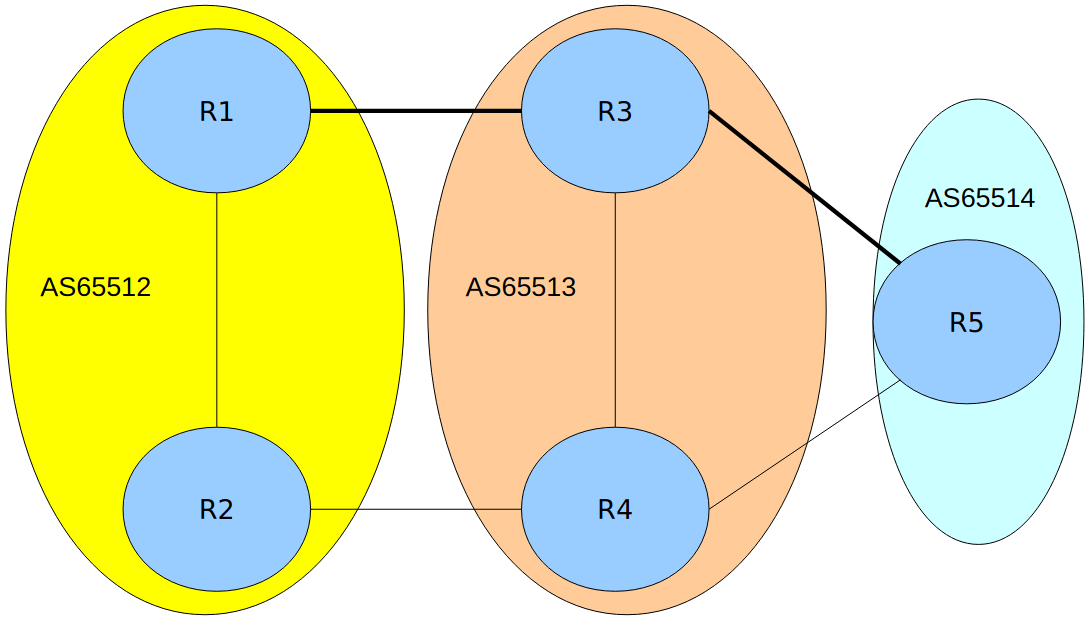
**OSPFv2 & OSPFv3 & BGP**



**Enunciado**

* Construye la arquitectura anterior atendiendo a los siguientes detalles:  
  + Las redes son de la forma 10.0.XY.0/24 y FDXY::/64
  + Las direcciones IPv4 serán de la forma 10.0.XY.<<UML\_ID>>/24
  + Los prefijos a anunciar 172.<<AS>>.0.0/24 y 2001:db8:<<AS>>::/48
  + A los enlaces entre los encaminadores hay que asignarles direcciones ULA (FD), esto se debe al atributo Next Hop (siguiente salto), para que los encaminadores sepan alcanzar el siguiente salto, que no va a estar conectados a ellos directamente.
  + Cuando un interfaz una un SA con otro, es importante que sea marcado como un interfaz pasivo.
  + Para no complicar la práctica añadiendo redes de usuario 172.12. y 2001:db8, vamos a anunciarlas pero sin llegar a configurarlas realmente.
  + No se puede hacer un traceroute desde R1 a R5, porque no se van a conocer, pero no pasa nada, porque lo importante es ver las rutas.
  + Los vecinos BGP son los externos que están conectados directamente, los internos no necesariamente.

