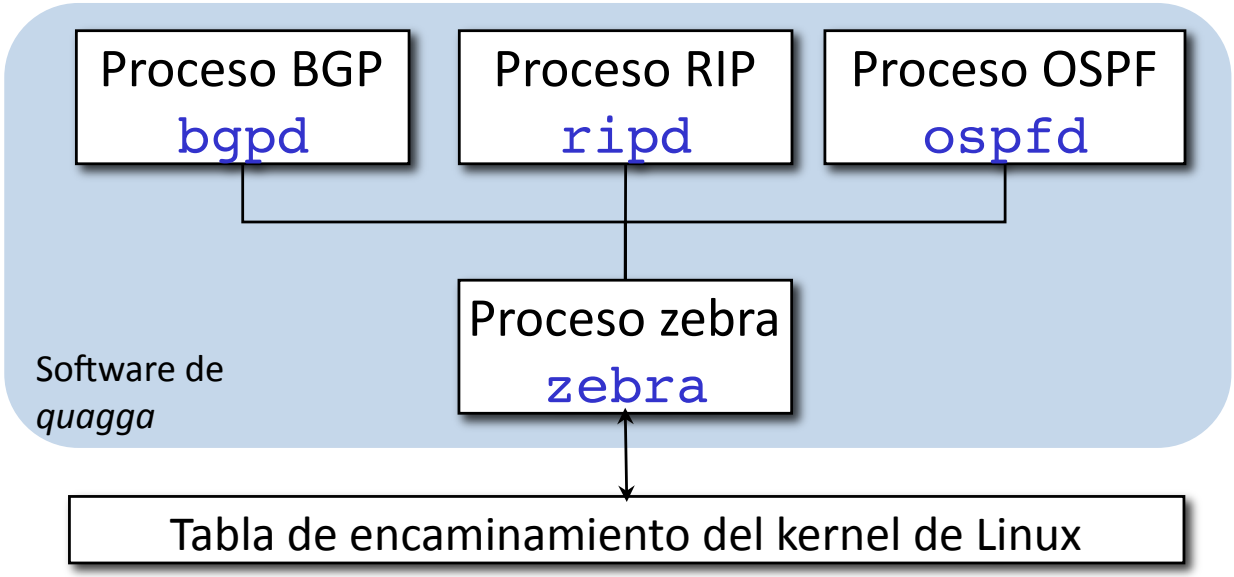
Eva M. Castro Barbero (eva.castro@urjc.es)
José Centeno González (jose.centeno@urjc.es)
Pedro de las Heras Quirós (pedro.delasheras@urjc.es)

Diciembre 2023

Contenidos

Quagga

- Quagga (www.quagga.net) es un software que gestiona la tabla de encaminamiento de una máquina Linux según el funcionamiento de varios protocolos de encaminamiento de la arquitectura TCP/IP.
- La arquitectura de Quagga está formada por un conjunto de procesos:
  - Proceso `zebra`: actualiza la tabla de encaminamiento e intercambia rutas según diferentes protocolos de encaminamiento
  - Proceso de cada protocolo de encaminamiento: `ripd`, `opsfd`, `bgpd`
- Utilizaremos Quagga para probar los protocolos: OSPFv2 y BGP-4.



Configuración y monitorización de los procesos de Quagga

- Configuración a través de los ficheros que se encuentran en la carpeta `/etc/quagga`:
  - `daemons` (ver pág. 7)
  - `ospfd.conf` (ver pág. 9)
- Monitorización a través de:
  - capturas de tráfico, utilizando `tcpdump` con la opción `-s 0` que permite capturar los paquetes completos.
  - Shell VTY (Virtual Terminal Interface): `vtysh` (págs. 14–28) La Shell VTY se comunica con cada uno de los procesos *quagga* de la máquina y permite configurar los protocolos de encaminamiento y monitorizar su comportamiento.

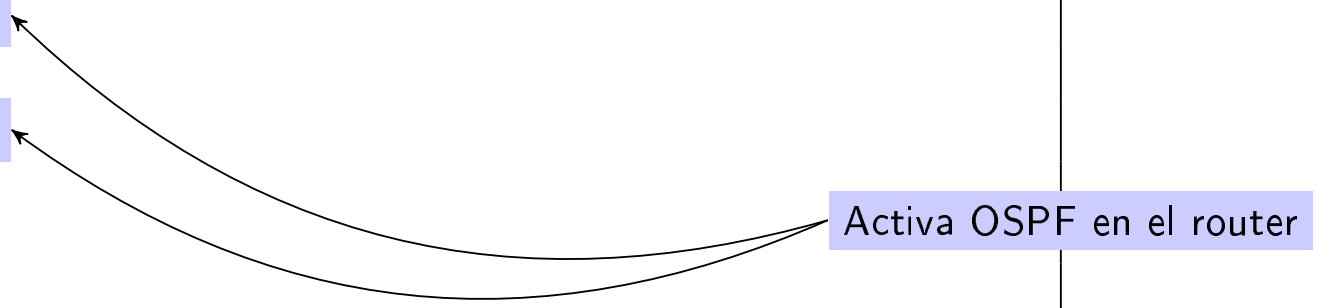
Contenidos

daemons

- Contiene el nombre de los procesos de encaminamiento que se desean activar.
- Para editarlo y activar OSPF podemos usar `mcedit` o `nano`:

```
mcedit /etc/quagga/daemons
nano /etc/quagga/daemons
```

```
# ...
# Entries are in the format:   <daemon>=(yes|no|priority)
# ...
# ...
# /usr/doc/quagga/README.Debian for details.
# Daemons are: bgpd quagga ospfd ospf6d ripd ripngd isisd
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
```



Las líneas que comienzan por # son comentarios.

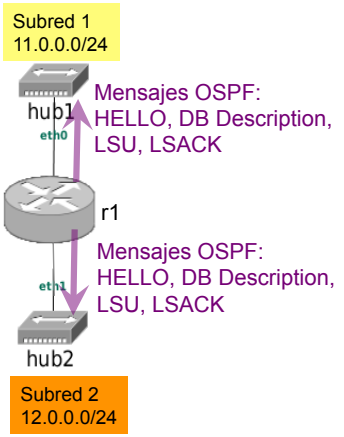
ospfd.conf, todas las interfaces en la misma área

- Contiene la configuración propia de OSPF.
- Para editarlo podemos usar `mcedit` o `nano`:

```
mcedit /etc/quagga/ospfd.conf
nano /etc/quagga/ospfd.conf
```

```
! -*- ospf -*-
!
! OSPFd sample configuration file
!
hostname ospfd
password zebra

router ospf
  router-id 12.0.0.1
  network 11.0.0.0/24 area 0
  network 12.0.0.0/24 area 0
```

Asignamos como ID del router la mayor de sus IPs por las que se activará OSPF

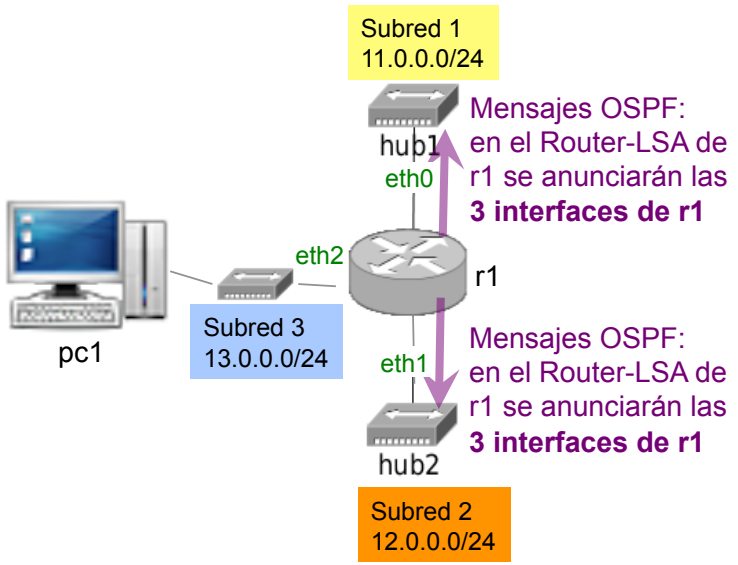


Activar OSPF en las interfaces conectadas a estas redes, pertenecientes al área 0: a través de `eth0` y `eth1` se anunciarán las rutas utilizando OSPF. Hay que especificar a qué área pertenece cada interfaz del *router* por la que se activa OSPF.

Las líneas que comienzan por ! son comentarios.

ospfd.conf: interfaces pasivas

```
! -*- ospf -*-
!
! OSPFd sample configuration file
!
hostname ospfd
password zebra

router ospf
router-id 13.0.0.1
passive-interface eth2
network 11.0.0.0/24 area 0
network 12.0.0.0/24 area 0
network 13.0.0.0/24 area 0
```

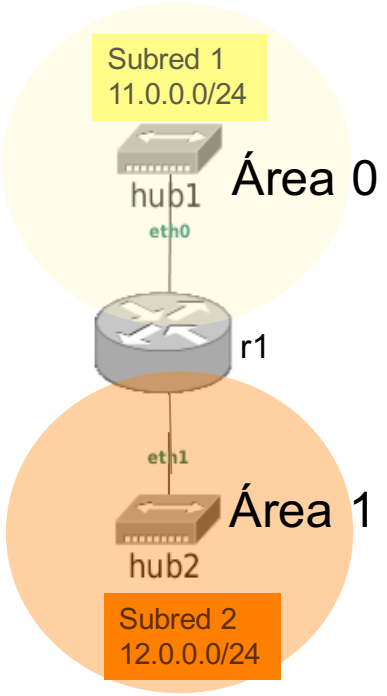


La interfaz `eth0` es pasiva, no se envían mensajes OSPF a través de ella

Es necesario incluir la subred de la interfaz pasiva para que se incluya la información de dicha interfaz en el Router-LSA

ospfd.conf, interfaces en diferentes áreas

```
! -*- ospf -*-
!
! OSPFd sample configuration file
!
hostname ospfd
password zebra

router ospf
 router-id 12.0.0.1
 network 11.0.0.0/24 area 0
 network 12.0.0.0/24 area 1
```



Contenidos

Iniciar Quagga

- Al iniciar un *router* en NetGUI normalmente el software de quagga no estará arrancado. Para realizar una configuración:
  1. Se editan los ficheros de configuración
  2. Se arranca quagga:
     `/etc/init.d/quagga start`
  3. Se realiza la monitorización.
  4. Si es necesario modificar la configuración:
     - se interrumpe la ejecución de quagga:
       `/etc/init.d/quagga stop`
     - se modifican los ficheros
     - se vuelve a arrancar quagga:
       `/etc/init.d/quagga start`
- En algunos escenarios de NetGUI puede que algunos *routers* estén preconfigurados para que arranquen con quagga ya lanzado.

Contenidos

vtysh

```
r1:~# vtysh
Copyright 1996-2005 Kunihiro Ishiguro, et al.

 r1# ?
  clear          Reset functions
  configure      Configuration from vty interface
  copy           Copy from one file to another
  debug          Debugging functions (see also 'undebug')
  disable                Turn off privileged mode command
  end                    End current mode and change to enable mode
  exit                   Exit current mde an down to previous mode
  list                   Print command list
  no                     Negate a command or set its defaults
  ping                   Send echo messages
  quit                           Exit current mode and down to previous mode
  show                   Show running system information
  ssh                            Open an ssh connection
  start-shell            Start UNIX shell
  telnet                 Open a telnet connection
  terminal               Set terminal line parameters
  traceroute             Trace route to destination
  undebug                Disable debugging functions (see also 'debug')
  write                  Write running configuration to memory, network, or terminal
 r1#
```

Contenidos

Tabla de encaminamiento OSPF si sólo hay 1 área

- El comando `show ip ospf route` muestra la información sobre la tabla de encaminamiento OSPF del *router* (el ejemplo muestra la configuración del *router* `r1` de la figura):

```
r1# show ip ospf route
========== OSPF network routing table =========
N    11.0.0.0/24          [10] area: 0.0.0.0
                          directly attached to eth0
N    12.0.0.0/24           [10] area: 0.0.0.0
                          directly attached to eth1
N    13.0.0.0/24           [20] area: 0.0.0.0
                          via 12.0.0.2, eth1
N    14.0.0.0/24           [30] area: 0.0.0.0
                          via 12.0.0.2, eth1

========== OSPF router routing table ==========
========== OSPF external routing table ========
```

El coste de un enlace en OSPF con quagga tiene como valor por defecto 10

Rutas aprendidas por OSPF

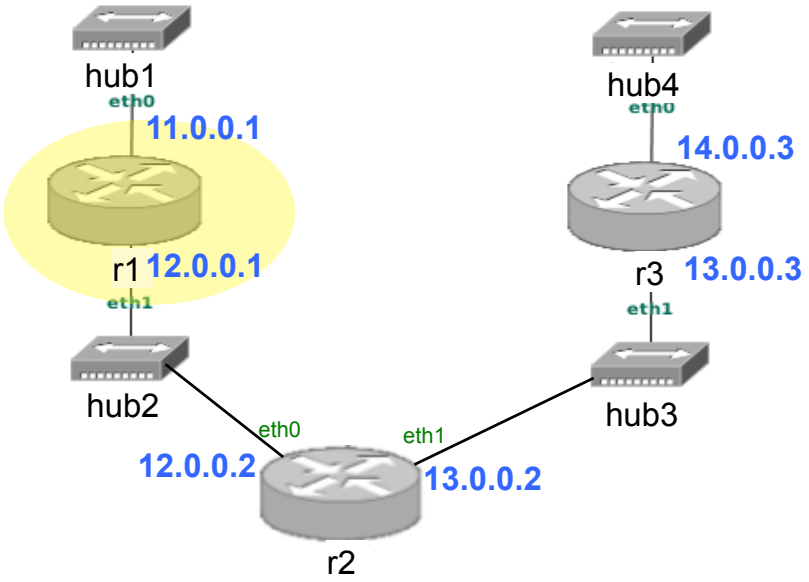


Tabla de encaminamiento OSPF si hay varias áreas

- Si la red tiene diferentes áreas, el comando `show ip ospf route` muestra información adicional:

```
r1#  show ip ospf route
============ OSPF network routing table ============
N    15.0.0.0/24           [10] area: 0.0.0.1
                           directly attached to eth0
N    16.0.0.0/24           [10] area: 0.0.0.1
                           directly attached to eth2
N    17.0.0.0/24           [20] area: 0.0.0.1
                           via 16.0.0.5, eth2
N IA 18.0.0.0/24           [30] area: 0.0.0.1
                           via 16.0.0.5, eth2
N IA 19.0.0.0/24           [40] area: 0.0.0.1
                           via 16.0.0.5, eth2
N IA 20.0.0.0/24           [40] area: 0.0.0.1
                           via 16.0.0.5, eth2

============ OSPF router routing table =============
R    18.0.0.4              [20] area: 0.0.0.1, ABR
                           via 16.0.0.5, eth2

============ OSPF external routing table ===========
```

ruta aprendida en el área 1 (si el router sólo tiene enlaces en el área 1, todas sus rutas están aprendidas en el área 1)

rutas Inter-Área (de otras áreas)

router frontera de área *(Area Border Router)*

Las rutas precedidas por N son rutas hacia una red.
Las rutas precedidas por R son rutas hacia un *router*.

Contenidos

Información de los vecinos OSPF

- El comando `show ip ospf neighbor` muestra la información sobre los vecinos que conoce el *router* (el ejemplo muestra el resultado del comando en el *router* `r2` de la figura):

```
r2#  show ip ospf neighbor
Neighbor ID     Pri   State           Dead Time   Address         Interface
12.0.0.1          1   Full/Backup     00:00:30    12.0.0.1        eth0:12.0.0.2
14.0.0.3          1   Full/DR         00:00:40    13.0.0.3        eth1:13.0.0.2
```

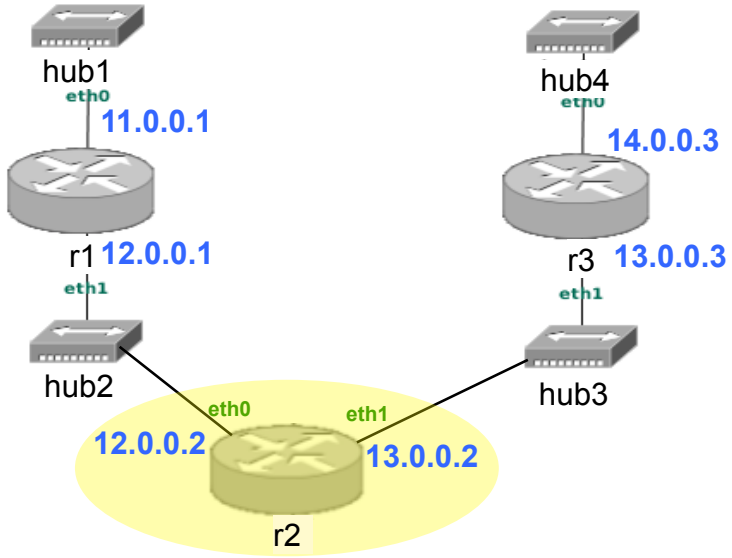
ID del *router* vecino

Indicación de si el vecino es DR o BDR de la subred que les une

Cuenta atrás desde el último `HELLO` recibido del vecino (por defecto 40 segs)

IP del vecino en la red común

IP de este router en la red común



Contenidos

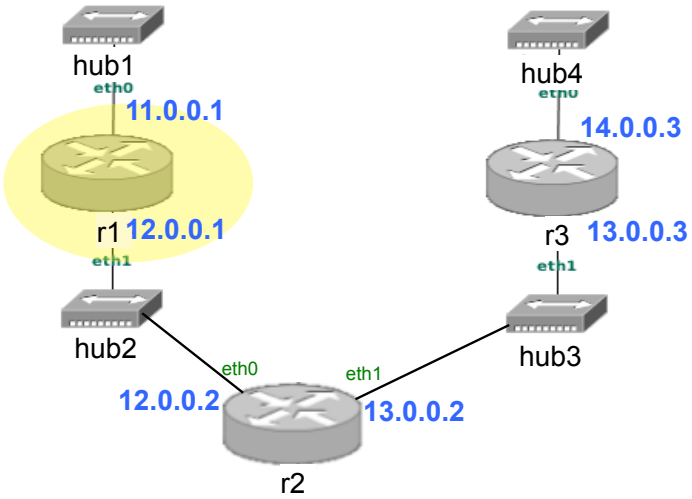
Router Link State DB

- El comando `show ip ospf database router` muestra la información sobre la base de datos de *Router Link States* que conoce el *router* (el ejemplo muestra el resultado del comando en el *router* `r1` de la figura):

```
r1#  show ip ospf database router

       OSPF Router with ID (12.0.0.1)

                Router Link States (Area 0.0.0.0)

  LS age: 1112
  Options: 2
  Flags: 0x0
  LS Type: router-LSA
  Link State ID: 12.0.0.1
  Advertising Router: 12.0.0.1
  LS Seq Number: 80000004
  Checksum: 0x549d
  Length: 48
   Number of Links: 2

    Link connected to: Stub Network
     (Link ID) Net: 11.0.0.0
     (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 12.0.0.2
     (Link Data) Router Interface address: 12.0.0.1
      Number of TOS metrics: 0
       TOS 0 Metric: 10
```

```
  LS age: 1107
  Options: 2
  Flags: 0x0
  LS Type: router-LSA
  Link State ID: 13.0.0.2
  Advertising Router: 13.0.0.2
  LS Seq Number: 80000004
  Checksum: 0x2ab0
  Length: 48
   Number of Links: 2

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 12.0.0.2
     (Link Data) Router Interface address: 12.0.0.2
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 13.0.0.3
     (Link Data) Router Interface address: 13.0.0.2
      Number of TOS metrics: 0
       TOS 0 Metric: 10
```

```
  LS age: 1107
  Options: 2
  Flags: 0x0
  LS Type: router-LSA
  Link State ID: 14.0.0.3
  Advertising Router: 14.0.0.3
  LS Seq Number: 80000003
  Checksum: 0xd210
  Length: 48
   Number of Links: 2

    Link connected to: a Transit Network
     (Link ID) Designated Router address: 13.0.0.3
     (Link Data) Router Interface address: 13.0.0.3
      Number of TOS metrics: 0
       TOS 0 Metric: 10

    Link connected to: Stub Network
     (Link ID) Net: 14.0.0.0
     (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metric: 10
```



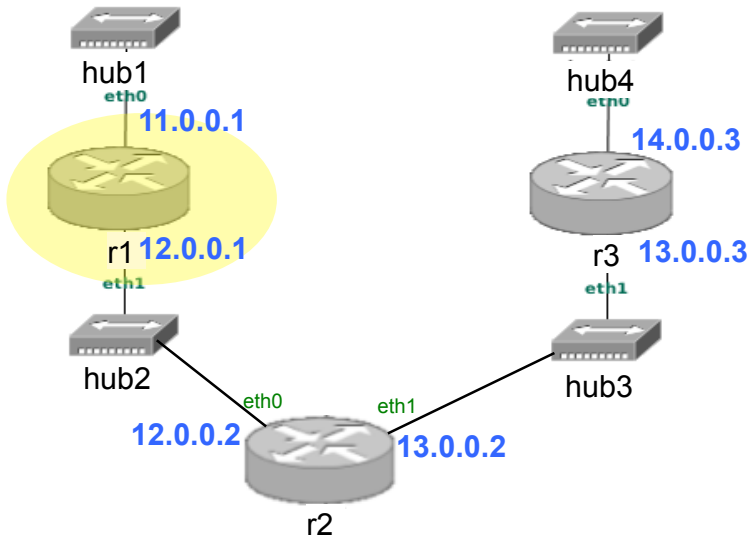
Contenidos

Network Link State DB

- El comando `show ip ospf database network` muestra la información sobre la base de datos de *Network Link States* que conoce el *router* (el ejemplo muestra el resultado del comando en el *router* r1 de la figura):

```
r1#  show ip ospf database network

       OSPF Router with ID (12.0.0.1)

                Net Link States (Area 0.0.0.0)

  LS age: 112
  Options: 2
  LS Type: network-LSA
  Link State ID: 12.0.0.2 (address of Designated Router)
  Advertising Router: 13.0.0.2
  LS Seq Number: 80000002
  Checksum: 0x5bc8
  Length: 32
  Network Mask: /24
        Attached Router: 12.0.0.1
        Attached Router: 13.0.0.2
```

```
  LS age: 105
  Options: 2
  LS Type: network-LSA
  Link State ID: 13.0.0.3 (address of Designated Router)
  Advertising Router: 14.0.0.3
  LS Seq Number: 80000002
  Checksum: 0x5fbc
  Length: 32
  Network Mask: /24
        Attached Router: 13.0.0.2
        Attached Router: 14.0.0.3
```



Contenidos

Summary Link State DB

- Si la red tiene diferentes áreas, el comando `show ip ospf database summary` muestra la información sobre la base de datos de *Summary Link States* que conoce el *router*:

```
r1#  show ip ospf database summary
      OSPF Router with ID (16.0.0.1)


                Summary Link States (Area 0.0.0.1)

  LS age: 592
  Options: 2
  LS Type: summary-LSA
  Link State ID: 18.0.0.0 (summary Network Number)
  Advertising Router: 18.0.0.4
  LS Seq Number: 80000006
  Checksum: 0x0c07
  Length: 28
  Network Mask: /24
        TOS: 0  Metric: 10

  LS age: 580
  Options: 2
  LS Type: summary-LSA
  Link State ID: 19.0.0.0 (summary Network Number)
  Advertising Router: 18.0.0.4
  LS Seq Number: 80000005
  Checksum: 0x63a4
  Length: 28
  Network Mask: /24
        TOS: 0  Metric: 20
```

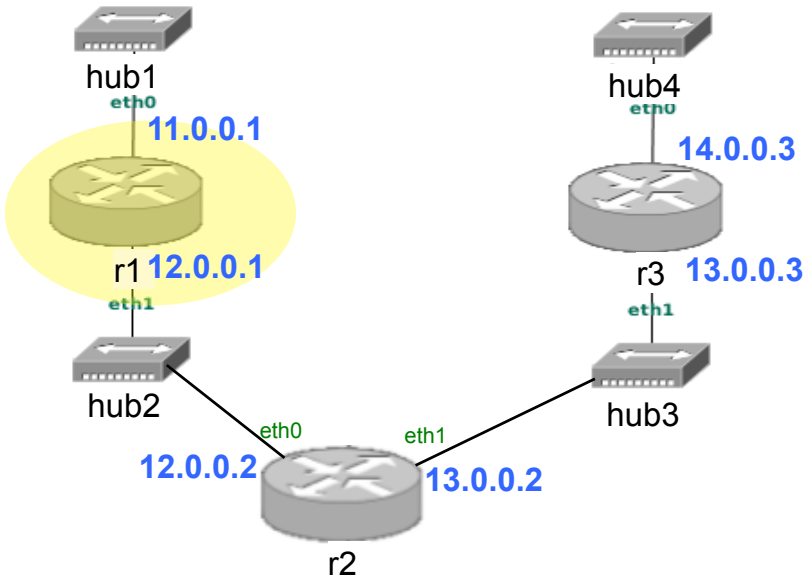
```
  LS age: 588
  Options: 2
  LS Type: summary-LSA
  Link State ID: 20.0.0.0 (summary Network Number)
  Advertising Router: 18.0.0.4
  LS Seq Number: 80000006
  Checksum: 0x56b0
  Length: 28
  Network Mask: /24
        TOS: 0  Metric: 20
```

Contenidos

Resumen de las DBs si sólo hay 1 área

- El comando `show ip ospf database` muestra un resumen de la información sobre las bases de datos del *router* (el ejemplo muestra el resultado del comando en el *router* `r1` de la figura):

```
r1#  show ip ospf database

       OSPF Router with ID (12.0.0.1)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum Link count
12.0.0.1        12.0.0.1        579 0x80000005 0x529e 2
13.0.0.2        13.0.0.2        574 0x80000005 0x28b1 2
14.0.0.3        14.0.0.3        574 0x80000004 0xd011 2

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
12.0.0.2        13.0.0.2        586 0x80000002 0x5bc8
13.0.0.3        14.0.0.3        579 0x80000002 0x5fbc
```



Resumen de las DBs si hay varias áreas

- Si la red tiene diferentes áreas, el comando `show ip ospf database` también muestra la información de los *Summary Link States*:

```
r1#  show ip ospf database

       OSPF Router with ID (12.0.0.1)

                Router Link States (Area 0.0.0.1)

Link ID         ADV Router      Age  Seq#       CkSum Link count
12.0.0.1        12.0.0.1        579 0x80000005 0x529e 2
13.0.0.2        13.0.0.2        574 0x80000005 0x28b1 2
14.0.0.3        14.0.0.3        574 0x80000004 0xd011 2

                Net Link States (Area 0.0.0.1)

Link ID         ADV Router      Age  Seq#       CkSum
12.0.0.2        13.0.0.2        586 0x80000002 0x5bc8
13.0.0.3        14.0.0.3        579 0x80000002  0x5fbc

                Summary Link States (Area 0.0.0.1)

Link ID         ADV Router      Age  Seq#       CkSum  Route
18.0.0.0        18.0.0.4        592 0x80000006 0x0c07 18.0.0.0/24
19.0.0.0        18.0.0.4        580 0x80000005 0x61a5 19.0.0.0/24
20.0.0.0        18.0.0.4        588 0x80000006 0x56b0 20.0.0.0/24
```

Tema 4: NetGUI: Configuración de BGP en Quagga

Sistemas Telemáticos

2º GIT – 2º GITT – 2º GIST

Eva M. Castro Barbero (eva.castro@urjc.es)
José Centeno González (jose.centeno@urjc.es)
Pedro de las Heras Quirós (pedro.delasheras@urjc.es)

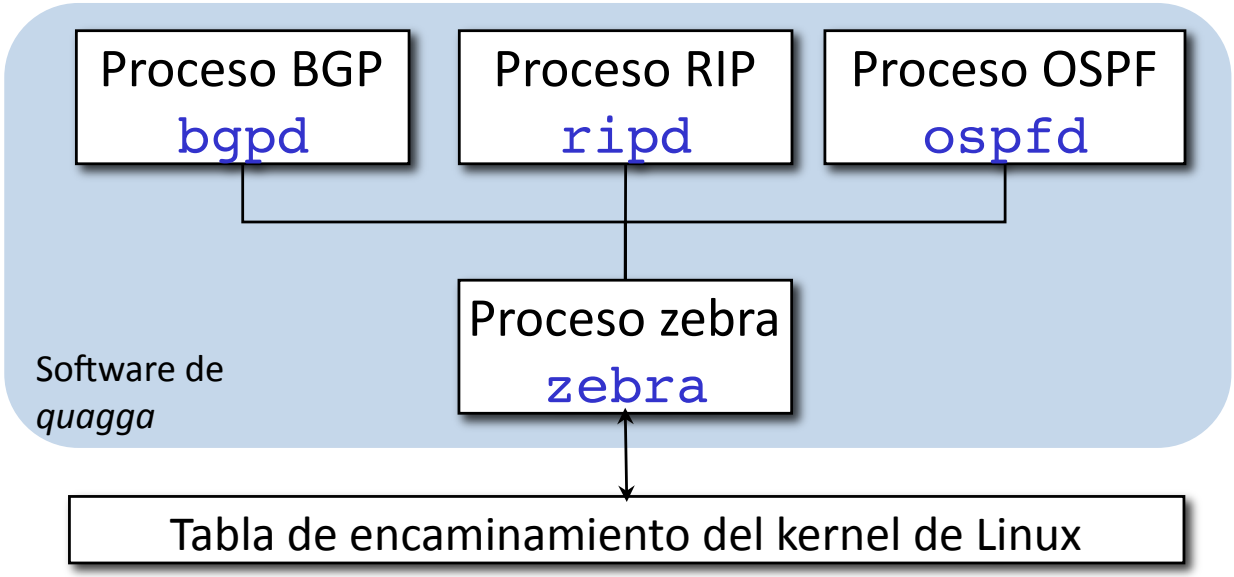
Diciembre 2023

Contenidos

Contenidos

Quagga

- Quagga (www.quagga.net) es un software que gestiona la tabla de encaminamiento de una máquina Linux según el funcionamiento de varios protocolos de encaminamiento de la arquitectura TCP/IP.
- La arquitectura de Quagga está formada por un conjunto de procesos:
  - Proceso `zebra`: actualiza la tabla de encaminamiento e intercambia rutas según diferentes protocolos de encaminamiento
  - Proceso de cada protocolo de encaminamiento: `ripd`, `opsfd`, `bgpd`
- Utilizaremos Quagga para probar los protocolos: OSPFv2 y BGP-4.



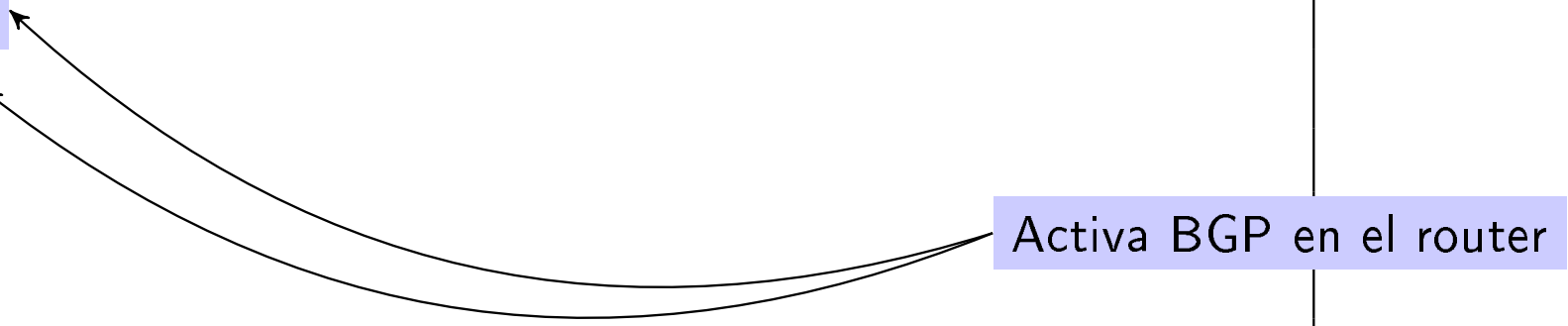
Configuración y monitorización de los procesos de Quagga

- Configuración a través de los ficheros:
  - `daemons` (ver pág. 8)
  - `bgpd.conf` (ver pág. 9)
- Monitorización a través de:
  - capturas de tráfico, utilizando `tcpdump` con la opción `-s 0` que permite capturar los paquetes completos.
  - Shell VTY (Virtual Terminal Interface): `vtysh` (págs. **??–??**) La Shell VTY se comunica con cada uno de los procesos *quagga* de la máquina y permite configurar los protocolos de encaminamiento y monitorizar su comportamiento.

Contenidos

daemons

```
# ...
# Entries are in the format:   <daemon>=(yes|no|priority)
# ...
# ...
# /usr/doc/quagga/README.Debian for details.
# Daemons are: bgpd quagga ospfd ospf6d ripd ripngd isisd
zebra=yes
bgpd=yes
ospfd=no
ospf6d=no
ripd=no
ripngd=no
```



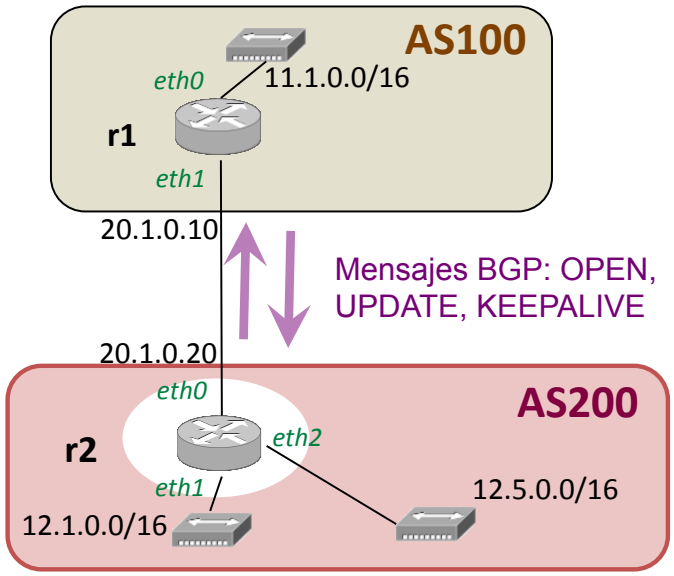
Las líneas que comienzan por # son comentarios.

bgpd.conf

Configuración en r2:

```
! -*- bgpf -*-
!
! BGPd sample configuration file
!
hostname bgpd
password zebra

router bgp 200
bgp router-id 20.1.0.20

neighbor 20.1.0.10 remote-as 100

redistribute connected
```

Activa BGP con identificador de AS=200



Dirección IP del *router* vecino (no es el identificador), indicando también el número de AS al que se enviarán anuncios

Identificador del *router* BGP que estoy configurando (r2). Es costumbre que sea su IP más alta

Se anunciarán por BGP las redes a las que el *router* está directamente conectado (en este caso: 20.1.0.0/16, 12.1.0.0/16, 12.5.0.0/16).
Esta configuración es equivalente a haber escrito:

```
network 20.1.0.0/16
network 12.1.0.0/16
network 12.5.0.0/16
```

Las líneas que comienzan por ! son comentarios.

bgpd.conf: Agregación de Rutas

- Utilizando CIDR pueden agruparse varias redes bajo un solo prefijo para optimizar el número de entradas en las tablas de los *routers*. De esta forma un router en vez de anunciar varias subredes de forma independiente, podrá anunciar el prefijo que las agrupa.
- En el fichero `bgpd.conf` se incluirá el comando:

  `aggregate-address a.b.c.d/prefix summary-only`

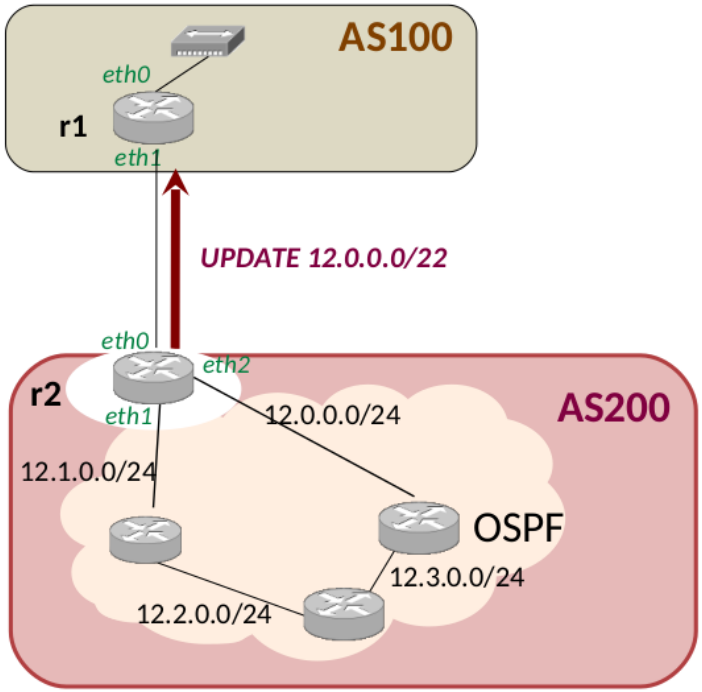
  - si una red a anunciar se encuentra incluida en `a.b.c.d/prefix`, se anunciará `a.b.c.d/prefix` en vez de dicha red.

bgpd.conf: Ejemplo de agregación de Rutas

Configuración en r2:

```
...
router bgp 200
...
  aggregate-address 12.0.0.0/22 summary-only
...
```

En vez de anunciar las subredes 12.0.0.0/24, 12.1.0.0/24, 12.2.0.0/24, 12.3.0.0/24, se anuncia un único prefijo: 12.0.0.0/22



Contenidos

Iniciar Quagga

- Al iniciar un *router* en NetGUI normalmente el software de quagga no estará arrancado. Para realizar una configuración:
  1. Se editan los ficheros de configuración
  2. Se arranca quagga:
     `/etc/init.d/quagga start`
  3. Se realiza la monitorización.
  4. Si es necesario modificar la configuración:
     - se interrumpe la ejecución de quagga:
       `/etc/init.d/quagga stop`
     - se modifican los ficheros
     - se vuelve a arrancar quagga:
       `/etc/init.d/quagga start`
- En algunos escenarios de NetGUI puede que algunos *routers* estén preconfigurados para que arranquen con quagga ya lanzado.

Contenidos

vtysh

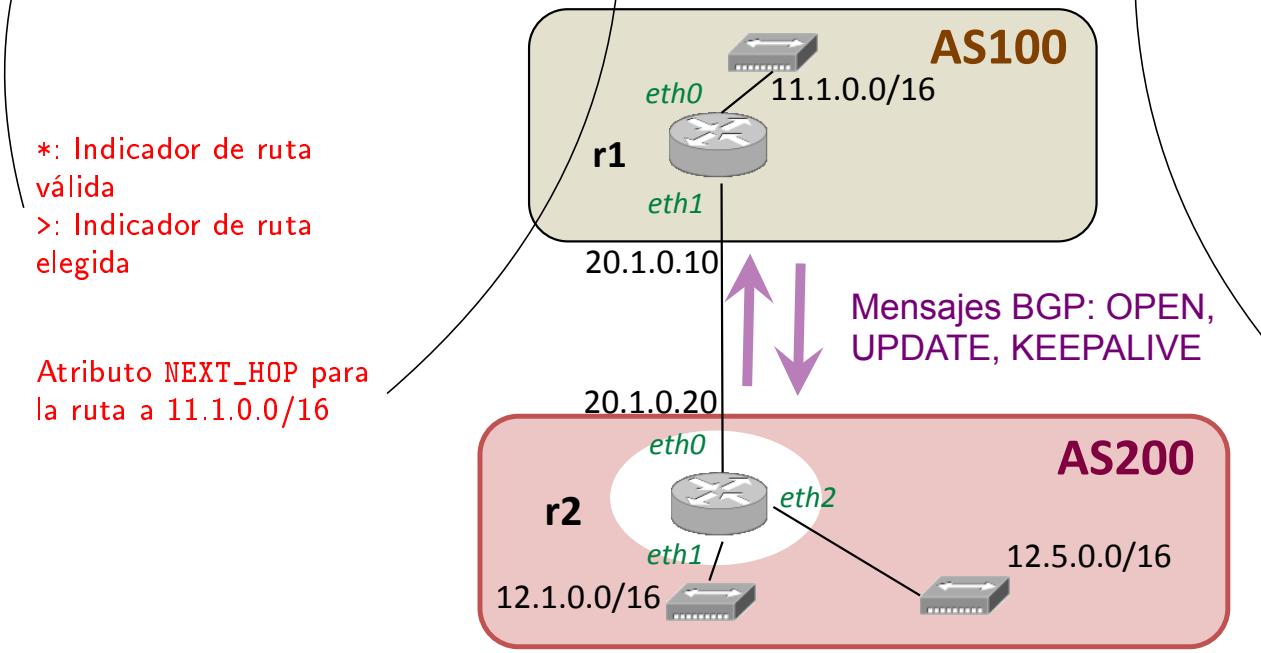
```
r1:~# vtysh
Copyright 1996-2005 Kunihiro Ishiguro, et al.

 r1# ?
  clear          Reset functions
  configure      Configuration from vty interface
  copy           Copy from one file to another
  debug          Debugging functions (see also 'undebug')
  disable                Turn off privileged mode command
  end                    End current mode and change to enable mode
  exit                   Exit current mde an down to previous mode
  list                   Print command list
  no                     Negate a command or set its defaults
  ping                   Send echo messages
  quit                           Exit current mode and down to previous mode
  show                   Show running system information
  ssh                            Open an ssh connection
  start-shell            Start UNIX shell
  telnet                 Open a telnet connection
  terminal               Set terminal line parameters
  traceroute             Trace route to destination
  undebug                Disable debugging functions (see also 'debug')
  write                  Write running configuration to memory, network, or terminal
 r1#
```

Contenidos

Tabla de encaminamiento BGP

- El comando `show ip bgp` muestra la información sobre la tabla de encaminamiento BGP del *router* (el ejemplo muestra la configuración del *router* r2 de la figura):

```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric  LocPrf  Weight  Path
*> 11.1.0.0/16      20.1.0.10                0                  100  i
 *>20.1.0.0/16      0.0.0.0                  0               32768  i
 *>12.1.0.0/16      0.0.0.0                  0               32768  i
 *>12.5.0.0/16      0.0.0.0                  0               32768  i
```



*: Indicador de ruta válida
>: Indicador de ruta elegida

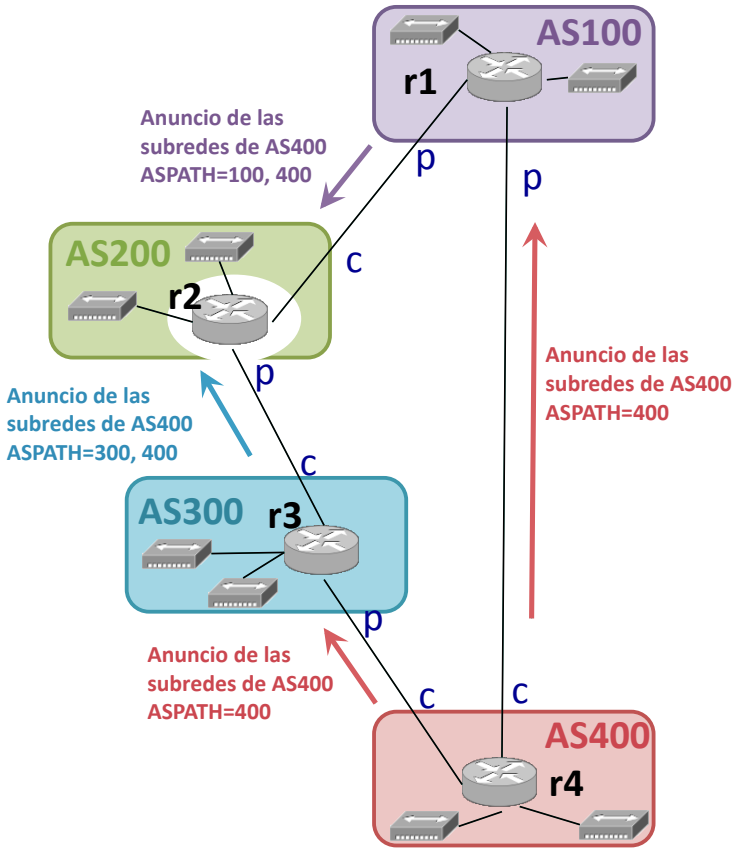
Atributo NEXT_HOP para la ruta a 11.1.0.0/16

Atributo ORIGIN indica si las subredes se han aprendido con:
i: líneas network, agregación de direcciones o rutas por defecto en bgpd.conf
e: otro protocolo externo
?: líneas redistribute en bgpd.conf

Atributo AS_PATH para la ruta a 11.1.0.0/16 que ha aprendido a través de r1.
No hay AS_PATH para las rutas internas de AS200 porque no me han llegado en anuncios BGP

AS_PATH en la tabla de encaminamiento BGP

- En la tabla BGP se muestra el valor AS_PATH asociado a cada ruta.
- Cada AS añade su número de sistema autónomo al atributo AS_PATH antes de anunciar una ruta. Para las subredes de AS400, 16.0.0.0/16:
  - AS300 añade el identificador 300 antes de exportar dichas subredes a AS200. Por tanto, AS200 recibe de AS300: AS_PATH=300 400.
  - AS100 añade el identificador 100 antes de exportar dichas subredes a AS200. Por tanto, AS200 recibe de AS100: AS_PATH=100 400.




```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric  LocPrf  Weight  Path
*>16.0.0.0/16       20.2.0.30                                0    300 400 i
*                   20.1.0.10                                0    100 400 i
```

Configuración BGP en r2.

Contenidos

Redistribución de rutas entre OSPF y BGP

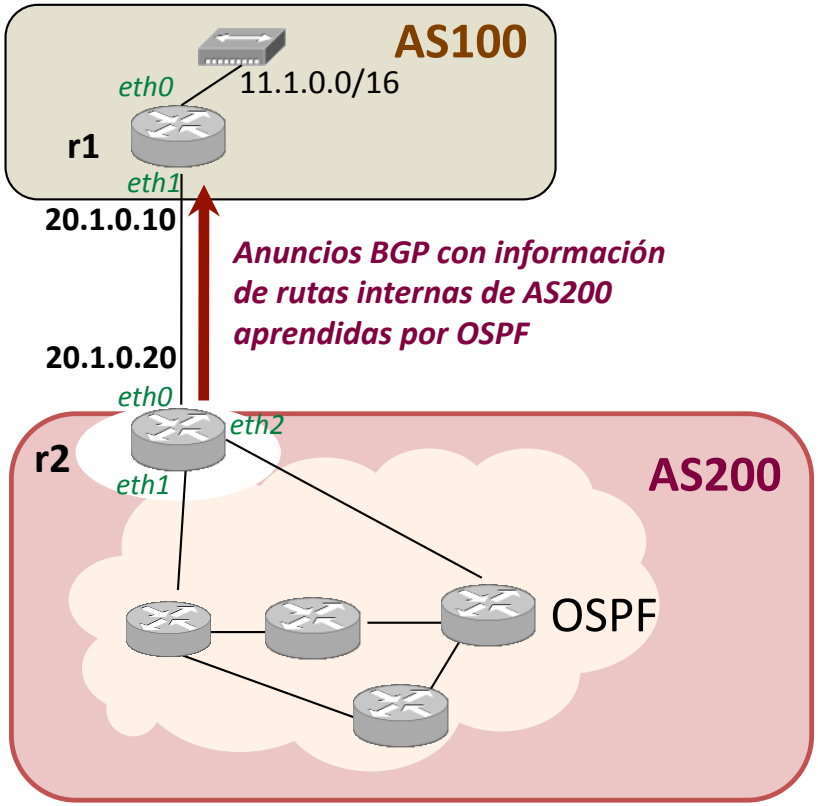
- Un router puede ejecutar varios protocolos de encaminamiento diferentes.
- Así, por ejemplo, un router frontera de un AS ejecutará tanto BGP como un protocolo interior (RIP u OSPF).
- Para que las rutas aprendidas por OSPF se propaguen hacia el exterior anunciándose a través de BGP es necesario configurarlo explícitamente **en el fichero** `bgpd.conf`.
- Para que las rutas aprendidas por BGP se propaguen internamente utilizando OSPF es necesario configurarlo explícitamente **en el fichero** `ospfd.conf`.