**Solución**

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TOPOLOGÍA DE LA RED \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

mkdir uml{1..5}

kwrite net.conf

// net.conf

defsw sw12 uml1.0 uml3.2

defsw sw13 uml1.1 uml2.0

defsw sw24 uml2.1 uml4.2

defsw sw34 uml3.1 uml4.0

defsw sw35 uml3.0 uml5.1

defsw sw45 uml4.1 uml5.0

sudo ifovsdel

sudo ifovsparse net.conf

lanza {1..5}

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CONFIGURACIÓN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ejecutar en todas las máquinas:

* sed -i -e ‘s/ospfd=no/ospfd=yes/’ /etc/quagga/daemons
* sed -i -e ‘s/ospf6d=no/ospf6d=yes/’ /etc/quagga/daemons
* sed -i -e ‘s/bgpd=no/bgpd=yes/’ /etc/quagga/daemons
* service quagga restart “o” systemctl restart quagga

1º) UML1 → Realiza los anuncios de las redes, UML2 aprenderá de este.

vtysh

write → Para comprobar que los demonios estén activados.

ip forwarding

ipv6 forwarding

configure terminal  
interface eth0

ip address 10.0.13.1/24

ipv6 address fd13::1/64

ipv6 ospf6 passive

interface eth1

ip address 10.0.12.1/24

no shutdown

do write

exit

router ospf

router-id 0.0.0.1

passive-interface eth0 → Primero se establecen los eth pasivos para no emitir *Hello*.

network 10.0.13.0/24 area 0

network 10.0.12.0/24 area 0

do write

exit

router ospf6

router-id 0.0.0.1

interface eth0 area 0.0.0.0

interface eth1 area 0.0.0.0

do write

exit

router bgp 65512

neighbor 10.0.13.3 remote-as 65513 → eBGP; Porque habla con un AS distinto.

neighbor 10.0.12.2 remote-as 65512 → iBGP; Porque habla con alguien dentro del AS.

network 172.12.0.0/16 → UML2 la aprende al anunciarse; UML1 no, pues no existe un ethX con la red.

do write  
address-family ipv6

neighbor 10.0.13.3 activate

neighbor 10.0.12.2 activate

network 2001:db8:12::/48

do write

2º) UML2

vtysh

write → Para comprobar que los demonios estén activados.

ip forwarding

ipv6 forwarding

configure terminal

interface eth0

ip address 10.0.12.2/24

no shutdown

interface eth1

ip address 10.0.24.2/24

ipv6 address fd24::2/64

ipv6 ospf6 passive

do write

exit

router ospf

router-id 0.0.0.2

passive-interface eth1 → Primero se establecen los eth pasivos para no emitir *Hello*.

network 10.0.12.0/24 area 0

network 10.0.24.0/24 area 0

do write

exit

router ospf6

router-id 0.0.0.2

interface eth0 area 0.0.0.0

interface eth1 area 0.0.0.0

do write

exit

router bgp 65512

neighbor 10.0.12.1 remote-as 65512 → iBGP; Porque habla con alguien dentro del AS.

neighbor 10.0.24.4 remote-as 65513 → eBGP; Porque habla con un AS distinto.

do write

address-family ipv6

neighbor 10.0.12.1 activate

neighbor 10.0.24.4 activate

do write

3º) UML3

vtysh

write → Para comprobar que los demonios estén activados.

ip forwarding

ipv6 forwarding

configure terminal

interface eth0

ip address 10.0.35.3/24

ipv6 address fd35::3/64

ipv6 ospf6 passive

interface eth1

ip address 10.0.34.3/24

no shutdown

interface eth2

ip address 10.0.13.3/24

ipv6 address fd13::3/64

ipv6 ospf6 passive

do write

exit

router ospf

router-id 0.0.0.3

passive-interface eth0 → Primero se establecen los eth pasivos para no emitir *Hello*.

passive-interface eth2 → Primero se establecen los eth pasivos para no emitir *Hello*.

network 10.0.35.0/24 area 0

network 10.0.13.0/24 area 0

network 10.0.34.0/24 area 0

do write

exit

router ospf6

router-id 0.0.0.3

interface eth0 area 0.0.0.0

interface eth1 area 0.0.0.0

interface eth2 area 0.0.0.0

do write

exit

router bgp 65513

neighbor 10.0.13.1 remote-as 65512 → eBGP; Porque habla con un AS distinto.