Redistribución entre OSPF y BGP: Rutas aprendidas por OSPF se envían en anuncios BGP

- Si AS200 está ejecutando OSPF entre todos sus routers internos, el router frontera r2 debe estar ejecutando tanto BGP como OSPF. El fichero `daemons` tendrá activado: `zebra`, `ospfd` y `bgpd`.

- Los ficheros `ospfd.conf` de cada uno de los routers de AS200 estarán configurados adecuadamente para que se anuncien por OSPF las rutas interiores de AS200.
- Para que las rutas aprendidas por r2 a través de OSPF se anuncien por BGP es necesario añadir la siguiente línea en el fichero `bgpd.conf`:

```
...
router bgp 200
...
  redistribute ospf
...
```



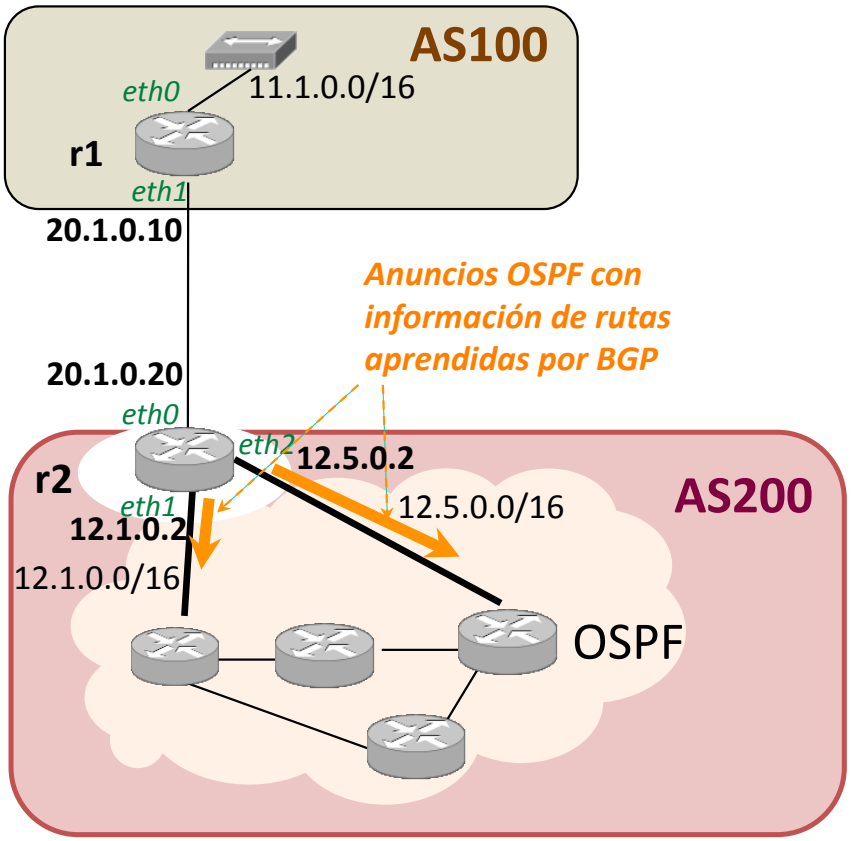
Redistribución entre OSPF y BGP: Rutas aprendidas por BGP se envían en anuncios OSPF

- Para que los routers internos de AS200 puedan alcanzar los destinos de AS100, r2 puede redistribuir la información que ha aprendido por BGP utilizando OSPF. Para ello, el fichero `ospfd.conf` de r2 debe incluir la siguiente línea:

```
...
router ospf
...
  redistribute bgp
...
```

- r2 NO tiene en su fichero de configuración `ospfd.conf` la línea `network 20.1.0.0/16`, ya que en esa subred no hay otros routers OSPF.
- Para que se anuncie la subred 20.1.0.0/16 dentro del AS es necesario añadir en `ospfd.conf` la línea `redistribute connected`.
- Así, en r2 el fichero `ospfd.conf` quedaría:

```
...
router ospf
  network 12.1.0.0/16 area 0
  network 12.5.0.0/16 area 0
  redistribute connected
  redistribute bgp
...
```



Contenidos

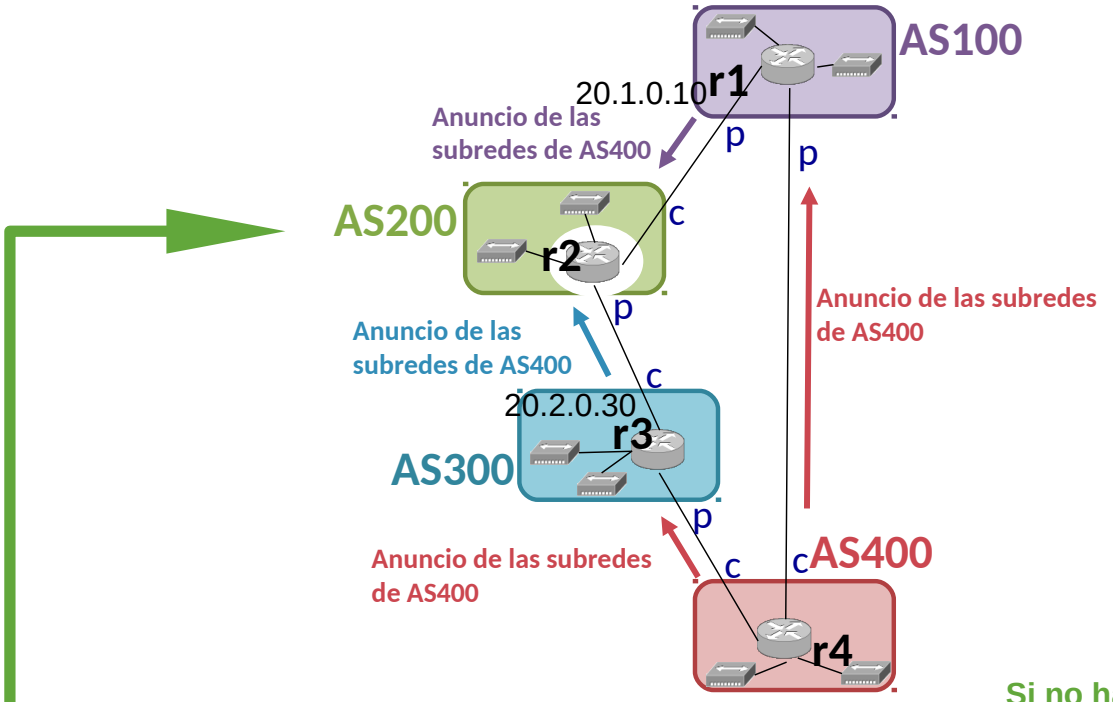
Selección de la mejor ruta

- Por defecto, todas las subredes recibidas en anuncios BGP se incorporan a la tabla BGP. Por tanto, en esta tabla se muestran todas las posibilidades que conoce un router para alcanzar cada una de las subredes.
- Si existe más de una posibilidad de ruta para alcanzar una determinada subred, BGP seleccionará cuál será la ruta preferida atendiendo a los siguientes criterios (sólo los criterios marcados en negrita son los que tendremos en cuenta al realizar las prácticas):
  1. Si el siguiente salto que debe utilizarse para alcanzar una subred es inaccesible, se descarta la ruta.
  2. Se prefiere ruta con mayor valor de `Weight` (parámetro configurable en Cisco y quagga).
  3. **Se prefiere ruta con mayor valor de atributo `LOCAL_PREF`.**
  4. Se prefiere ruta generada localmente en el fichero de configuración de BGP.
  5. **Se prefiere ruta con el atributo `AS_PATH` más corto.**
  6. Se prefiere ruta en función de atributo `ORIGIN`: IGP mejor que EGP y mejor que `INCOMPLETE`.
  7. Se prefiere ruta con menor atributo Multi-Exit Discriminator (`MED`).
  8. ...
- Una vez seleccionadas las rutas preferidas:
  - Las rutas preferidas se incorporan a la tabla de encaminamiento de la máquina.
  - Las rutas preferidas se anuncian (si es conveniente) a los ASs vecinos.

Contenidos

Varias rutas para el mismo destino

- Cuando un *router* recibe diferentes rutas para alcanzar un mismo destino, **INCLUYE todas ellas en su tabla BGP**.




```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
Network          Next Hop    Metric     LocPrf     Weight       Path
* 16.0.0.0/16    20.2.0.30                                     300 400 i
*                20.1.0.10                                     100 400 i
```

Si no hay valor, se usa el valor por defecto =100

El atributo LOCAL_PREF

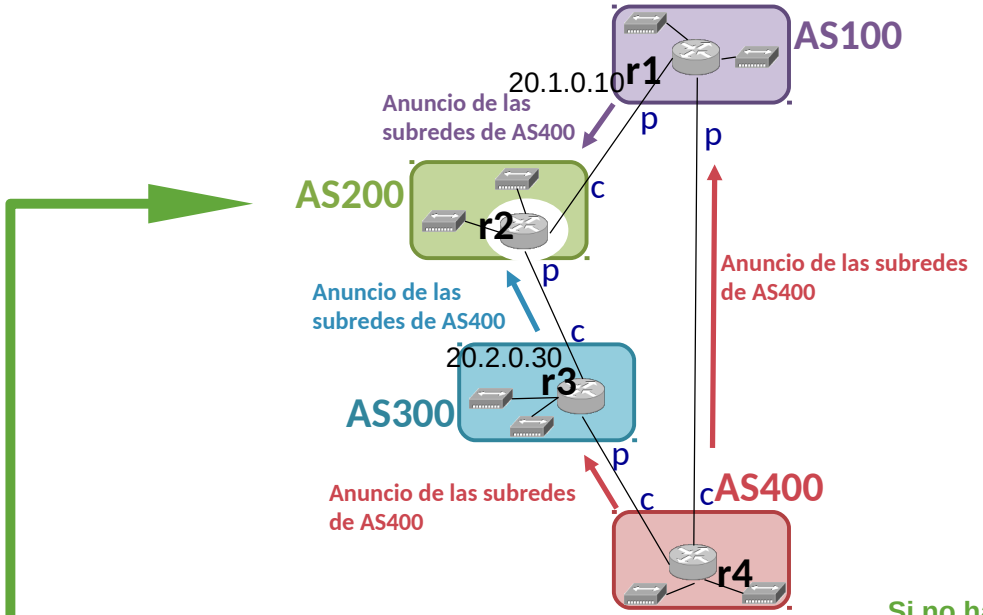
- Los administradores deben configurar el atributo LOCAL_PREF con cada vecino para que se adecúe a las relaciones con los ASs vecinos y se utilice esta información para la **selección de la ruta preferida**. En un AS se configurará:
  - Mayor LOCAL_PREF para un AS vecino que sea su cliente.
  - Menor LOCAL_PREF para un AS vecino que sea su proveedor.
  - Valor intermedio de LOCAL_PREF para un AS vecino que sea entre iguales.

- En caso de empate en LOCAL_PREF, la ruta preferida se elegirá por el AS_PATH más corto.

- **LOCAL_PREF es un atributo que sólo tiene sentido dentro de un AS y no se propaga fuera del mismo. No viaja en los anuncios UPDATE.**

Necesidad de configuración de LOCAL_PREF

- El administrador debe configurar localmente el atributo LOCAL_PREF para forzar la selección de la ruta preferida.
- r2 debería elegir la ruta hacia AS400 a través de AS300 (su cliente). Deberá expresarlo con atributo LOCAL_PREF mayor hacia AS300.



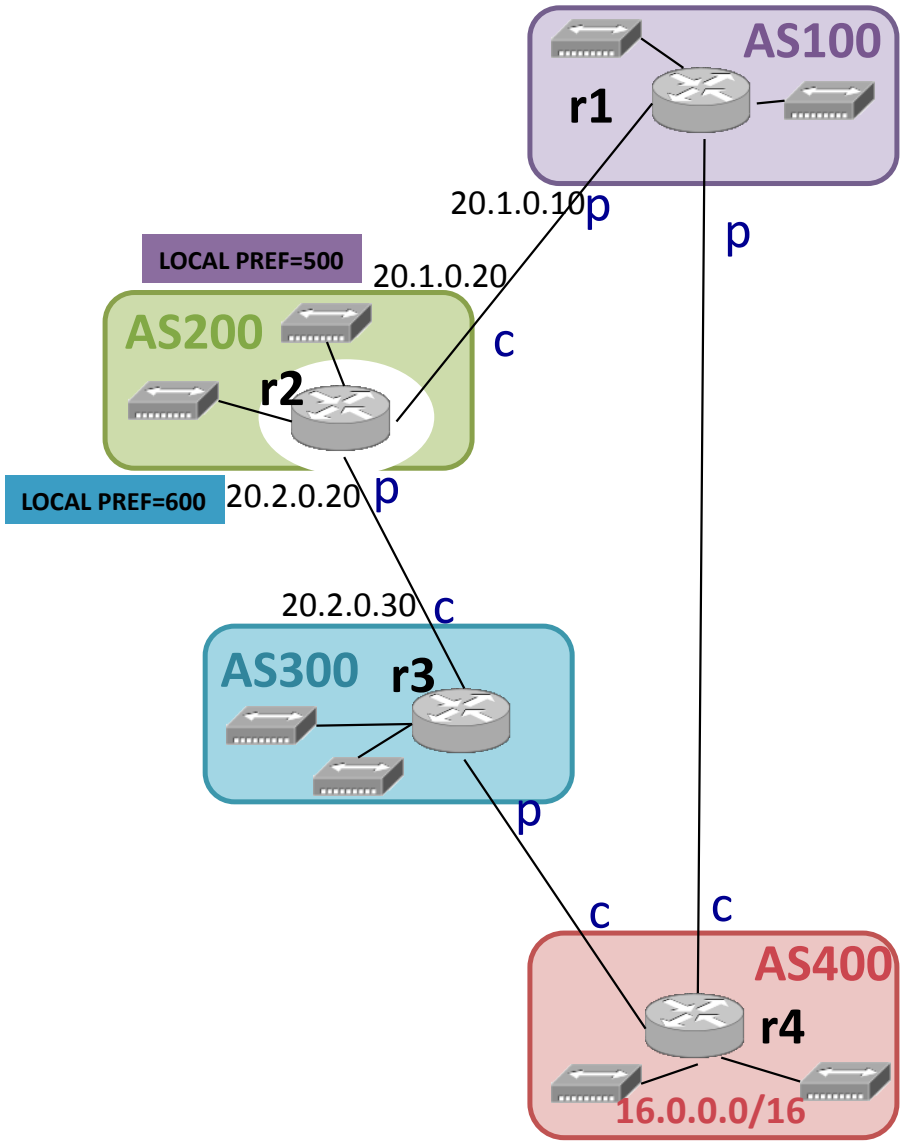

```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
Network          Next Hop    Metric     LocPrf     Weight       Path
* 16.0.0.0/16    20.2.0.30                                      300 400 i
*                20.1.0.10                                      100 400 i
```

Si no hay valor, se usa el valor por defecto =100

Configuración del atributo LOCAL PREF (I)

- Para que r2 seleccione la ruta hacia AS400 a través de AS300, en r2 el atributo LOCAL PREF del vecino de AS300 debe ser mayor que el LOCAL PREF del vecino de AS100.

```
router bgp 200
  neighbor 20.1.0.10 remote-as 100
  neighbor 20.2.0.30 remote-as 300


  neighbor 20.1.0.10 route-map confLocalPrefAS100 in
  neighbor 20.2.0.30 route-map confLocalPrefAS300 in

  redistribute ...
  redistribute ...

  aggregate-address ...
  aggregate-address ...

  route-map confLocalPrefAS100 permit 10
     set local-preference 500


  route-map confLocalPrefAS300 permit 10
     set local-preference 600
```

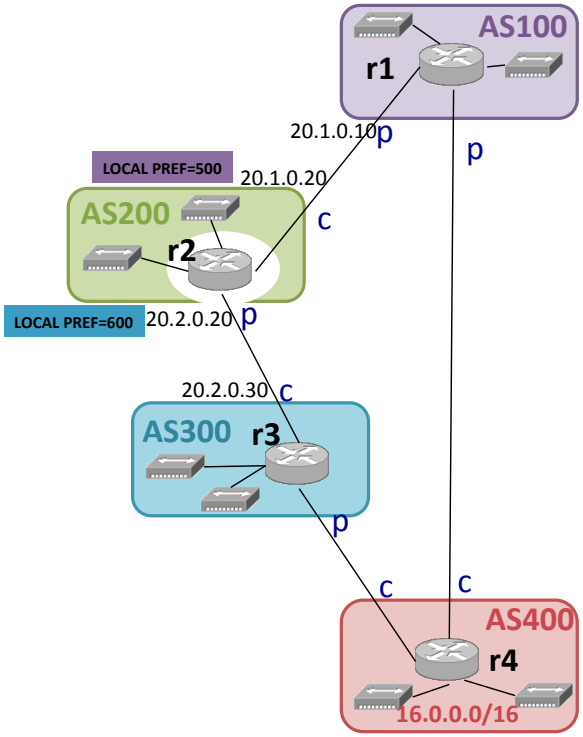


Configuración del atributo LOCAL PREF (II)

- En la tabla BGP se muestra el valor LOCAL PREF asociado a cada ruta.
- En r2 se puede consultar la tabla BGP para ver los atributos LOCAL PREF asignados a las rutas aprendidas.

```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric  LocPrf  Weight  Path
*>  16.0.0.0/16      20.2.0.30                    600       0    300 400 i
 *                   20.1.0.10                    500       0    100 400 i
```

Selección de la mejor ruta



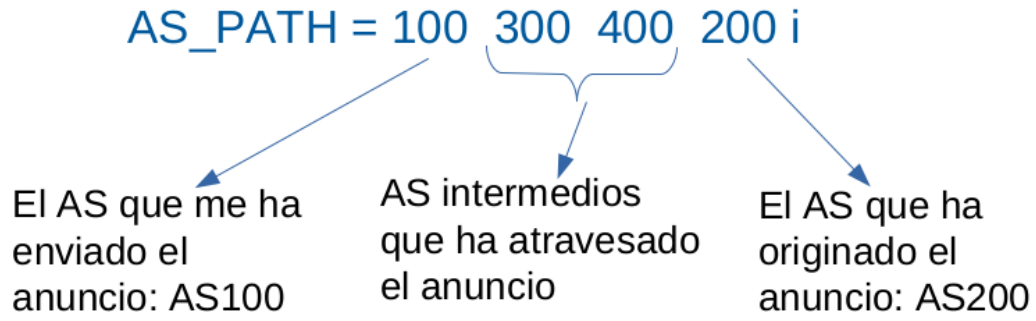
Contenidos

Exportación de rutas

- Aunque un AS puede tener en su tabla BGP varias rutas para alcanzar un destino, el AS exporta **únicamente la ruta preferida** a cada destino.
- Hay que analizar la exportación de rutas a un AS proveedor y a un AS con relación entre iguales. A un AS cliente se le exportan todas las rutas.
- La exportación de rutas se puede configurar de diferentes formas, vamos a expresar la exportación según AS_PATH que tenga una ruta.

Funcionamiento de AS_PATH

- El AS_PATH va modificándose cuando la ruta se propaga por diferentes ASs:
  - El AS que ha originado la ruta se encuentra a la derecha dentro de la lista AS_PATH.
  - El AS que me ha anunciado una ruta se encuentra a la izquierda dentro de la lista AS_PATH.
  - Cuando un router propaga una ruta a otro AS, incluye su número de sistema autónomo insertándolo al principio, a la parte izquierda de AS_PATH,
- Por ejemplo si un router BGP `r1` del sistema autónomo AS500 tiene una ruta con:



  - Si el router `r1` tiene que propagar ese anuncio a otro AS vecino, enviará dicho anuncio con AS_PATH= 500, 100, 300, 400, 200 i

El AS_PATH en las políticas de exportación

- Voy a decidir exportar una ruta preferida en función de:
  - La relación con el AS que me ha anunciando dicha ruta (AS del que proviene).
  - La relación con el AS al que estoy considerando enviar dicha ruta.
- Si la ruta que voy a exportar:
  - Proviene de un AS cliente:
    - se la tengo que exportar a todos los ASs vecinos (salvo al AS que me la está anunciando).
  - Proviene de un AS proveedor:
    - se la tengo que exportar sólo a los ASs clientes.
  - Proviene de AS entre iguales:
    - se la tengo que exportar sólo a los ASs clientes.
- **Las políticas de exportación se escriben para cada uno de los ASs vecinos como una lista de subredes a permitir/denegar su exportación dentro de bgpd.conf.**

Política de exportación a un vecino

- Creación de un filtro de reglas de exportación para un vecino:
  - **Definición de filtro para vecino:** Para un vecino BGP <vecino-BGP> se define un filtro **out** (de exportación) al cuál le asignamos un nombre <nombreLista>:

```
neighbor <vecino-BGP> filter-list <nombreLista> out
```

  - **Reglas del filtro:** El filtro contiene una lista **ordenada** de reglas. En cada regla se define una acción deny/permit y una condición expresada como un patrón en el atributo AS_PATH de una ruta. Si la ruta que se está comprobando cumple la condición, se aplicará la acción definida. Si una ruta no cumple la condición, no se aplica la acción y se pasa a la comprobación de la siguiente regla.

```
ip as-path access-list <nombreLista> [deny/permit] <patrónAS_PATH>
...
ip as-path access-list <nombreLista> [deny/permit] <patrónAS_PATH>
```

    Las reglas de un filtro se aplicarán en el orden en el que las hemos escrito.

- Antes de anunciar una subred a un vecino, BGP consulta si hay un filtro de exportación y **para cada ruta prefererida en la tabla BGP**:
  - Comprueba si el atributo AS_PATH de esa ruta cumple la condición descrita en la primera regla filtro (<patrónAS_PATH>). En caso afirmativo, aplicará la acción (deny/permit), denegando la exportación o permitiéndola y ya no se comprobarán más reglas para esa ruta.
  - Si la ruta no cumple la primera regla, se comprueba la siguiente y así sucesivamente.
  - Si no se cumple ninguno de los patrones AS_PATH de las reglas del filtro, la acción por defecto es deny. No se anuncia.

Patrón para AS_PATH en las políticas de exportación

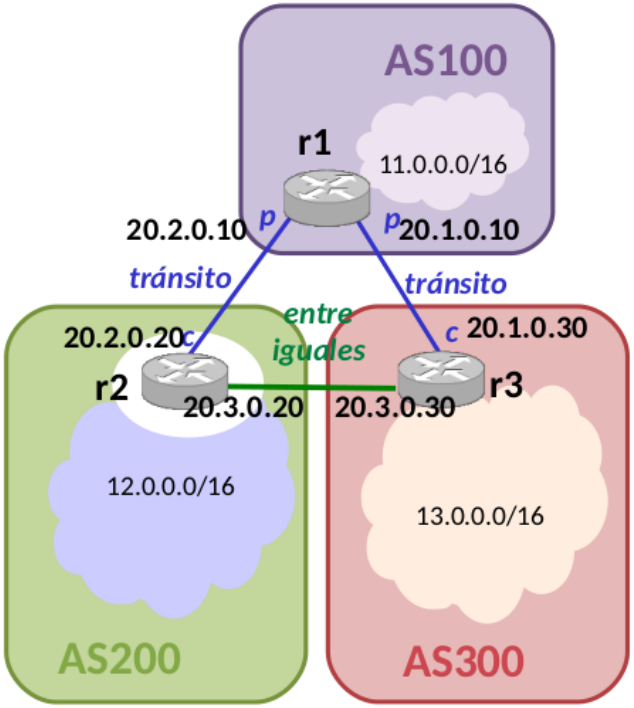
- Los ejemplos de <patrónAS_PATH> que vamos a utilizar en las políticas de exportación son:

^100	AS_PATH que comience con el número 100. Cualquier ruta que me haya anunciado AS100. Se refiere al último valor que se ha insertado en AS_PATH (se encuentra a la izquierda)
.*	Cualquier AS_PATH.

Políticas de exportación de rutas: Ejemplo

- Si ningún router ha configurado lista de exportación, r3 debería tener una tabla BGP:

```
r3# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric  LocPrf  Weight  Path
*> 12.0.0.0/16      20.3.0.20                    200             200 i
*                   20.1.0.10                                    100 200 i
*> 11.0.0.0/16      20.3.0.20                    200             200 100 i
*                   20.1.0.10                                    100 i
```

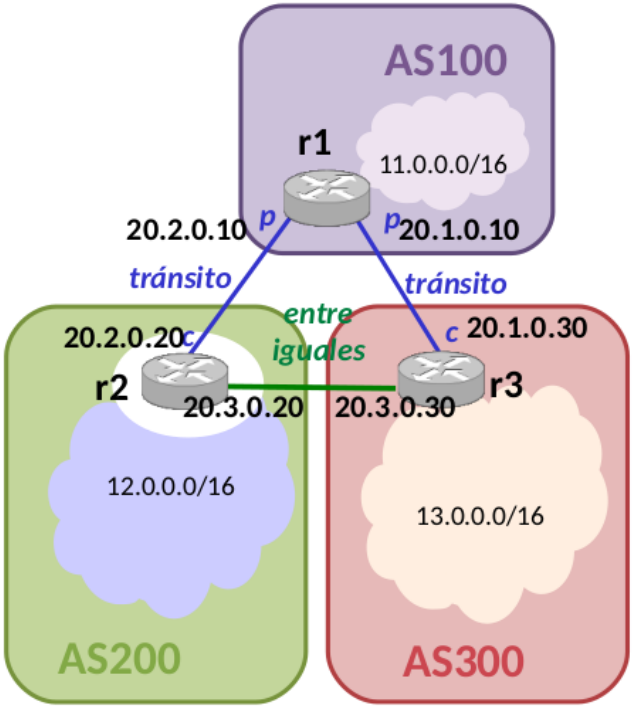


- r2 debe modificar su lista de exportación para **NO EXPORTAR** la ruta de AS100 a r3.

Políticas de exportación de rutas: Ejemplo

- Si ningún router ha configurado lista de exportación, r1 debería tener una tabla BGP:

```
r1# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric  LocPrf  Weight  Path
*> 12.0.0.0/16      20.2.0.20                    200             200 i
*                   20.1.0.30                                    300 200 i
*> 13.0.0.0/16      20.2.0.20                    200             200 300 i
*                   20.1.0.30                                    300 i
```

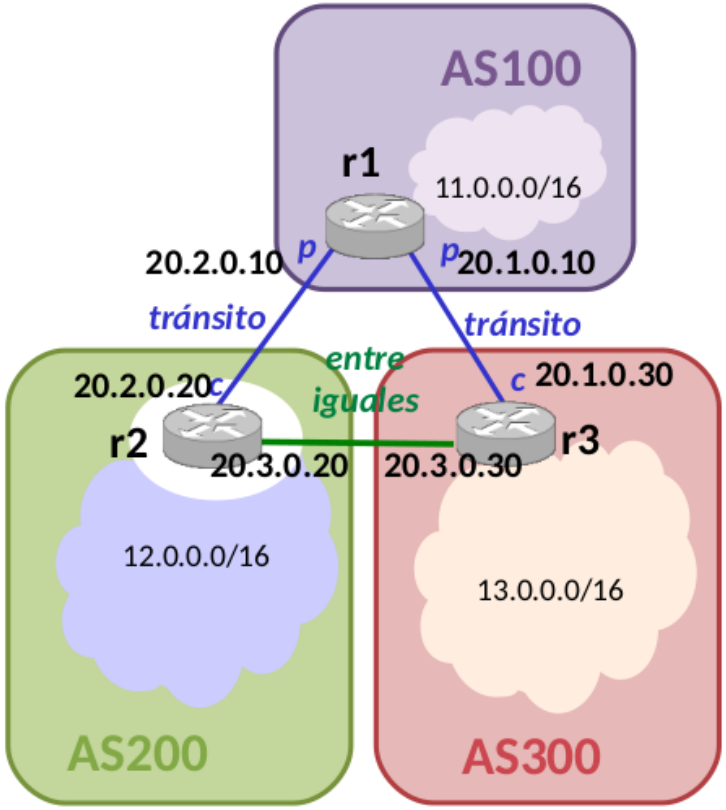


- r2 debe modificar su lista de exportación para **NO EXPORTAR** la ruta de AS300 a r1.
- r3 debe modificar su lista de exportación para **NO EXPORTAR** la ruta de AS200 a r1.

Políticas de exportación de rutas: Ejemplo

- Configuración de `bgpd.conf` en r2:

```
router bgp 200
  neighbor 20.3.0.30 remote-as 300
  neighbor 20.2.0.10 remote-as 100

  neighbor 20.3.0.30 filter-list listaHaciaAS300 out
  neighbor 20.2.0.10 filter-list listaHaciaAS100 out

  redistribute ...
  redistribute ...

  aggregate-address ...
  aggregate-address ...

  ip as-path access-list listaHaciaAS300 deny ^100
  ip as-path access-list listaHaciaAS300 permit .*

  ip as-path access-list listaHaciaAS100 deny ^300
  ip as-path access-list listaHaciaAS100 permit .*
```



Desde `r2`:

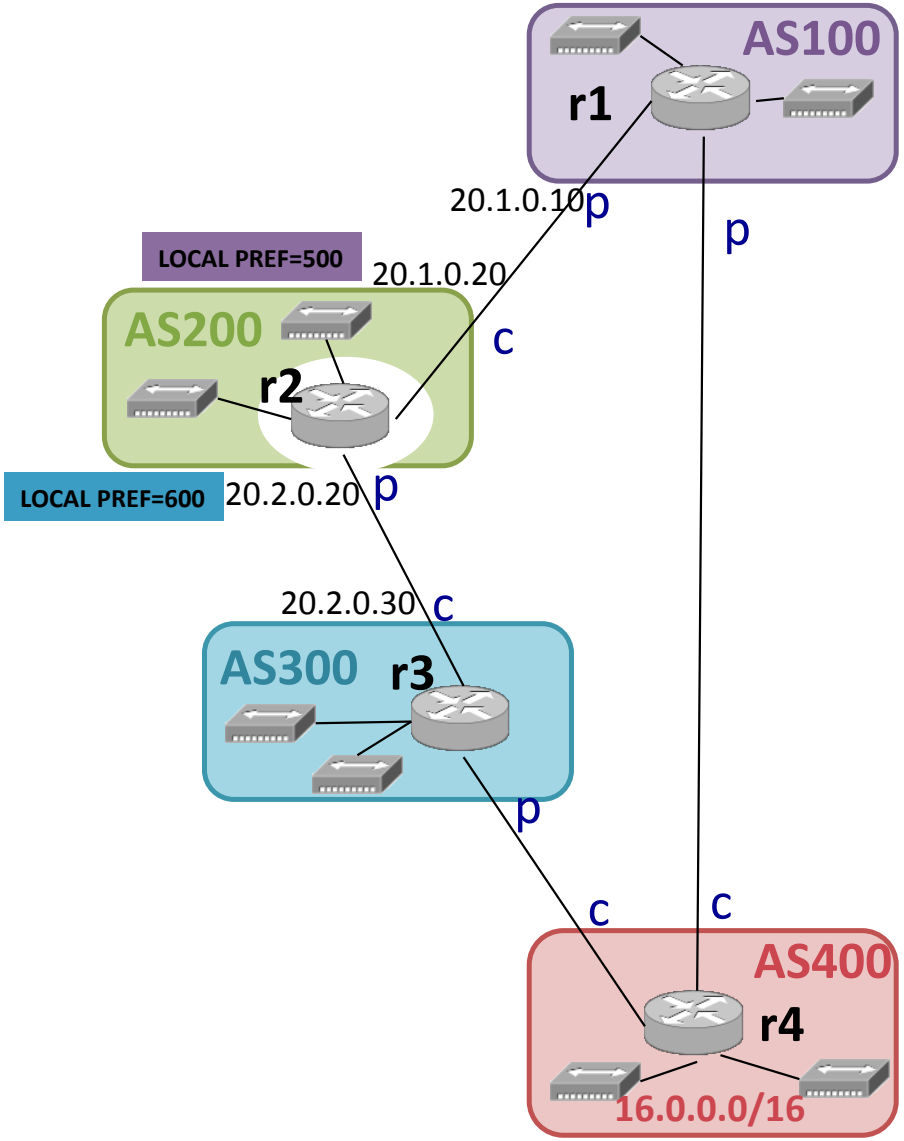
- No se envía ningún anuncio a `r3` que contenga subredes con `AS_PATH` cuyo primer AS sea AS100. Por tanto, **a r3 no se envían las subredes que me haya anunciado AS100. Sí se permite el envío del resto de las subredes.**
- No se envía ningún anuncio a `r1` que contenga las subredes cuyo primer AS de AS_PATH sea AS300. Por tanto **a r1 no se envían las subredes que me haya anunciado AS300. Sí se permite el envío del resto de las subredes.**

Contenidos

Configuración de una ruta por defecto (I)

- Si un AS tiene un solo proveedor, su proveedor podría anunciarle simplemente una ruta por defecto, en vez de todas las subredes que conoce.
- Así, por ejemplo, r2 podría anunciar a r3 todas las subredes que conoce con una ruta por defecto
- Para configurarlo, en el fichero de configuración de r2:

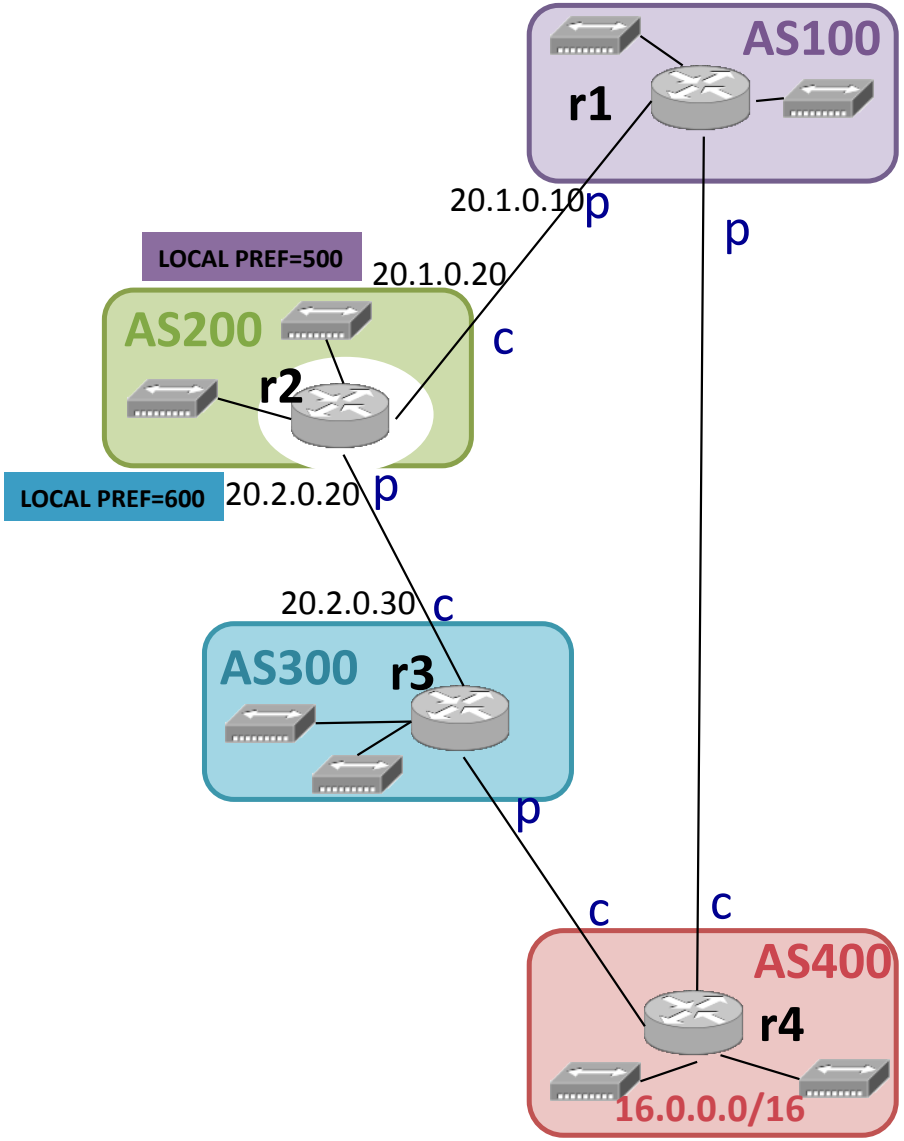
```
router bgp 200

  ...

  neighbor 20.2.0.30 remote-as 300
  neighbor 20.2.0.30 default-originate

  ...
```



Configuración de una ruta por defecto (II)

- La línea `default-originate` por sí sola no evita los anuncios de las redes originales, sino que simplemente anuncia una ruta por defecto.
- Para eliminar los anuncios de las redes originales hay que crear adicionalmente para ese vecino una `filter-list` que evite esos anuncios:

```
router bgp 200

  ...

  neighbor 20.2.0.30 remote-as 300
  neighbor 20.2.0.30 default-originate
  neighbor 20.2.0.30 filter-list listaHaciaAS300 out
  ...
  redistribute ...
  ...
  aggregate-address ...
  ...
  ip as-path access-list listaHaciaAS300 deny .*
  ...
```



Tema 5: Herramientas para el análisis de comunicaciones TCP/UDP

Sistemas Telemáticos

2º GIT – 2º GITT – 2º GIST

Eva M. Castro Barbero (eva.castro@urjc.es)
José Centeno González (jose.centeno@urjc.es)
Pedro de las Heras Quirós (pedro.delasheras@urjc.es)

Diciembre 2023

Contenidos

netstat

- La herramienta `netstat` permite obtener información sobre varios aspectos del estado de la red en un sistema Unix/Linux.
- Entre otros usos, permite ver el listado de comunicaciones activas en una máquina: detalles de las conexiones TCP y comunicaciones UDP que hay establecidas en ese momento.
- Sintaxis:

```
netstat -tna
```

```
netstat -una
```

  - la opción `-t` muestra información de las conexiones TCP
  - la opción `-u` muestra información de las comunicaciones UDP
  - la opción `-n` muestra direcciones IP (si se omite, se trata de mostrar nombres de máquinas por DNS en su lugar)
  - la opción `-a` muestra información de todas las comunicaciones, incluyendo aquellas en las que está máquina ha lanzado un servidor que está esperando recibir mensajes de clientes

netstat

- `netstat` mostrará la siguiente información para las comunicaciones activas:

```
pc1:~# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
```

  - La columna `Proto` indica el protocolo utilizado (UDP o TCP)
  - La columna `Local Address` muestra la dirección IP local de la máquina donde se esperan recibir datos y el número de puerto.
  - En la columna `Foreign Address` muestra la dirección IP y puerto de las máquinas remota con la que se ha establecido una comunicación.
  - Las columnas `Recv-Q` (`receiving queue`) y `Send-Q` (`sending queue`) muestran la cantidad de bytes que hay almacenados en los buffers locales reservados para la recepción de datos y emisión de datos de este servidor.
  - La columna `State` indicará el estado de la comunicación.

netstat: comunicaciones UDP (I)

- Para visualizar las comunicaciones UDP activas:

```
pc1:~# netstat -una
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:7777            0.0.0.0:*
```

  - El resultado de ejecutar este comando muestra un servidor UDP esperando recibir conexiones de clientes en el puerto 7777.
  - La columna `Local Address` muestra la dirección 0.0.0.0 que indica que se esperan recibir comunicaciones UDP en cualquiera de las direcciones IP configuradas actualmente en la máquina local.
  - En la columna `Foreign Address` se mostrarán las direcciones IP y puertos de las máquinas clientes remotos que se conecten con este servidor. Actualmente no hay ninguna.
  - Las columnas `Recv-Q` y `Send-Q` muestran que no hay datos almacenados en los buffers.

netstat: comunicaciones UDP (II)

- Para visualizar las comunicaciones UDP activas:

```
pc1:~# netstat -una
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 11.0.0.1:7777           11.0.0.2:32768          ESTABLISHED
```

  - El resultado de ejecutar este comando muestra una comunicación UDP entre la dirección IP local 11.0.0.1 y puerto 7777 y la dirección IP remota 11.0.0.2 y puerto 32768.
  - Las columnas `Recv-Q` y `Send-Q` muestran que no hay datos almacenados en los buffers.

netstat: comunicaciones TCP (I)

- Para visualizar las comunicaciones TCP activas:

```
pc1:~# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:7777            0.0.0.0:*               LISTEN
```

  - El resultado de ejecutar este comando muestra un servidor TCP esperando recibir conexiones de clientes en el puerto 7777.
  - La columna `Local Address` muestra la dirección 0.0.0.0 que indica que se esperan recibir comunicaciones UDP en cualquiera de las direcciones IP configuradas actualmente en la máquina local.
  - En la columna `Foreign Address` se mostrarán las direcciones IP y puertos de las máquinas clientes remotos que se conecten con este servidor. Actualmente no hay ninguna.
  - Las columnas `Recv-Q` y `Send-Q` muestran que no hay datos almacenados en los buffers.

netstat: comunicaciones TCP (II)

- Para visualizar las comunicaciones TCP activas:

```
pc1:~# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 11.0.0.1:7777           11.0.0.2:33715          ESTABLISHED
```

  - El resultado de ejecutar este comando muestra una comunicación TCP entre la dirección IP local 11.0.0.1 y puerto 7777 y la dirección IP remota 11.0.0.2 y puerto 33715.
  - Las columnas `Recv-Q` y `Send-Q` muestran que no hay datos almacenados en los buffers.

Contenidos

Herramienta nc

- Usaremos `nc` para generar tráfico TCP y UDP según el modelo de comunicaciones cliente/servidor.
- Se arrancarán 2 aplicaciones: una funcionando con el rol cliente y la otra con el rol servidor.
- **Siempre es necesario lanzar primero la aplicación que funciona como servidor**. El servidor quedará a la espera de recibir tráfico procedente de la aplicación cliente, que se deberá lanzar después.
  - Una aplicación lanzada con `nc` como **cliente** lee de la entrada estándar (por omisión el teclado) los caracteres introducidos y al pulsar la tecla `INTRO` la línea de texto es enviada usando TCP o UDP a la aplicación servidor.
  - Al recibir la línea de texto, la aplicación **servidor** lanzada con `nc` mostrará en la pantalla los datos recibidos de la aplicación cliente lanzada con `nc`.

Contenidos

Aplicación servidor UDP

- Para arrancar `nc` como **servidor** utilizando el protocolo UDP ejecutaremos la siguiente orden:

```
nc -u -l -p <Pto-Loc>
```

  - `-u`: UDP
  - `-l`: *listen* = modo servidor
  - `-p <Pto-Loc>`: número de puerto local UDP en el que la aplicación servidor esperará recibir los datagramas UDP de una aplicación cliente.

- Por ejemplo, si queremos arrancar una aplicación servidor UDP en el puerto 7777 de la máquina `pc1` utilizaremos la siguiente orden:

```
pc1:~# nc -u -l -p 7777
```

Aplicación cliente UDP

- Para arrancar `nc` como **cliente** utilizando el protocolo UDP ejecutaremos la siguiente orden:

```
nc -u -p <Pto-Loc> <IP-dest> <Pto-dest>
```

  Donde:

  - `-u`: UDP
  - `-p <Pto-Loc>`: número de puerto local UDP en el que la aplicación cliente esperará recibir los datagramas UDP que vengan del servidor.
  - `<IP-dest>`: dirección IP de la máquina donde se está ejecutando la aplicación servidor UDP.
  - `<Pto-dest>` es el número de puerto UDP en el que escucha la aplicación servidor UDP.

- Por ejemplo, si queremos arrancar una aplicación cliente UDP en `pc2` que espere recibir datagramas UDP en el puerto 6666 y que envíe datagramas UDP a la dirección IP 200.0.0.1 y puerto 7777 (donde se encuentra esperando recibir datagramas UDP la aplicación servidor) utilizaremos la siguiente orden:

```
pc2:~# nc -u -p 6666 200.0.0.1 7777
```

Envío de datos UDP

- Una vez lanzadas las aplicaciones servidor UDP y cliente UDP, el cliente puede enviarle líneas de texto al servidor.
- Después de que el cliente haya enviado al menos una línea de texto al servidor, todo lo que escribamos a través de la entrada estándar de un extremo será enviado al otro extremo como datagramas UDP: si escribimos en el terminal de la aplicación cliente, esto será enviado a la aplicación servidor, y viceversa.
- Pasado un cierto tiempo desde el último mensaje del cliente, el servidor "olvida" al cliente (recuerda que en UDP no hay conexiones) y es necesario volver a enviar un mensaje desde el cliente para que el servidor pueda volver a enviarle mensajes.
- Para interrumpir la ejecución de estas aplicaciones se debe utilizar `Ctrl+C`.

Contenidos

Aplicación servidor TCP

- Para arrancar `nc` como **servidor** utilizando el protocolo TCP ejecutaremos la siguiente orden:

```
nc -l -p <Pto-Loc>
```

  Donde:

  - `-l`: *listen* = modo servidor
  - `-p <Pto-Loc>`: es el número de puerto local TCP en el que la aplicación servidor esperará recibir mensajes TCP de una aplicación cliente.

- Por ejemplo, si queremos arrancar una aplicación servidor TCP en el puerto 7777 de la máquina `pc1` utilizaremos la siguiente orden:

```
pc1:~# nc -l -p 7777
```

Aplicación cliente TCP

- Para arrancar `nc` como **cliente** utilizando el protocolo TCP ejecutaremos la siguiente orden:

```
nc -p <Pto-Loc> <IP-dest> <Pto-dest>
```

  Donde:

  - `-p <Pto-Loc>`: número de puerto local TCP en el que la aplicación cliente esperará recibir los mensajes de la aplicación servidor TCP.
  - `<IP-dest>`: dirección IP de la máquina donde se está ejecutando la aplicación servidor TCP.
  - `<Pto-dest>`: número de puerto TCP en el que escucha la aplicación servidor TCP.

- Por ejemplo, si queremos arrancar una aplicación cliente TCP en `pc2` que utilice el puerto origen 6666 para establecer una conexión TCP con un servidor TCP que escuche en el puerto destino 7777 de la máquina 200.0.0.1, utilizaremos la siguiente orden:

```
pc2:~# nc -p 6666 200.0.0.1 7777
```

Envío de datos TCP

- Una vez iniciada la aplicación servidor TCP, ésta se queda esperando recibir mensajes de una aplicación cliente TCP.
- Una vez iniciada la aplicación cliente TCP, ésta intercambiará unos mensajes de control (apertura de conexión) con la aplicación servidor, por lo que es imprescindible que dicha aplicación servidor haya sido lanzada antes.
- Si la comunicación entre ambas aplicaciones es posible, a partir de este momento todo lo que escribamos a través de la entrada estándar de una aplicación será enviada a la otra: si escribimos en el terminal de la aplicación cliente, esto será enviado a la aplicación servidor, y viceversa.
- Para interrumpir la ejecución de estas aplicaciones se debe utilizar `Ctrl+C`.

Contenidos

Gráfica de *tcptrace* dentro de Wireshark

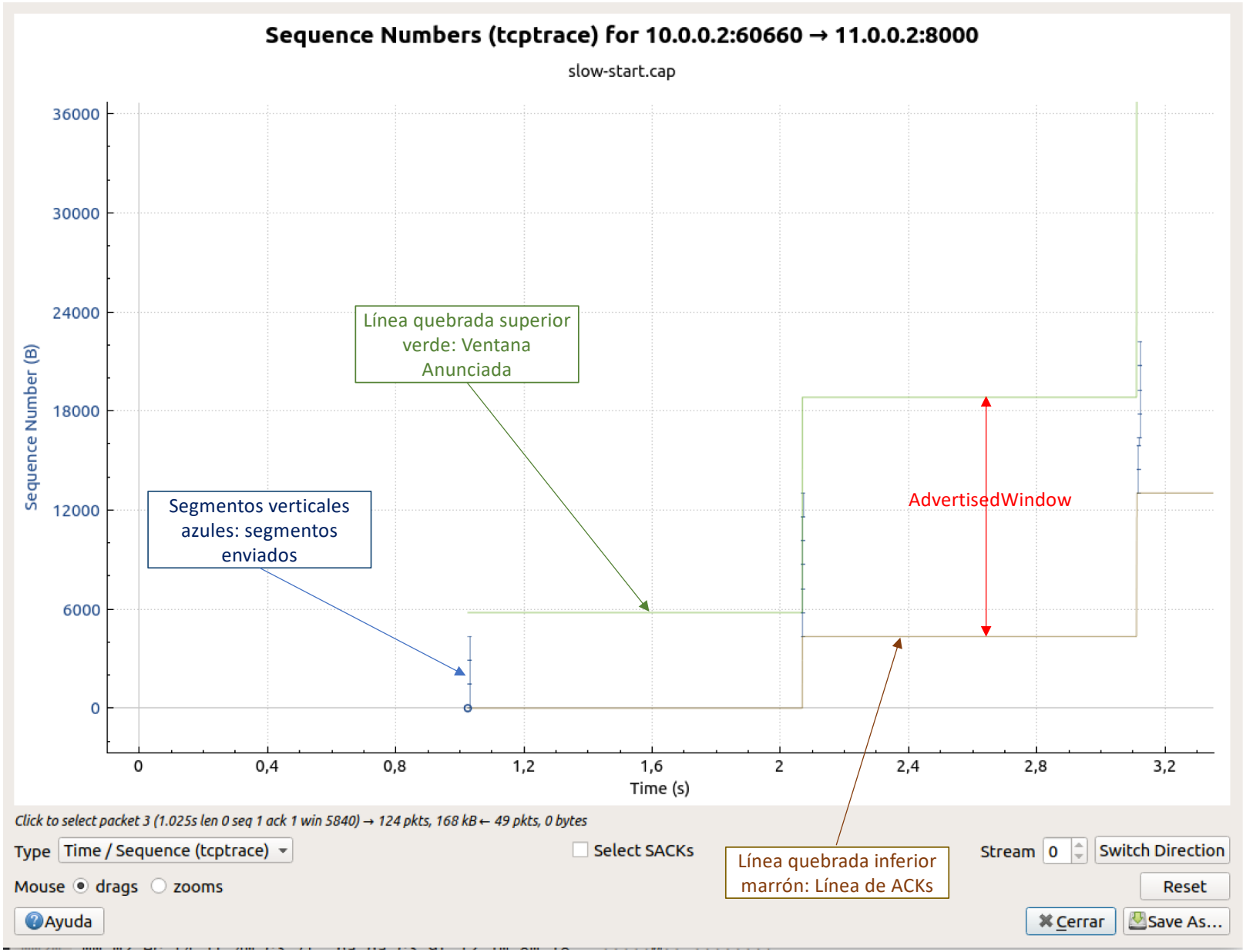
- En Wireshark, además de mirar el contenido de los paquetes de una conexión TCP, puede verse en una gráfica la **evolución del envío de datos y recepción de acks respecto al tiempo**.
- Wireshark permite mostrar varios tipos de gráficas de una conexión TCP: Nosotros utilizaremos la gráfica de *tcptrace*.
- Como una conexión TCP permite el envío de datos en ambos sentidos, se pueden visualizar 2 gráficas de *tcptrace* diferentes: las correspondientes a cada sentido de la comunicación.
- Para ver en Wireshark la gráfica de *tcptrace* de uno de los sentidos de una conexión TCP es necesario:
  - Cargar el fichero de una captura que contenga los paquetes de una conexión TCP.
  - Seleccionar un segmento de la conexión del sentido de la comunicación que queremos analizar (si el segmento seleccionado va del proceso A al proceso B, la gráfica que se mostrará será la correspondiente al envío de datos de A a B).
  - Seleccionar en el menú de Wireshark:
    Statistics→TCP Stream Graph→Time-Sequence Graph (tcptrace)

Versiones de Wireshark

- En las versiones recientes de Ubuntu hay dos versiones diferentes de Wireshark, en las que varía un poco la apariencia de las gráficas *tcptrace*:
  - `wireshark` (a veces queda instalado con nombre `wireshark-qt`)
  - `wireshark-gtk`
- Por defecto suele instalarse al versión "nueva" (`wireshark`). Si se quiere tener instalada también la versión "antigua":

  `sudo apt install wireshark-gtk`.

wireshark: Apariencia



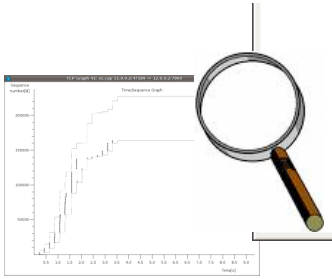
wireshark: Controles

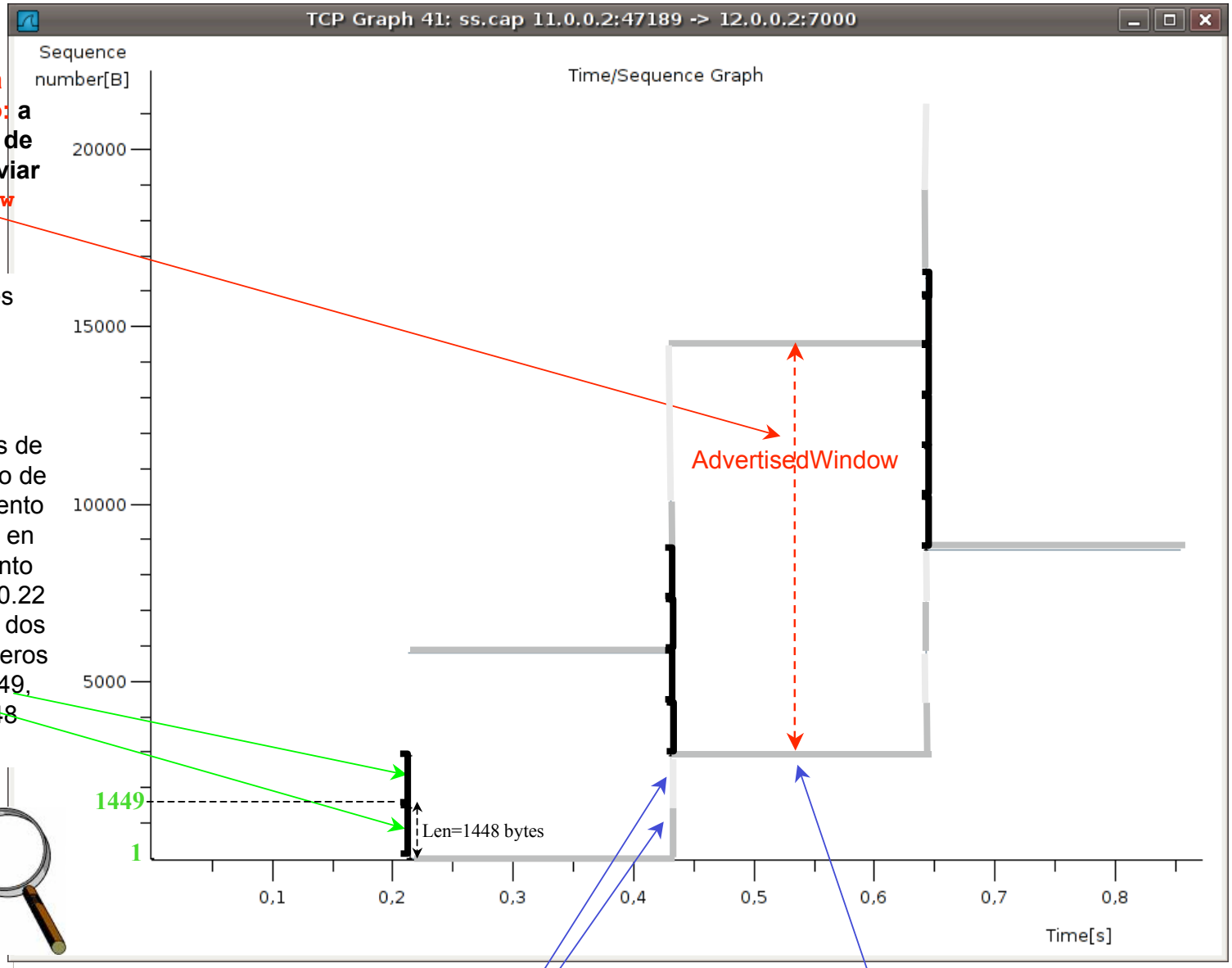
- **Rueda del ratón**: zoom in/out
- **Arrastrar con el botón izquierdo**: desplazar el gráfico (útil si se ha hecho "zoom in")
- **ESPACIO**: activa/desactiva una cruz para ayudar a ver sobre los ejes la posición del ratón.
- **Click izquierdo sobre un segmento**: seleccionar el paquete concreto en la lista de paquetes de Wireshark.
- **Botón 'Switch Direction'**: pasa a mostrar el otro sentido de la conexión.

wireshark-gtk: Apariencia

Ventana anunciada por el otro extremo: a partir del último nº de ACK se pueden enviar `AdvertisedWindow` bytes

Segmentos verticales negros: **segmentos TCP de datos enviados**. Cada segmento ocupa un conjunto de números de secuencia: el número de secuencia del segmento TCP más la longitud en bytes de ese segmento TCP. En el instante 0.22 segundos se envían dos segmentos con números de secuencia 1 y 1449, cuya longitud es 1448 bytes.





Segmentos verticales grises: **segmentos TCP de ACK** recibidos

Línea inferior gris claro: **último nº de ACK recibido**

wireshark-gtk: Controles

- **Click central**: zoom in
- **MAYS + Click central**: zoom out
- **Arrastrar con el botón derecho**: desplazar el gráfico (útil si se ha hecho "zoom in")
- **ESPACIO**: activa/desactiva una cruz para ayudar a ver sobre los ejes la posición del ratón.
- **Click izquierdo sobre un segmento**: seleccionar el paquete concreto en la lista de paquetes de Wireshark.
- **CTRL + arrastrar con el botón derecho**: lupa
- **s**: Alterna entre números de secuencia relativos y absolutos, sólo si está desactivada la opción Edit→Preferences→Protocols→TCP→Relative sequence numbers and window scaling.

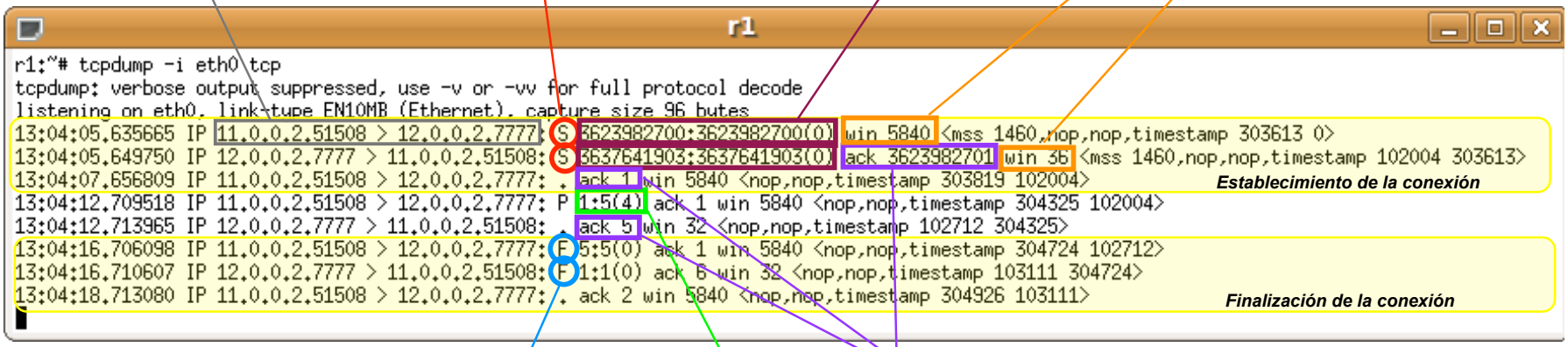
Seguimiento de conexiones con tcpdump en el terminal

IP1.puerto1 > IP2.puerto2
IP1.puerto1: origen
IP2.puerto2: destino

Flag SYN

x:y(z)
x: Número de secuencia inicial real
y: x+z
z: Número de bytes de datos enviados

Tamaño de ventana anunciada



Flag FIN

Número de asentimiento,
siguiente nº de secuencia que espero recibir

x:y(z)
x: Número de secuencia relativo del primer byte de datos del segmento
y: x+z
z: Número de bytes de datos enviados

Tema 6: Cortafuegos (Firewalls) en Linux con iptables

Sistemas Telemáticos

2º GIT – 2º GITT – 2º GIST

Eva M. Castro Barbero (eva.castro@urjc.es)
José Centeno González (jose.centeno@urjc.es)
Pedro de las Heras Quirós (pedro.delasheras@urjc.es)

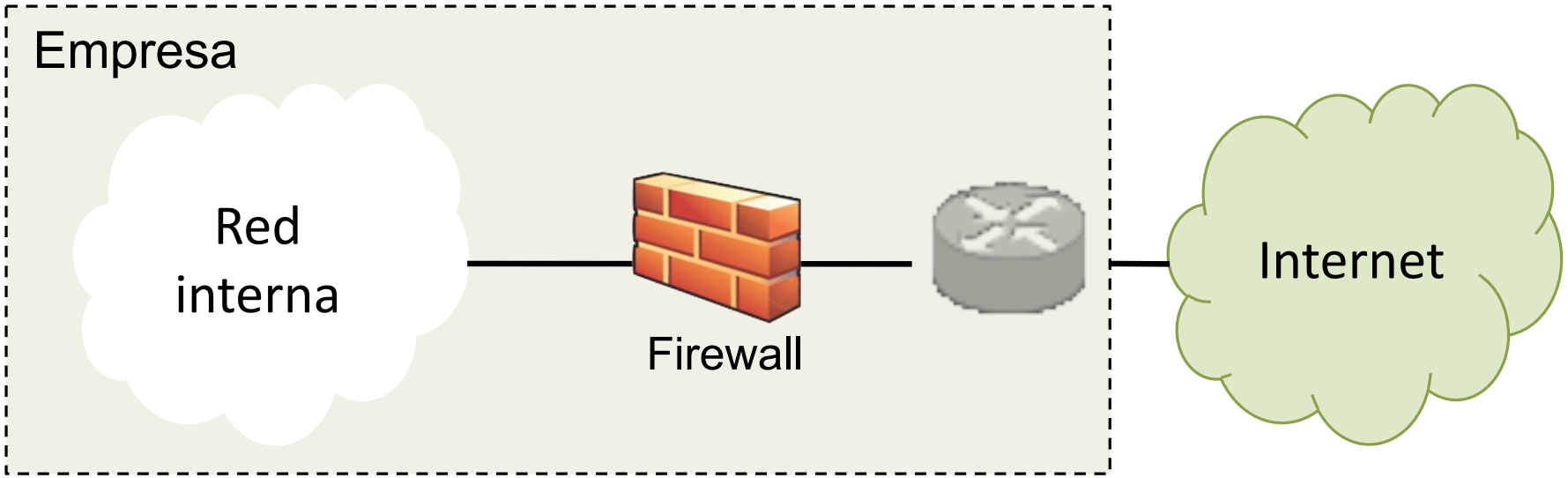
Diciembre 2023

Contenidos

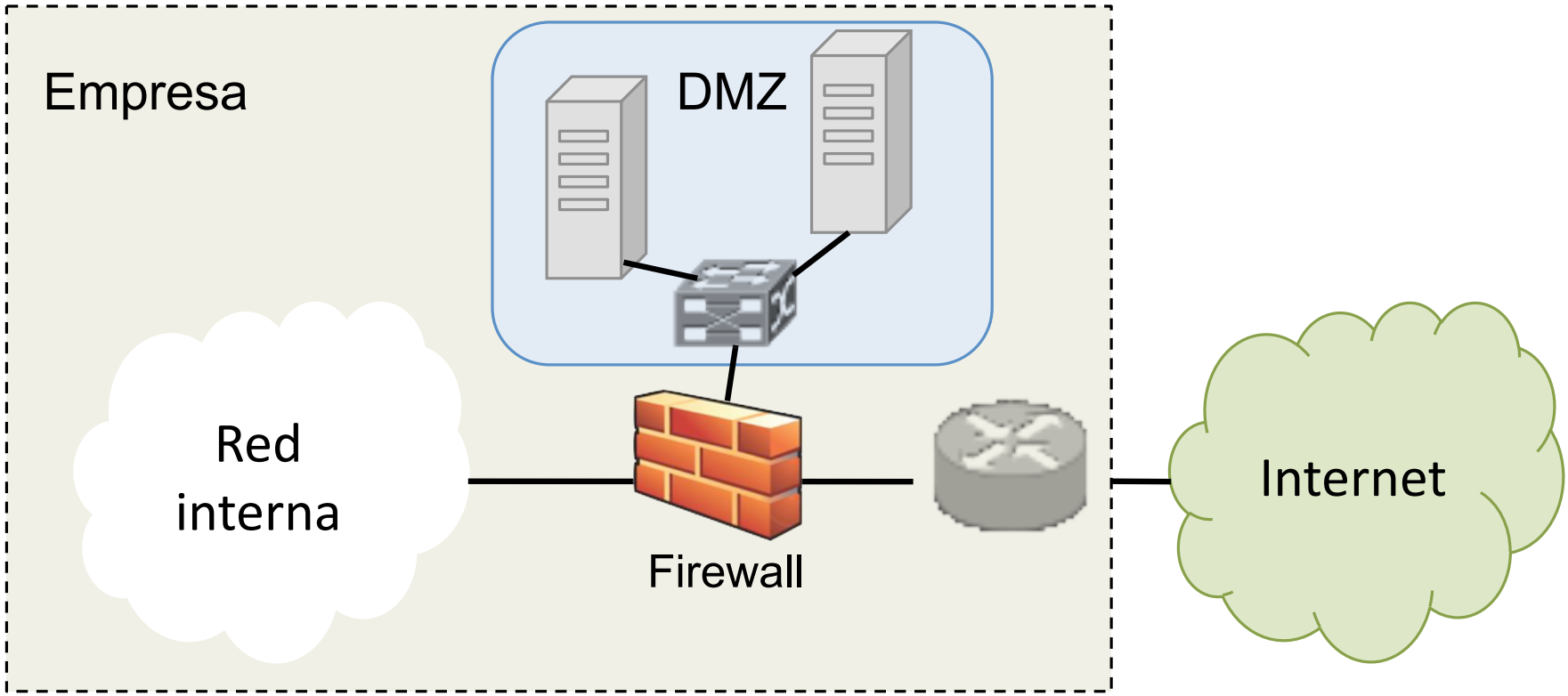
La red frontera

- La red frontera es la parte de la red que comunica la red interna de una empresa con otras redes externas.
- La seguridad en la red frontera es clave para proteger los equipos y servicios de la empresa de ataques externos. Para ello, las empresas instalan *firewalls* que permiten filtrar el tráfico y detectar posibles ataques maliciosos desde el exterior. Adicionalmente los *firewalls* permiten restringir el tráfico que sale de los equipos internos de la empresa.

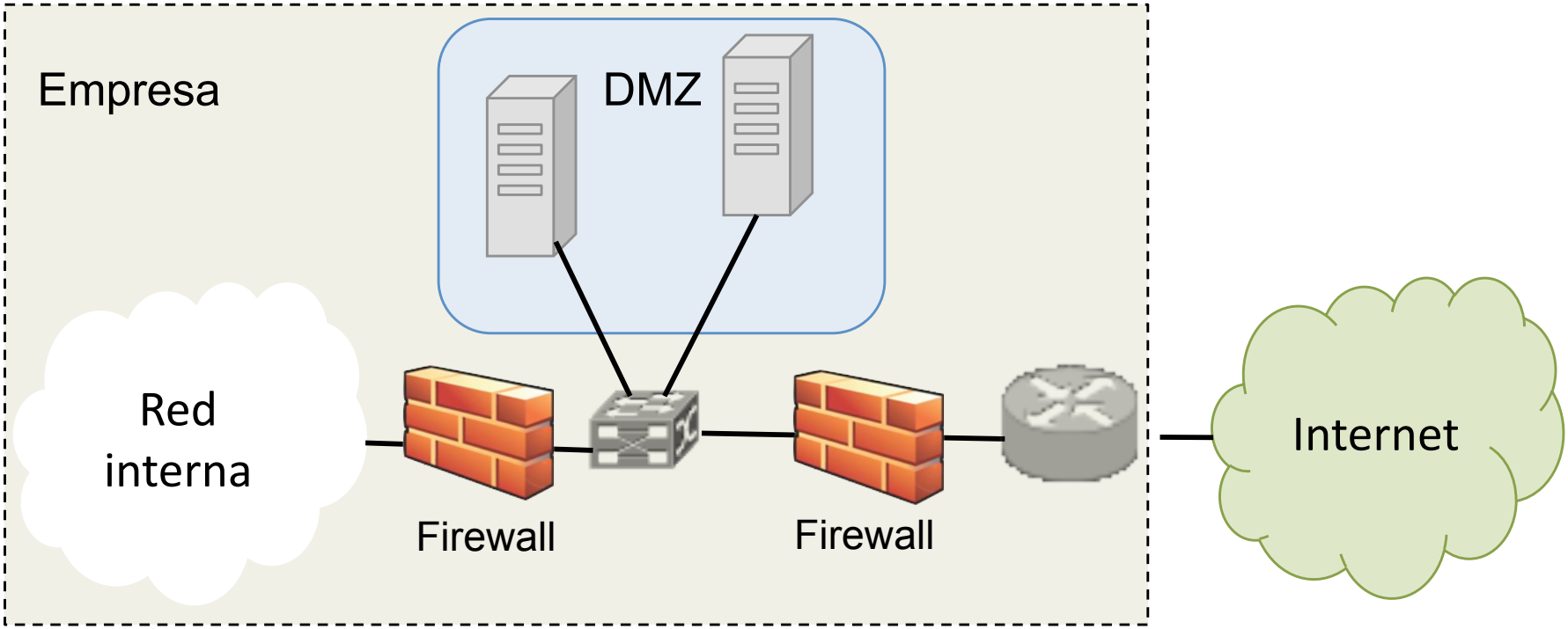
Un único firewall



Un único firewall con zona DMZ



Dos firewalls



Contenidos

Contenidos

Netfilter - iptables

- Netfilter[1] es un framework de Linux que permite interceptar y modificar paquetes IP.
- `iptables` es una herramienta de Netfilter que permite al administrador la definición de conjuntos de reglas aplicables a los paquetes IP que entran y/o salen de una máquina para realizar las siguientes operaciones:
  - Filtrado de paquetes *(packet filtering)*.
  - Seguimiento de conexiones *(connection tracking)*.
  - Traducción de direcciones IP y puertos (NAT, *Network Address Translation*).
- Hay 3 conceptos básicos en `iptables`:
  - **reglas**
  - **cadenas**
  - **tablas**

[1] `http://www.netfilter.org`

Reglas

- Una regla de `iptables` especifica una **condición** y una **acción**:
  - **condición**: características que debe cumplir un paquete para que la regla le sea aplicable. Ejemplos de condiciones:

    `-p tcp --dport 80`: el protocolo es TCP y el puerto destino es 80

    `-s 13.1.2.0/24`: la dirección de origen es de la subred 13.1.2.0/24.

  - **acción**: indica lo que se hace con el paquete si cumple la condición de la regla. Ejemplos de acciones:

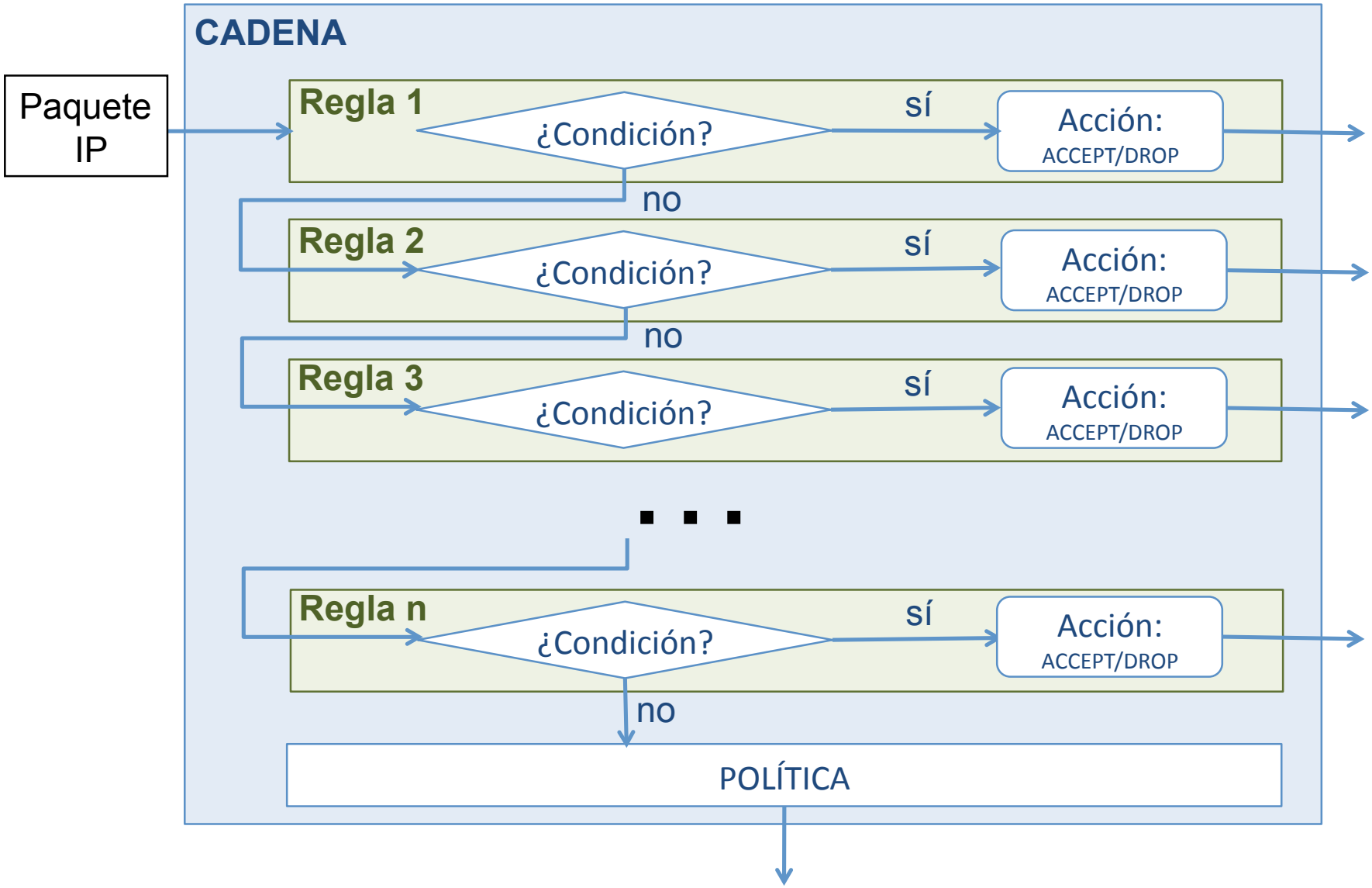
    `ACCEPT`: el paquete se acepta
    `DROP`: el paquete se descarta
    `SNAT --to-source 13.1.2.1`: se cambia la IP origen del paquete

- Las reglas se agrupan en listas de reglas, llamadas cadenas.
- Las cadenas se agrupan en tablas.

Cadenas (I)

- Una cadena es una **lista ordenada de reglas**.
- Para cada paquete se va comprobando si se le aplica cada regla de la cadena (es decir, si cumple la **condición**):
  - Si una regla NO se aplica a un paquete, se pasa a la siguiente regla de la cadena.
  - Si una regla SÍ se aplica a un paquete, se ejecuta la **acción** definida en dicha regla. Dependiendo del tipo de acción:
    - El paquete abandona la comprobación del resto de las reglas y pasa a la siguiente cadena (acciones ej: ACCEPT, DROP)
    - El paquete continúa con la siguiente regla de la cadena (acción ej: LOG)
- Una cadena puede tener definida una política, que es la **acción por defecto** para la cadena. La política predefinida para todas las cadenas predefinidas es `ACCEPT` (es decir, aceptar el paquete).
- Cuando para un paquete NO se aplica NINGUNA de las reglas de la cadena, se ejecuta para él la política de la cadena (si dicha cadena la tiene).

Cadenas (II)

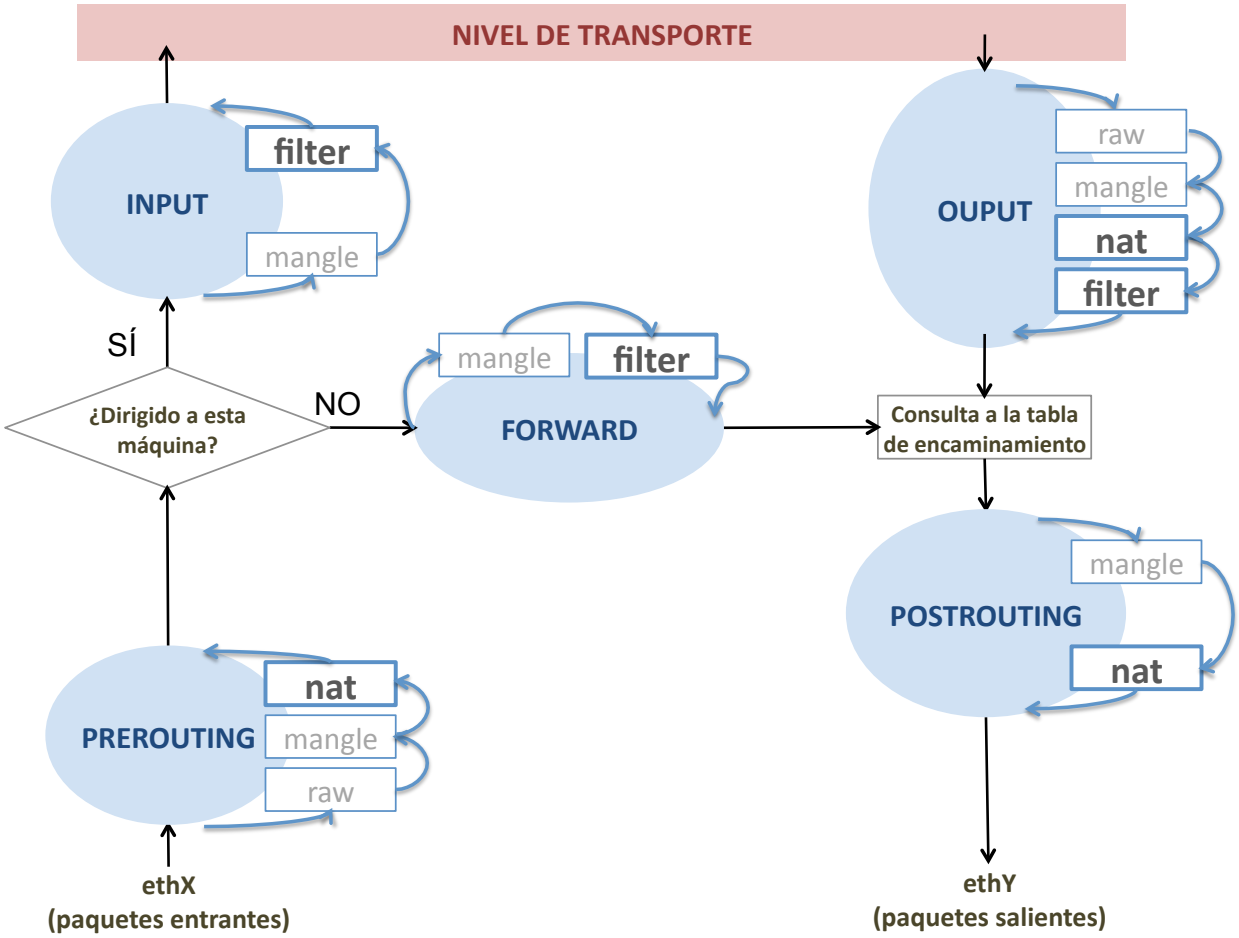

CADENA
Paquete IP
Regla 1
¿Condición?
sí
Acción:
ACCEPT/DROP
no
Regla 2
¿Condición?
sí
Acción:
ACCEPT/DROP
no
Regla 3
¿Condición?
sí
Acción:
ACCEPT/DROP
Regla n
¿Condición?
sí
Acción:
ACCEPT/DROP
no
POLÍTICA

Cadenas (III)

- Las reglas tienen una posición determinada (**número de regla**) dentro de la cadena. A la hora de añadir una nueva regla en una cadena, hay tres posibilidades:
  - **añadir la regla al final de la cadena**, detrás de las ya existentes
  - **reemplazar** en una posición a otra regla ya existente
  - **insertar** la regla en una posición ya existente, desplazando un lugar a las reglas existentes desde esa posición en adelante.

Cadenas (IV): Tipos de cadenas

- Existen diferentes tipos de cadenas:
  - Predefinidas: `PREROUTING`, `INPUT`, `FORWARD`, `OUTPUT`, `POSTROUTING`
  - Definidas por el usuario. Dichas cadenas no tienen **política** predefinida.
- Cuando un paquete llega a una máquina se le aplican las reglas de las cadenas predeterminadas en distintos momentos según el esquema de la figura



Cadenas (V): Cadenas predefinidas

- Cadena `PREROUTING`:
  - Reglas que se aplican a los paquetes que llegan a la máquina. Esta cadena se ejecuta antes de comprobar si el paquete es para la propia máquina o hay que reenviarlo.
- Cadena `INPUT`:
  - Reglas que se aplican a los paquetes destinados a la propia máquina. Esta cadena se ejecuta justo antes de entregarlos a la aplicación local.
- Cadena `FORWARD`:
  - Reglas que se aplican a los paquetes que han llegado a la máquina pero van destinados a otra y hay que reenviarlos. Esta cadena se ejecuta antes de consultar la tabla de encaminamiento.
- Cadena `OUTPUT`:
  - Reglas que se aplican a los paquetes creados por la propia máquina. Esta cadena se ejecuta justo después de que la aplicación le pase los datos a enviar al *kernel* del sistema operativo y antes de consultar la tabla de encaminamiento.
- Cadena `POSTROUTING`:
  - Reglas que se aplican a los paquetes que salen de la máquina, tanto los creados por ella como los que se reenvían. Esta cadena se ejecuta después de consultar la tabla de encaminamiento.

Tablas (I)

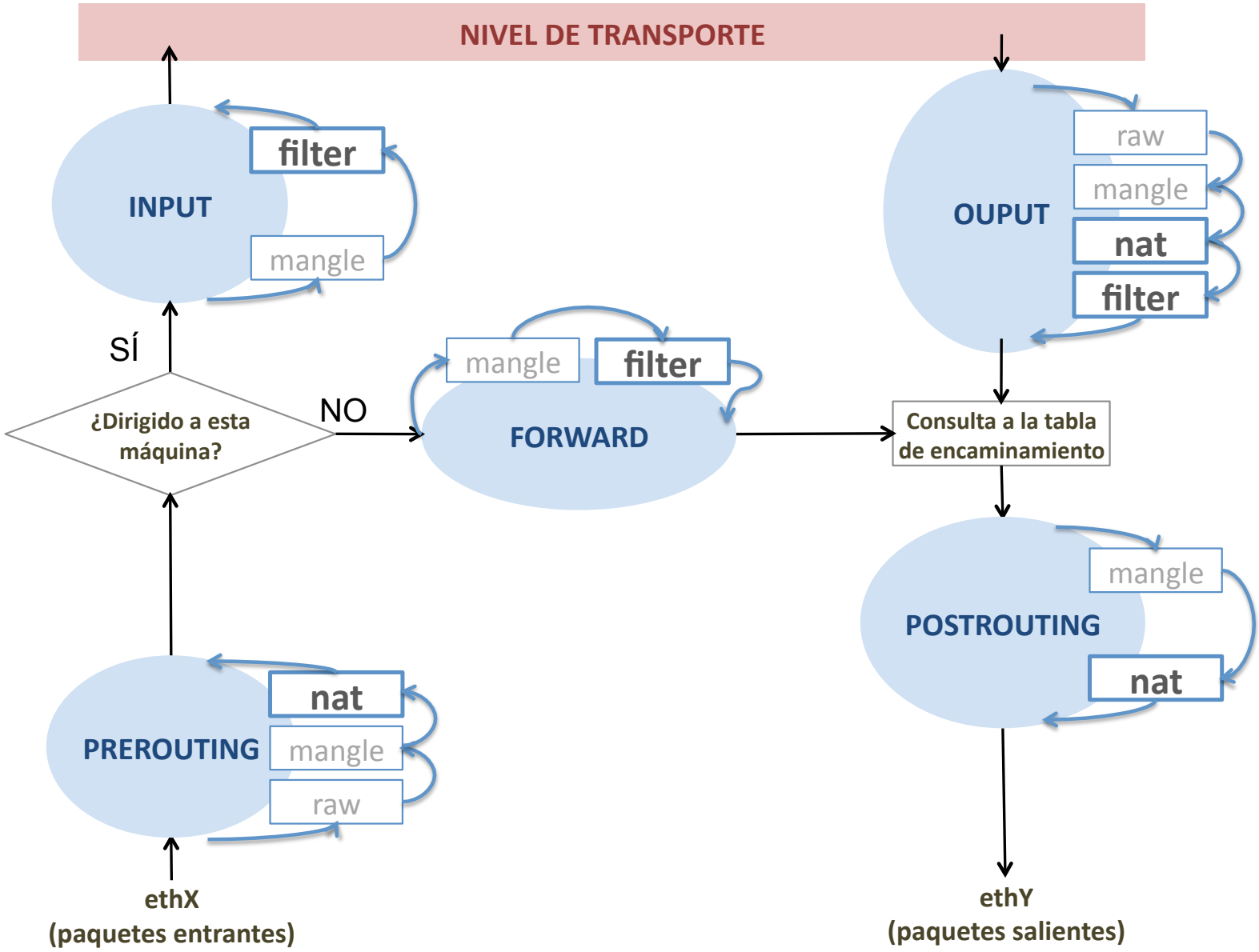
- Una tabla de `iptables` contiene un conjunto de cadenas, tanto predefinidas como de usuario.
- Una tabla concreta engloba las reglas (agrupadas en cadenas) relacionadas con un tipo de procesamiento de los paquetes.
- `Netfilter` define las siguientes tablas:
  - `filter`: engloba las reglas de filtrado de paquetes, es decir, de las que deciden que un paquete continúe su camino o sea descartado.
  - `nat`: engloba las reglas de modificación de direcciones IP y puertos de los paquetes
  - `mangle`: engloba las reglas de modificación de algunos campos de las cabeceras del paquete. Ejemplo: ToS
  - `raw`: engloba las reglas que permiten marcar excepciones al seguimiento que hace el *kernel* de las "conexiones"[2] de la máquina.

[2] "conexiones" en sentido amplio: no sólo conexiones TCP, sino también tráfico UDP enviado/recibido para las mismas direcciones y puertos, tráfico ICMP de petición/respuesta de eco...

Tablas (II): Cadenas predefinidas de cada tabla

- La tabla `filter` incluye las cadenas:
  - `FORWARD`
  - `INPUT`
  - `OUTPUT`
- La tabla `nat` incluye las cadenas:
  - `PREROUTING`
  - `OUTPUT`
  - `POSTROUTING`
- La tabla `mangle` incluye las cadenas:
  - `PREROUTING`
  - `FORWARD`
  - `INPUT`
  - `OUTPUT`
  - `POSTROUTING`
- La tabla `raw` incluye las cadenas:
  - `PREROUTING`
  - `OUTPUT`

Movimiento de los paquetes por tablas y cadenas



Contenidos

iptables: comandos

`iptables [-t <tabla>] <comando> [<condición>] [<acción>]`

Si no se especifica una tabla se utilizará por defecto la tabla `filter`.

- Comandos más utilizados:

`iptables [-t <tabla>] -L [<cadena>] [-v] [-n]` lista las reglas definidas en una `cadena` de una `tabla`. Si se omite la cadena el comando actúa sobre todas. Con `-v` se mostrará también el número de paquetes y bytes que han cumplido la condición de cada regla.
`iptables [-t <tabla>] -F [<cadena>]` borra la lista de reglas que hay en una `cadena` de una `tabla`. Si se omite la cadena el comando actúa sobre todas.
`iptables [-t <tabla>] -Z [<cadena>]` reinicia los contadores de una `cadena` de una `tabla`: número de paquetes y bytes que cumplen las condiciones de sus reglas. Si se omite la cadena el comando actúa sobre todas.
`iptables [-t <tabla>] -N [<cadena-usuario>]` crea en una tabla una nueva cadena definida por el usuario.
`iptables [-t <tabla>] -P <cadena> <política>` establece la política por defecto para una cadena predefinida de una tabla, donde la política puede ser `DROP` o `ACCEPT`.
`iptables [-t <tabla>] -A <cadena> <condición> <acción>` añade una regla al final de las reglas que tiene definidas una cadena de una tabla. La regla queda definida por la ejecución de una `acción` si un paquete cumple una `condición`.

`iptables [-t <tabla>] -D <cadena> <condición> <acción>`
`iptables [-t <tabla>] -D <cadena> <numregla>`
borra una regla de una cadena de una tabla dada su especificación o dado su número de regla.

`iptables [-t <tabla>] -R <cadena> <numregla> <condición> <acción>` reemplaza la regla número `numregla` de una cadena por una nueva regla.
`iptables [-t <tabla>] -I <cadena> <numregla> <condición> <acción>` inserta una regla en la posición `numregla` en una cadena de una tabla.

iptables: condiciones

- Condiciones:

Interfaz	`-i <interfaz>`: interfaz de entrada `-o <interfaz>`: interfaz de salida
Dirección IP	`-s <dirIP[/máscara]>`: dirección (o direcciones) origen `-d <dirIP[/máscara]>`: dirección (o direcciones) destino
Protocolo	`-p <protocolo>` Se pueden especificar adicionalmente números de puerto: `-p <protocolo> --sport <puerto|puertoInicio:puertoFin>`: puerto origen `-p <protocolo> --dport <puerto|puertoInicio:puertoFin>`: puerto destino
Estado de la conexión[3]	`-m state --state <estado>` situación de un paquete con respecto a la conexión a la que pertenece. Estado: `INVALID`: no pertenece a una conexión existente `ESTABLISHED`: es parte de una conexión existente con paquetes en ambos sentidos `NEW`: es parte de una nueva conexión que aún no está establecida `RELATED`: está relacionado con otra conexión ya existente Ejemplo: un mensaje ICMP de error
Flags TCP	`-p tcp --syn`: segmento SYN
	`-p tcp --tcp-flag <flagsAComprobar> <flagsQueDebenEstarActivados>` flags: `SYN, FIN, ACK, RST, PSH, URG, ALL, NONE` Ejemplo: `-p tcp --tcp-flags ALL SYN,ACK` (deben estar activados `SYN, ACK` y desactivados `FIN, RST, PSH, URG`)

- La negación de una condición se expresa anteponiendo el caracter `!` al valor de la condición. Ejemplos:

`-p tcp --sport ! 80`	protocolo TCP y puerto origen distinto del 80
`-p ! icmp`	protocolo distinto de icmp

[3] "conexión" en sentido amplio

iptables: acciones (I)

La acción se especifica empezando con `-j`

Tabla `filter`	`-j ACCEPT` se acepta el paquete
	`-j DROP` se descarta el paquete
	`-j REJECT [--reject-with <tipo>]` se rechaza el paquete, informando al origen con un ICMP, se puede especificar el tipo de ICMP, por defecto `icmp-port-unreachable`
Tabla `nat`	`-j SNAT --to-source [<dirIP>][:<puerto>]` Realiza *Source NAT* sobre los paquetes salientes (es decir, se cambia dirección IP y/o puerto origen). Sólo se puede realizar en la cadena `POSTROUTING`. **NOTA: esta regla hace que también se cambie automáticamente la dirección de destino del tráfico entrante de respuesta al saliente de la misma "conexión".**
	`-j DNAT --to-destination [<dirIP>][:<puerto>]` Realiza *Destination NAT* sobre los paquetes entrantes (es decir, se cambia dirección IP y/o puerto destino). Sólo se puede realizar en la cadena `PREROUTING`. Esta regla sólo es necesaria para "abrir puertos", es decir, permitir tráfico entrante nuevo. **NOTA: esta regla hace que también se cambie automáticamente la dirección de origen del tráfico saliente de respuesta al entrante de la misma "conexión".**

iptables: acciones (II)

Todas las tablas	`-j LOG [--log-prefix <texto>]` se guarda información de ese paquete en el fichero de `/var/log/kern.log` anteponiendo la cadena de caracteres `<texto>` y **se continúa con la siguiente regla de la cadena**
	`-j <cadena-de-usuario>` Salta a aplicar al paquete las reglas de una cadena definida por el usuario. **Si termina esa cadena sin cumplirse la condición de ninguna de sus reglas, continuará en la cadena desde la que se saltó, por la regla siguiente a la que hizo la llamada.**

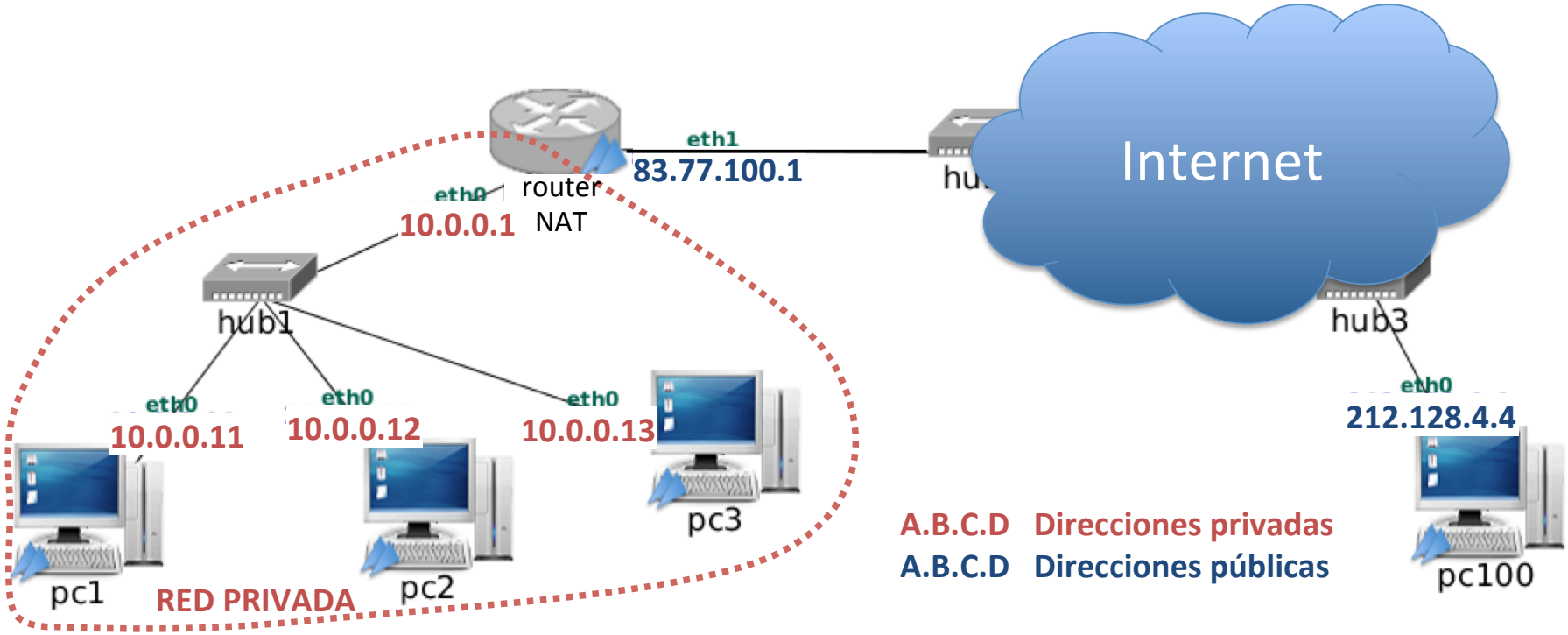
Si en una regla **no se especifica ninguna acción** (no hay cláusula `-j`), si se cumple la condición se actualizan los contadores de paquetes y bytes para la regla, pero **se continúa aplicando la siguiente regla de la cadena para ese paquete**.

Contenidos

NAT (*Network Address Translation*)

- Se denomina NAT a la reescritura por parte de un *router* de algunos campos de la cabecera de los paquetes que encamina:
  - cambia dirección IP origen y puerto origen en el tráfico saliente
  - cambia dirección IP destino y puerto destino en el tráfico entrante
- Esta técnica se desarrolla con el propósito principal de paliar la escasez de direcciones IP: Gracias al NAT, una organización puede usar direcciones privadas internamente, y tener una sola dirección IP global (pública) en el *router* que le da acceso a Internet.
- Pero la técnica ha tenido también el objetivo secundario de la seguridad: Un *router* NAT es (también) un *firewall* básico.
- Direcciones IP para redes «privadas»:
  - De `10.0.0.0` a `10.255.255.255`: 1 red de clase A
  - De `172.16.0.0` a `172.31.255.255`: 16 redes de clase B
  - De `192.168.0.0` a `192.168.255.255`: 256 redes de clase clase C

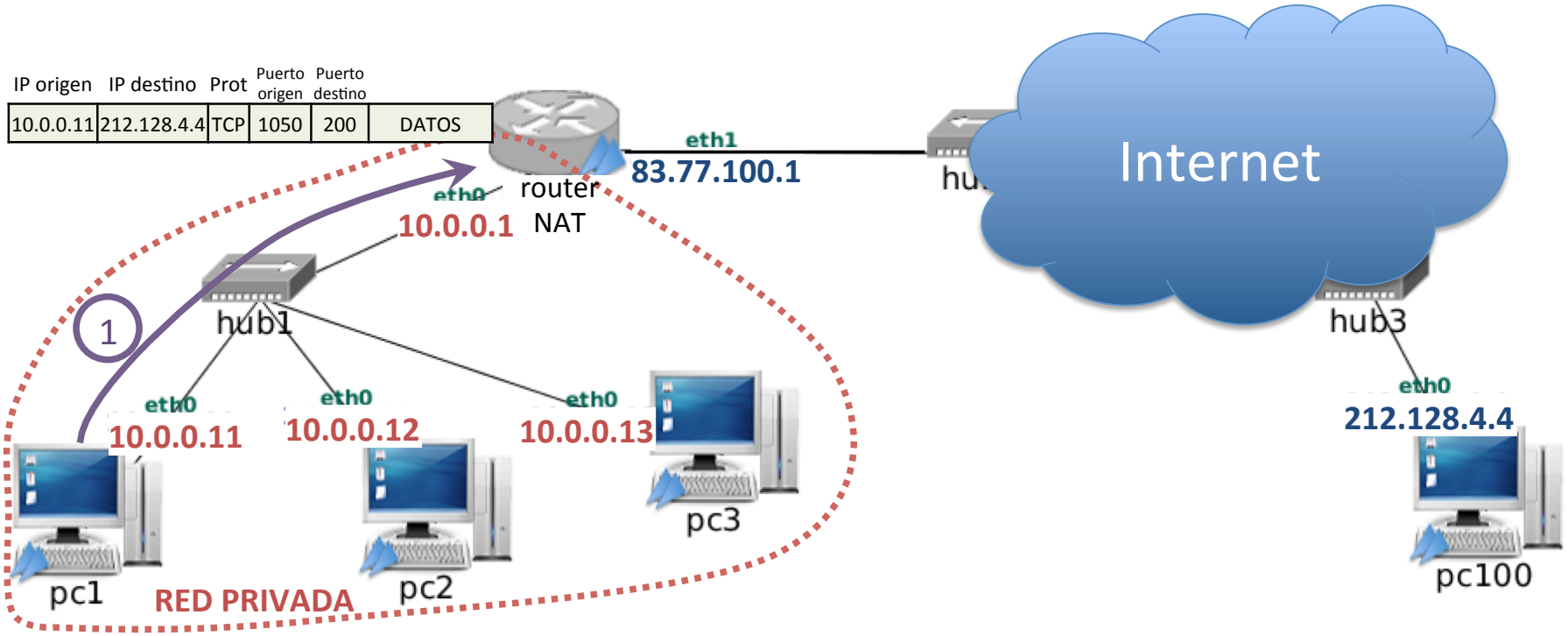
Direccionamiento



- Todos los ordenadores de la red interna utilizan direcciones «privadas», que no son válidas en Internet.
- El router que da acceso a Internet tiene una dirección «pública», válida en Internet.
  - En algunos casos, el router podría disponer de más de una IP pública. Siempre que haya disponibles menos IPs públicas que máquinas en la red interna habrá que utilizar NAT.

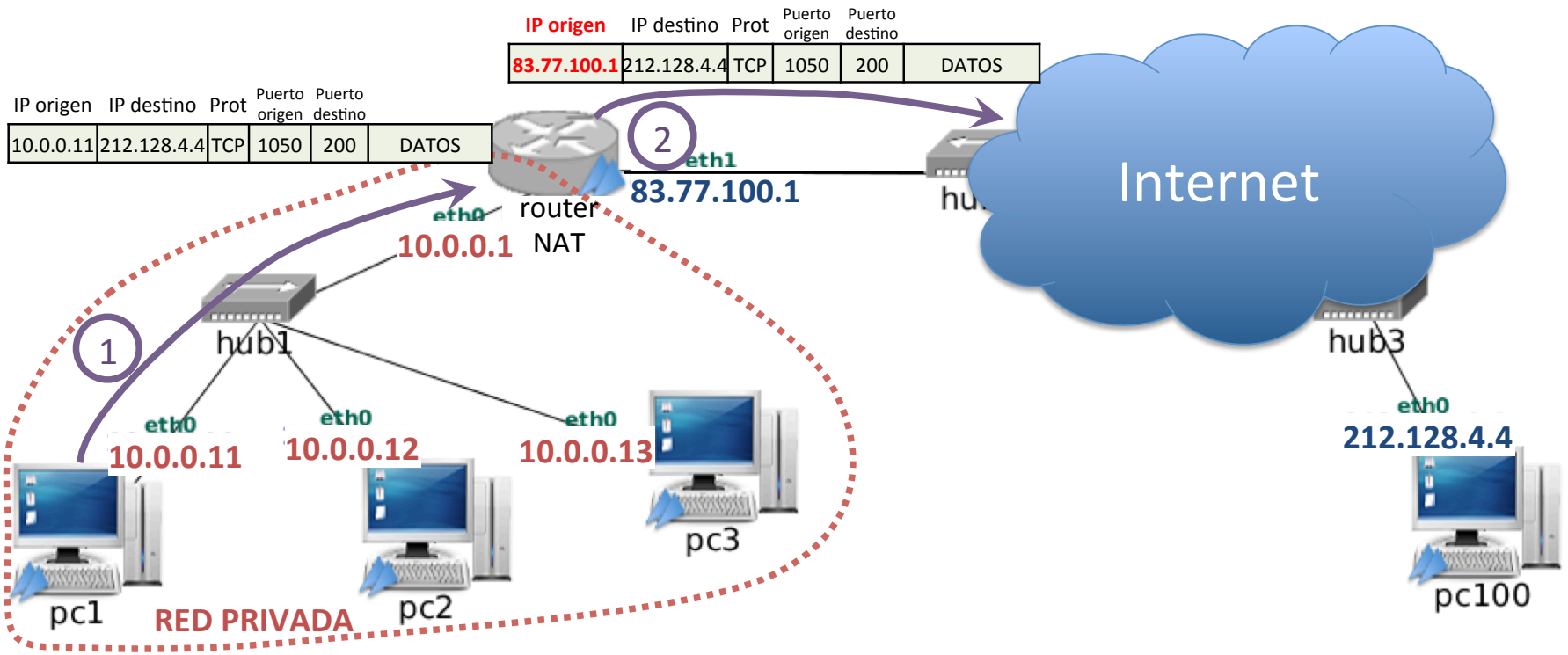
Contenidos

Tráfico saliente, mecanismo básico (1/3)



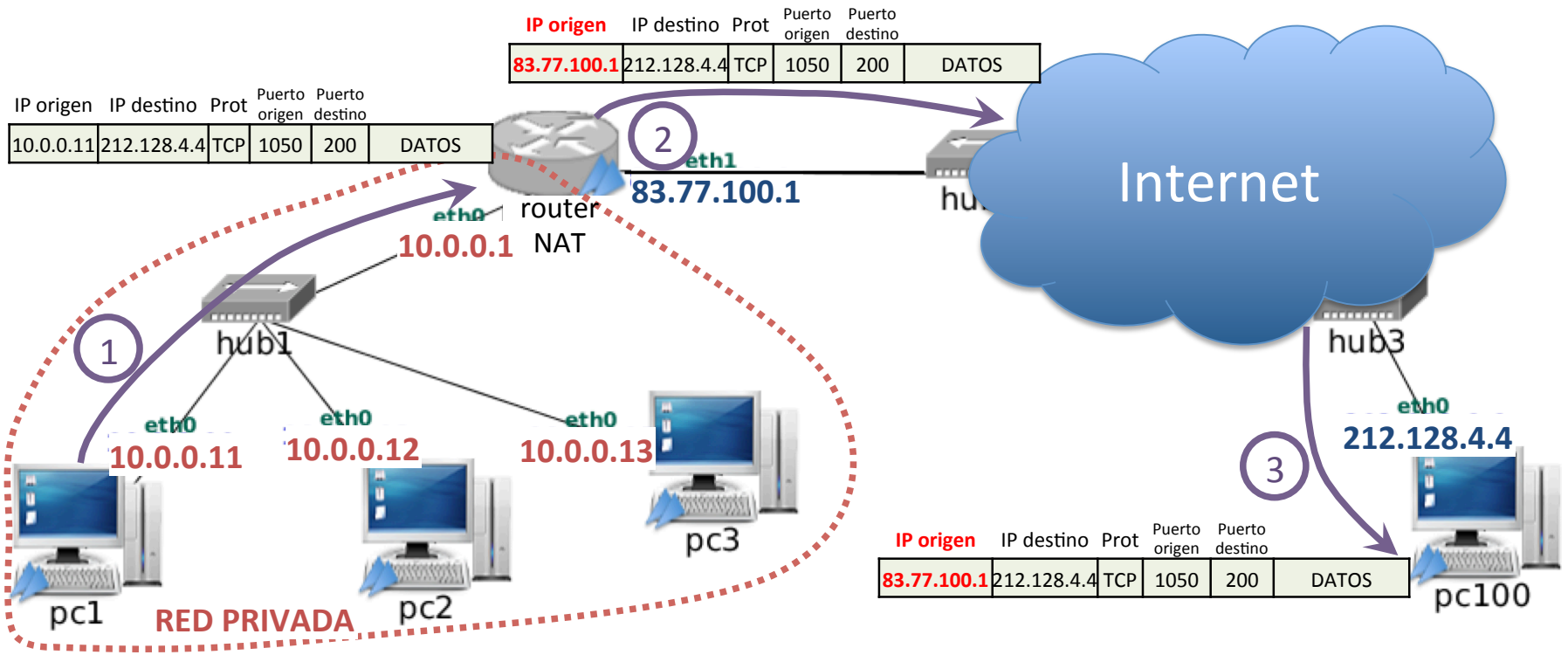
- La máquina pc1 envía un datagrama IP dirigido a pc100
- pc1 usa como IP origen su dirección IP privada, que no es válida en Internet (pues los *routers* de Internet no tienen rutas hacia esas direcciones).

Tráfico saliente, mecanismo básico (2/3)



- El router NAT cambia la **IP de origen** sustituyendo la IP privada de la máquina que creó el datagrama (`10.0.0.11`) por la IP pública del router (`83.77.100.1`).
- El seguimiento de conexiones del router NAT tendrá en cuenta el cambio para hacerlo a la inversa en el tráfico de respuesta.

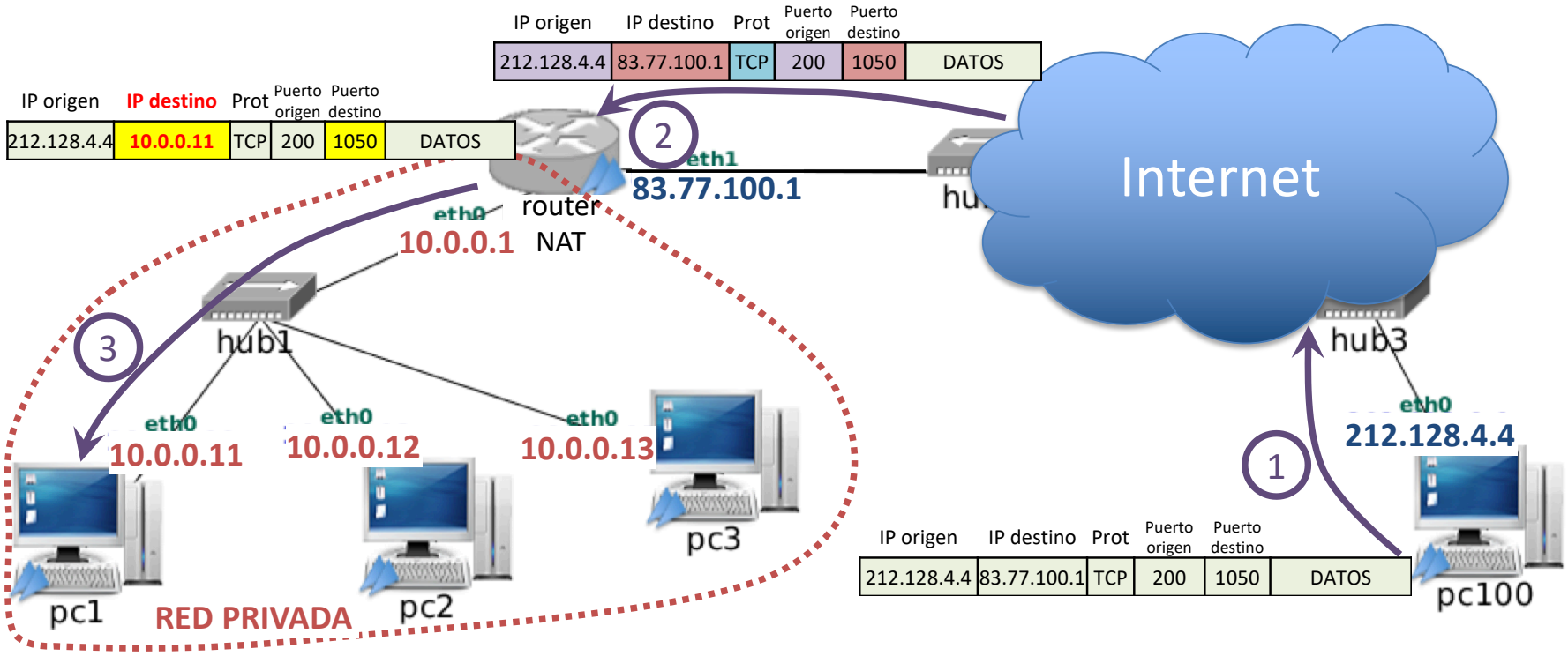
Tráfico saliente, mecanismo básico (3/3)



- La máquina que recibe el datagrama cree que el origen del mismo es el propio router NAT.

Contenidos

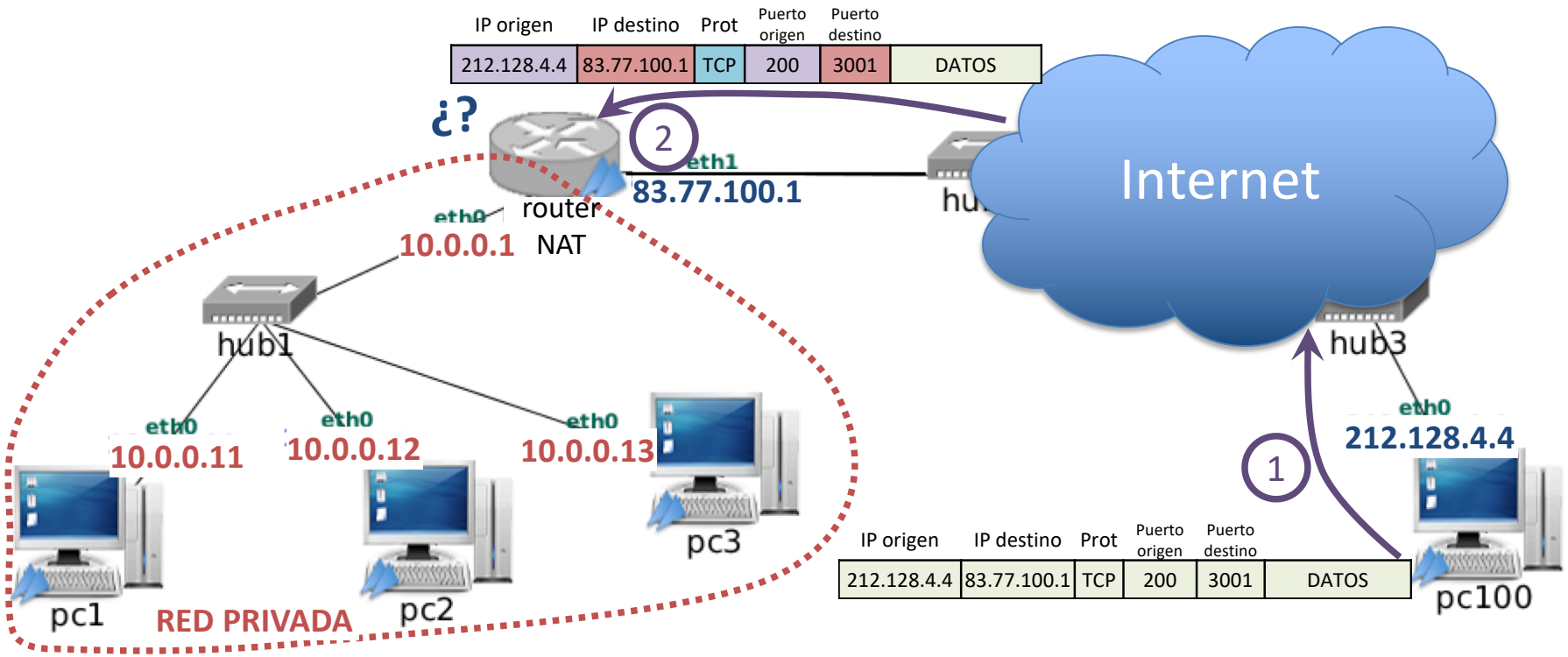
Tráfico entrante que responde al saliente



- El router NAT reenvía a la red privada el datagrama cambiando los campos de destino que ha recibido (dirección IP y puerto del router NAT `IP=83.77.100.1, puerto=1050`) por los que extrae del seguimiento de conexiones.

Contenidos

El problema del tráfico entrante nuevo



- Cuando llega al router NAT el datagrama, el router NAT no tiene forma de saber a qué máquina interna redirigirlo, pues no el seguimiento de conexiones no reconoce a dicho tráfico
- En este caso la configuración por defecto de un router NAT es descartar el datagrama recibido (*firewall*)

Tráfico entrante nuevo: abrir puertos

- **Solución**: En la configuración del router NAT se añade **a priori** una regla de traducción de direcciones para el tráfico entrante, reenviándolo a una máquina interna concreta.
- Para elegir a qué máquina concreta se reenvía se utiliza como criterio el **puerto de destino** del tráfico entrante,
- Esta solución se conoce informalmente como **"abrir puertos"** en el router NAT.
- Si en la red privada hubiera otro servidor que utilizara el mismo puerto, sería necesario al abrir el nuevo puerto usar un puerto público diferente para el segundo servidor.
- Nótese que desde los clientes de Internet debe saberse a priori qué puertos públicos se están usando para los servidores que haya en la red privada.

Contenidos

Contenidos

Ejemplos de traducción de direcciones IP y puertos con iptables (I)

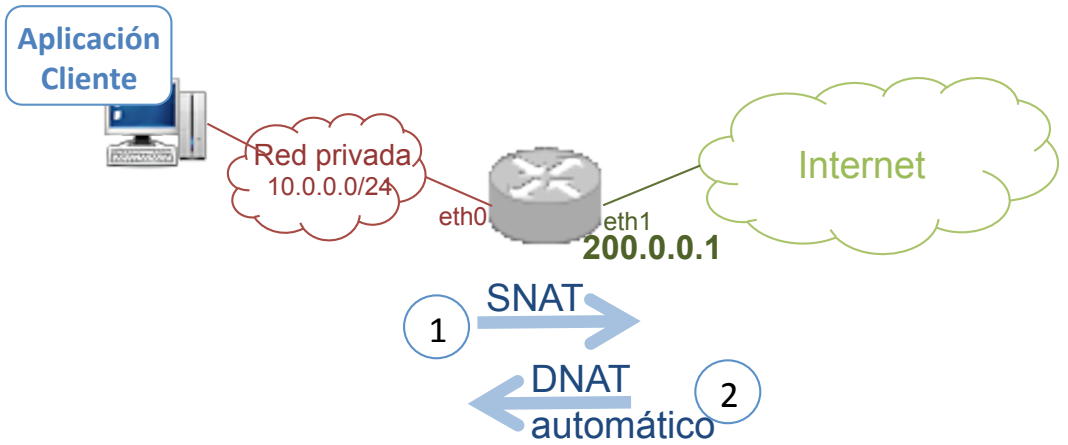
- **Inicialización**
  - Borrar las reglas y reiniciar los contadores:

```
iptables -t nat -F
iptables -t nat -Z
```

Ejemplos de traducción de direcciones IP y puertos con iptables (II)

- **Source NAT**
  - Modificar la dirección IP origen de los datagramas IP al salir de una red privada (10.0.0.0/24) a través de la interfaz de salida (`eth1`) de un router NAT. Todos los datagramas llevarán la dirección IP pública del router NAT (200.0.0.1):

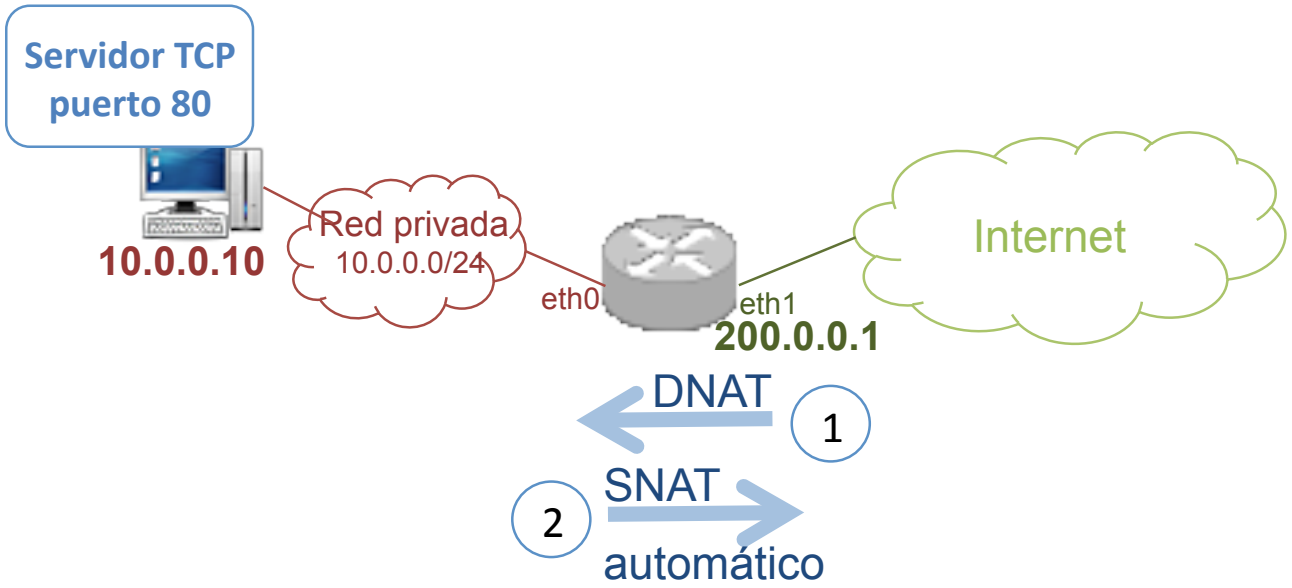
```
iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o eth1 \
         -j SNAT --to-source 200.0.0.1
```



Ejemplos de traducción de direcciones IP y puertos con iptables (III)

- **Destination NAT**
  - Modificar la dirección IP destino y puerto destino de los segmentos TCP al entrar dentro de una red privada (10.0.0.0/24). Los segmentos van dirigidos inicialmente a la dirección IP del router NAT (200.0.0.1) y puerto 8080, recibiéndose en su interfaz (`eth1`). Antes de comprobar la tabla de encaminamiento (PREROUTING) se modificará su dirección IP destino a 10.0.0.10 y puerto destino 80.

```
iptables -t nat -A PREROUTING -i eth1 -d 200.0.0.1 \
   -p tcp --dport 8080 -j DNAT --to-destination 10.0.0.10:80
```



Ejemplos de traducción de direcciones IP y puertos con iptables (IV)

- Al consultar las reglas de todas las cadenas de la tabla nat:

```
r1:~# iptables -t nat -L -v --line-numbers -n
  Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source      destination
1    0    0    DNAT    tcp  --  eth0 any   anywhere    200.0.0.1   tcp dpt:8080 to:10.0.0.10:80

  Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source      destination
1    0    0    SNAT    all  --  any eth1  10.0.0.0/24 anywhere    to:200.0.0.1

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt  in  out   source  destination
```

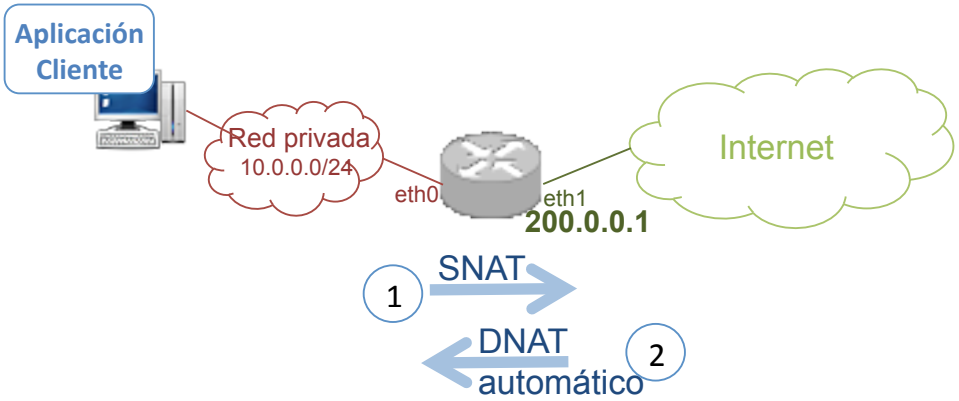
La opción `-line-numbers` imprime al principio de cada regla, la posición en la que se encuentra la regla dentro de la cadena (columna `num`).

La opción `-n` se utiliza para que `iptables` no realice una resolución de DNS inversa de las direcciones IP que haya configuradas en sus reglas.

Prueba1: Conexión con un servidor en Internet desde la máquina interna (I)

- Nada más cargar la configuración del firewall, si desde el pc 10.0.0.10 se inicia una conexión TCP con la máquina en Internet (100.0.0.100) y puerto 13 en la que hay varios paquetes intercambiados, al mostrar la tabla nat, se observa que sólo se ha utilizado la regla SNAT (una sola vez):

```
r1:~# iptables -t nat -L -v --line-numbers -n
  Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source      destination
1    0    0     DNAT    tcp  --  eth0 any   anywhere    200.0.0.1   tcp dpt:8080 to:10.0.0.10:80

  Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source      destination
1    1    60    SNAT    all  --  any eth1  10.0.0.0/24 anywhere    to:200.0.0.1

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source  destination
```

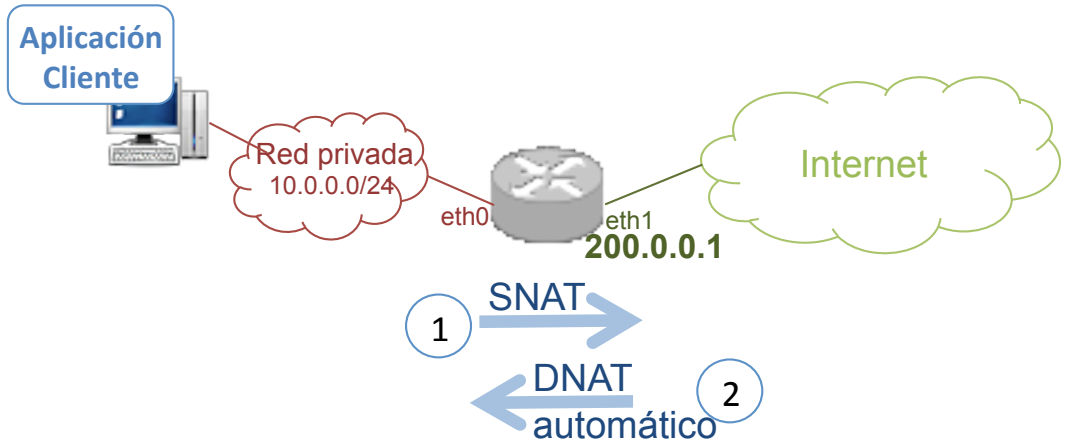


No se ejecuta la regla DNAT, el cambio de la dirección IP destino para los paquetes de entrada es automático.

Prueba1: Conexión con un servidor en Internet desde la máquina interna (II)

- Si consultamos la información del seguimiento de conexiones puede verse como se han contabilizado:
  - 4 paquetes de salida: desde la máquina interna 10.0.0.10 con destino a la máquina en Internet 100.0.0.100. El firewall debe modificar la dirección IP origen (SNAT).
  - 4 paquetes de entrada: desde la máquina en Internet 100.0.0.100 con destino al firewall 200.0.0.1. El firewall debe modificar la dirección IP destino (DNAT automático).

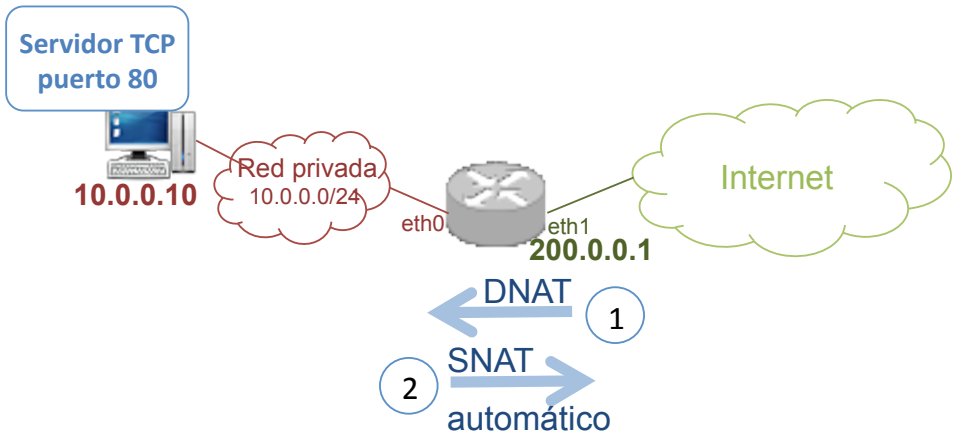
```
r1:~# cat /proc/net/ip_conntrack
tcp      6 98 TIME_WAIT src=10.0.0.10 dst=100.0.0.100 sport=38323 dport=13 packets=4 bytes=216
            src=100.0.0.100 dst=200.0.0.1 sport=13 dport=38323 packets=4 bytes=242 [ASSURED]
```



Prueba2: Conexión con el servidor TCP en 10.0.0.10 puerto 80 desde Internet (I)

- Nada más cargar la configuración del firewall, si desde Internet se inicia una conexión TCP con la máquina 200.0.0.1 y puerto 8080 en la que hay varios paquetes intercambiados, al mostrar la tabla nat, se observa que sólo se ha utilizado la regla DNAT (una sola vez):

```
r1:~# iptables -t nat -L -v --line-numbers -n
  Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source      destination
1    1    60    DNAT    tcp  --  eth0 any   anywhere    200.0.0.1   tcp dpt:8080 to:10.0.0.10:80

  Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source      destination
1    0    0     SNAT    all  --  any eth1  10.0.0.0/24 anywhere    to:200.0.0.1

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
num  pkts bytes target  prot opt  in  out   source  destination
```

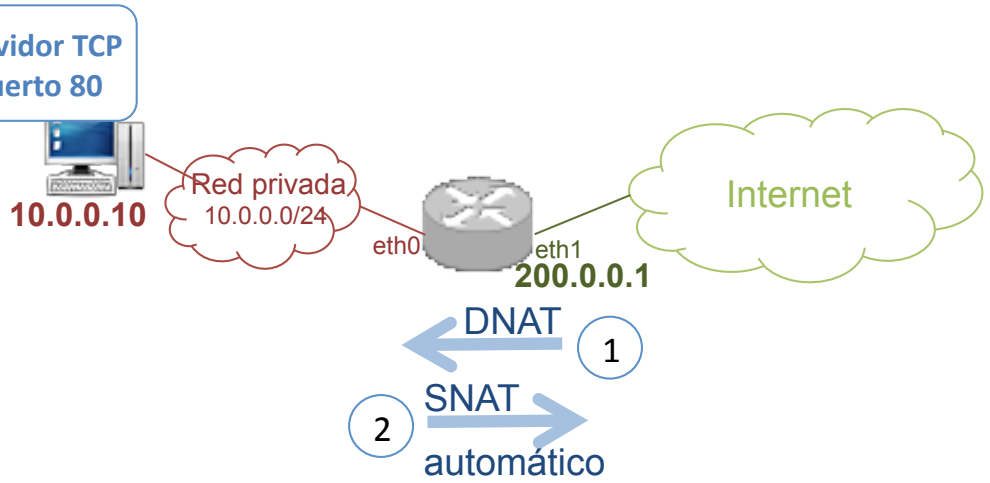


No se ejecuta la regla SNAT, el cambio de la dirección IP origen para los paquetes de salida es automático.

Prueba2: Conexión con el servidor TCP en 10.0.0.10 puerto 80 desde Internet (II)

- Si consultamos la información del seguimiento de conexiones puede verse como se han contabilizado:
  - 4 paquetes de entrada: desde la máquina en Internet 100.0.0.100 con destino el firewall 200.0.0.1. El firewall debe modificar la dirección IP destino y puerto (DNAT).
  - 4 paquetes de salida: desde la máquina interna 10.0.0.10 con destino en Internet 100.0.0.100. El firewall debe modificar la dirección IP origen (SNAT automático).

```
r1:~# cat /proc/net/ip_conntrack
tcp      6 98 TIME_WAIT src=100.0.0.100 dst=200.0.0.1 sport=32775 dport=8080 packets=4 bytes=216
            src=10.0.0.10 dst=100.0.0.100 sport=80 dport=32775 packets=4 bytes=242 [ASSURED]
```



Contenidos

Ejemplos de configuración de filtrado con iptables (I)

1. **Inicialización**
   - Borrar las reglas y reiniciar los contadores:

```
iptables -t filter -F
iptables -t filter -Z
```

   - Definir las políticas por defecto: Descartar cualquier cosa salvo paquetes de salida:

```
iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD DROP
iptables -t filter -P OUTPUT ACCEPT
```

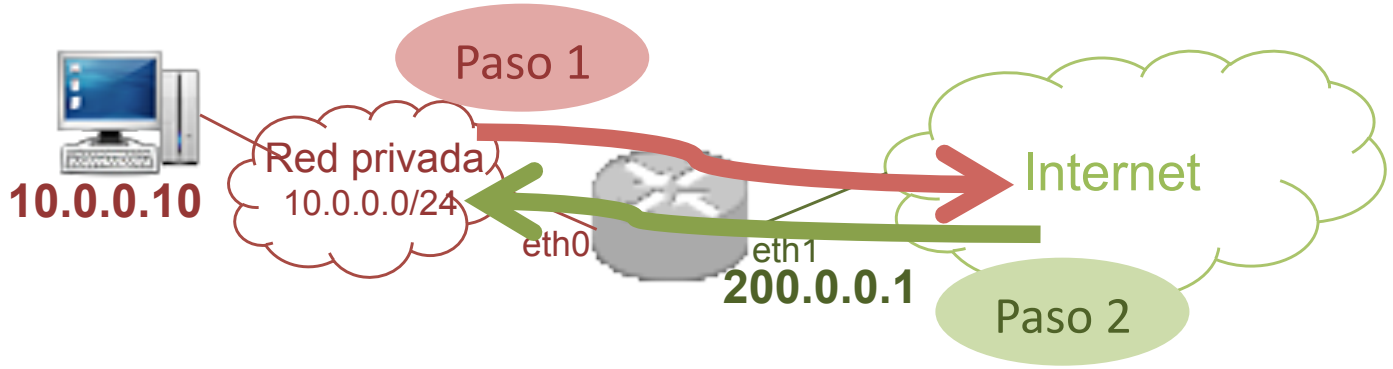
Ejemplos de configuración de filtrado con iptables (II)

2. **Filtrado: permitir cualquier tráfico saliente de la red privada y el tráfico entrante de respuesta**
   - Permitir el reenvío de todos los paquetes que se reciben en un router a través de una interfaz (eth0) para que se envíen a través de otra interfaz (eth1) (por ejemplo, permitir tráfico saliente de una organización):

```
iptables -t filter -A FORWARD -i eth0 -o eth1 -j ACCEPT
```

   - Permitir el reenvío paquetes entrantes que pertenezcan a "conexiones" ya existentes:

```
iptables -t filter -A FORWARD -i eth1 -o eth0 -m state \
         --state RELATED,ESTABLISHED -j ACCEPT
```



Ejemplos de configuración de filtrado con iptables (III)

- **Mostrar la información de una configuración**
  Se ha enviado un paquete ICMP *echo request* desde la red privada a Internet y se ha recibido respuesta. La información muestra la cantidad de paquetes a los que se les ha aplicado cada regla.

```
r1:~# iptables -t filter -L -v -n
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt  in  out   source  destination

Chain FORWARD (policy ACCEPT 4 packets, 336 bytes)
 pkts bytes target  prot opt  in  out   source  destination
 1    84    ACCEPT  all  --  eth0 eth1 anywhere  anywhere
 1    84    ACCEPT  all  --  eth1 eth0 anywhere  anywhere  state RELATED,ESTABLISHED

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target  prot opt  in  out   source  destination
```

Ejemplos de configuración de filtrado con iptables (IV)

- **Consultar el sistema de seguimiento ip_conntrack**
  El sistema de seguimiento no muestra ninguna información porque hay una anotación del paquete ICMP *echo request* pero en cuanto se recibe el paquete ICMP *echo reply* se borra la información de dicha "conexión". Por tanto, el siguiente comando no muestra ninguna información.

```
r1:~# watch -n 1 cat /proc/net/ip_conntrack
```

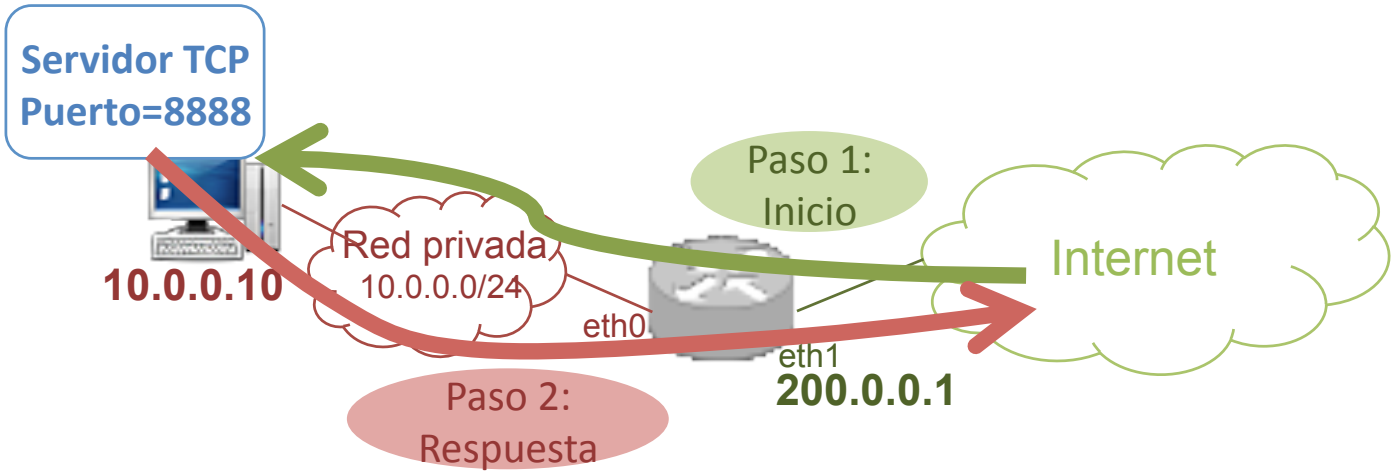
Nótese que si la máquina a la que se dirige el paquete ICMP *echo request* no responde, el sistema de seguimiento tendría anotado durante un tiempo el paquete ICMP echo request:

```
r1:~# watch -n 1 cat /proc/net/ip_conntrack

icmp 1 28 src=11.0.0.10 dst=12.0.0.56 type=8 code=0 id=11023 packets=1 bytes=84 [UNREPLIED]
          src=12.0.0.56 dst=11.0.0.10 type=0 code=0 id=11023 packets=0 bytes=0 mark=0 use=2
```

Ejemplos de configuración de filtrado con iptables (V)

3. **Filtrado: permitir tráfico tcp entrante en la red privada**
   - Permitir el paso de segmentos TCP de establecimiento de conexión de entrada dirigidos a una dirección IP de la red interna (10.0.0.10) y a un puerto (8888).

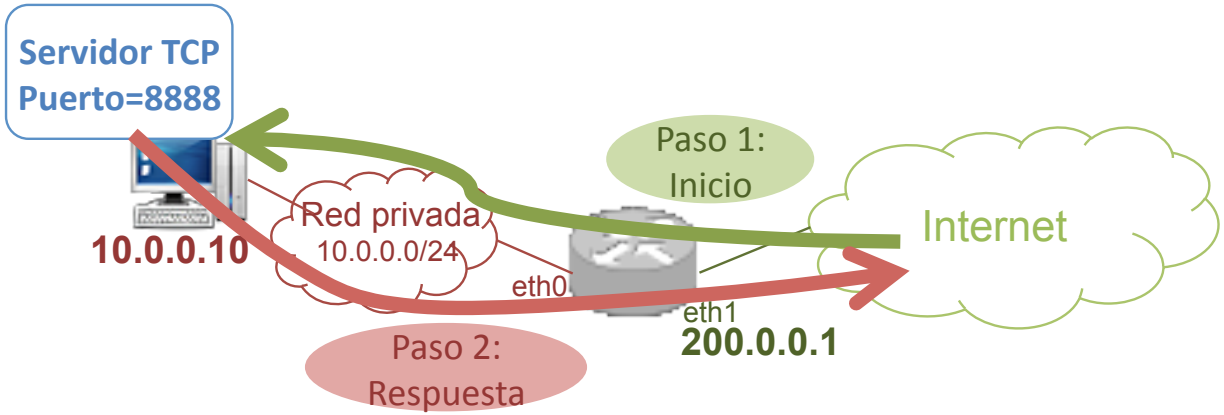
```
iptables -t filter -A FORWARD -d 10.0.0.10 \
      -p tcp --dport 8888 --syn -j ACCEPT
```



Ejemplos de configuración de filtrado con iptables (VI)

- Si añadimos esta regla a las anteriores:

```
iptables -t filter -A FORWARD -i eth0 -o eth1 -j ACCEPT

iptables -t filter -A FORWARD -i eth1 -o eth0 \
      -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -p tcp -d 10.0.0.10 --dport 8888 --syn -j ACCEPT
```



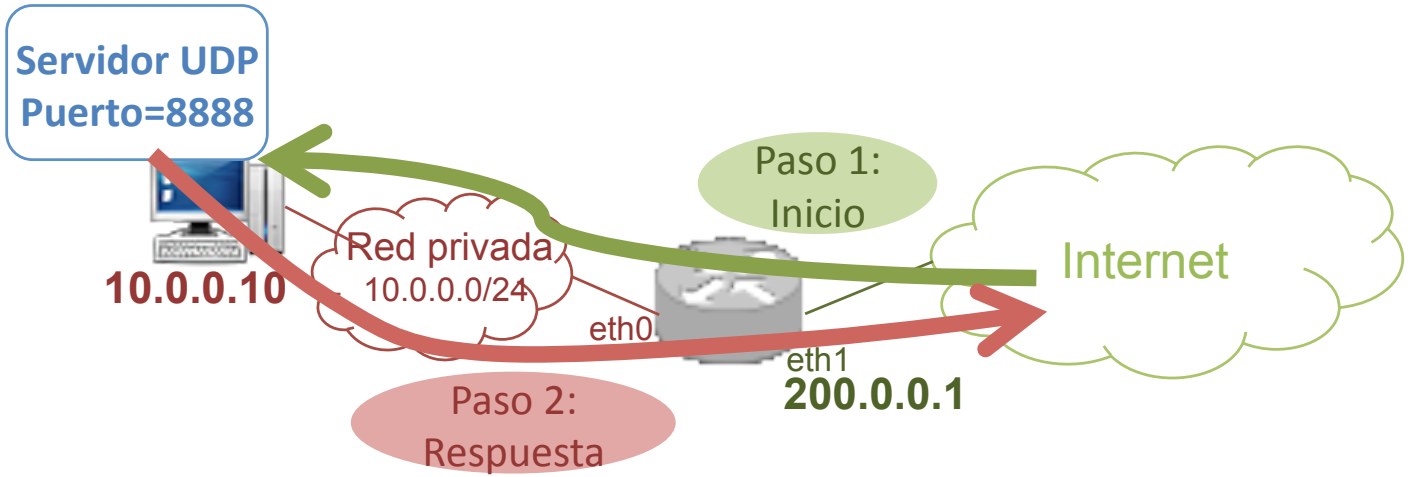
- Al mostrar el sistema de seguimiento ip_conntrack después del establecimiento de la conexión:

```
r1:~# watch -n 1 cat /proc/net/ip_conntrack

tcp 6 430970 ESTABLISHED src=12.0.0.10 dst=10.0.0.10 sport=58228 dport=8888 packets=2 bytes=112
   src=10.0.0.10 dst=12.0.0.10 sport=8888 dport=58228 packets=1 bytes=60 [ASSURED] mark=0 use=1
```

Ejemplos de configuración de filtrado con iptables (VII)

4. **Filtrado: permitir tráfico UDP entrante en la red privada**
   - Permitir el paso de datagramas UDP de entrada dirigidos a una dirección IP de la red interna (10.0.0.10) y a un puerto (8888).

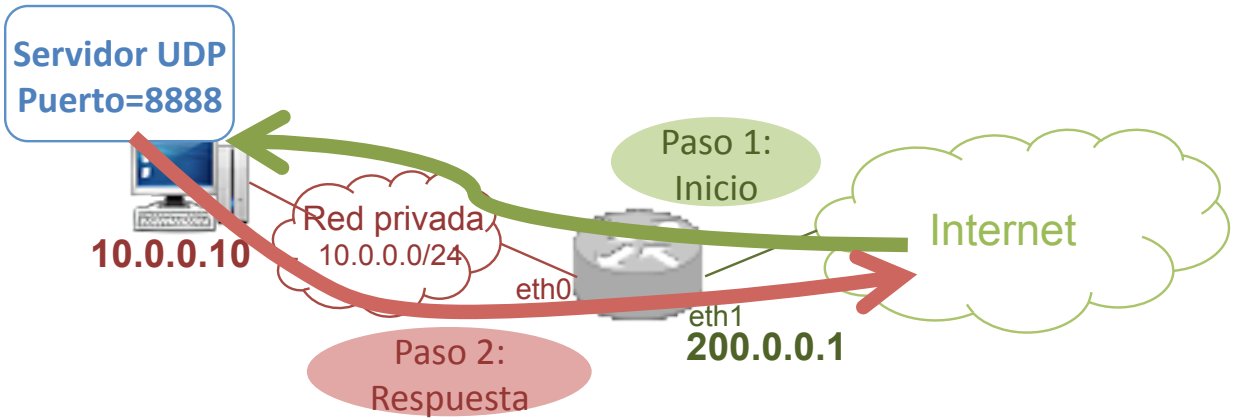
```
iptables -t filter -A FORWARD -d 10.0.0.10 \
    -p udp --dport 8888 -j ACCEPT
```



Ejemplos de configuración de filtrado con iptables (VIII)

- Si añadimos esta regla a las anteriores:

```
iptables -t filter -A FORWARD -i eth0 -o eth1 -j ACCEPT

iptables -t filter -A FORWARD -i eth1 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT

iptables -t filter -A FORWARD -p tcp -d 10.0.0.10 --dport 8888 --syn -j ACCEPT

iptables -t filter -A FORWARD -p udp -d 10.0.0.10 --dport 8888 -j ACCEPT
```



- Al mostrar el sistema de seguimiento ip_conntrack:

```
r1:~# watch -n 1 cat /proc/net/ip_conntrack

udp 17 21  src=12.0.0.10 dst=10.0.0.10 sport=58300 dport=8888 packets=1 bytes=34 [UNREPLIED]
   src=10.0.0.10 dst=12.0.0.10 sport=8888 dport=58300 packets=0 bytes=0 mark=0 use=1
```

Referencias

- Iptables Tutorial:
  `http://www.iptables.info`
- Linux Advanced Routing & Traffic Control:
  `http://www.lartc.org`