neighbor 10.0.34.4 remote-as 65513 → iBGP; Porque habla con alguien dentro del AS.

neighbor 10.0.35.5 remote-as 65514 → eBGP; Porque habla con un AS distinto.

network 172.13.0.0/16 → UML2 la aprende al anunciarse; UML1 no, pues no existe un ethX con la red.

do write

address-family ipv6

neighbor 10.0.13.1 activate

neighbor 10.0.34.4 activate

neighbor 10.0.35.5 activate

network 2001:db8:13::/48

do write

4º) UML4

vtysh

write → Para comprobar que los demonios estén activados.

ip forwarding

ipv6 forwarding

configure terminal

interface eth0

ip address 10.0.34.4/24

no shutdown

interface eth1

ip address 10.0.45.4/24

ipv6 address fd45::4/64

ipv6 ospf6 passive

interface eth2

ip address 10.0.24.4/24

ipv6 address fd24::4/64

ipv6 ospf6 passive

do write

exit

router ospf

router-id 0.0.0.4

passive-interface eth1 → Primero se establecen los eth pasivos para no emitir *Hello*.

passive-interface eth2 → Primero se establecen los eth pasivos para no emitir *Hello*.

network 10.0.34.0/24 area 0

network 10.0.24.0/24 area 0

network 10.0.45.0/24 area 0

do write

exit

router ospf6

router-id 0.0.0.4

interface eth0 area 0.0.0.0

interface eth1 area 0.0.0.0

interface eth2 area 0.0.0.0

do write

exit

router bgp 65513

neighbor 10.0.24.2 remote-as 65512 → eBGP; Porque habla con un AS distinto.

neighbor 10.0.34.3 remote-as 65513 → iBGP; Porque habla con alguien dentro del AS.

neighbor 10.0.45.5 remote-as 65514 → eBGP; Porque habla con un AS distinto.

do write

address-family ipv6

neighbor 10.0.34.3 activate

neighbor 10.0.45.5 activate

neighbor 10.0.24.2 activate

do write

5º) UML5

vtysh

write → Para comprobar que los demonios estén activados.

ip forwarding

ipv6 forwarding

configure terminal

interface eth0

ip address 10.0.45.5/24

ipv6 address fd45::5/64

no shutdown → Introducir una IP ya levanta un interfaz, pero en realidad se haría.

ipv6 ospf6 passive

interface eth1

ip address 10.0.35.5/24

ipv6 address fd35::5/64

no shutdown → Introducir una IP ya levanta un interfaz, pero en realidad se haría.

ipv6 ospf6 passive

do write

exit

router bgp 65514

neighbor 10.0.35.3 remote-as 65513 → eBGP; Porque habla con un AS distinto.

neighbor 10.0.45.4 remote-as 65513 → eBGP; Porque habla con un AS distinto.

network 172.14.0.0/16 → UML2 la aprende al anunciarse; UML1 no, pues no existe un ethX con la red.

do write

address-family ipv6

neighbor 10.0.35.3 activate

neighbor 10.0.45.4 activate

network 2001:db8:14::/48

do write

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FILTRADO \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Queremos que los enlaces entre R1 y R3, y entre R3 y R5 sean prioritarios (principales), esto implica, por ejemplo, que R1 se comunicará con todo lo demás a través del enlace que tiene con R3, por lo que en este caso, todas las rutas van a pasar por R3. Cuando R4 se quiera comunicar con las redes de AS 65512, también va a pasar por R3 en contra de su criterio, que sería dirigir el tráfico por los caminos más cortos al destino. Esto es ingeniería del tráfico, queremos que las rutas pasen por un sitio determinado, y el resto estén a modo de backup (enlaces entre R2 y R4).

Tenemos que modificar los anuncios de dos formas para hacer esto:

* Incrementando el LOCAL\_PREF (Local Preference): indica la preferencia local hacia rutas externas. Se prefiere la ruta que tenga el valor más alto.
* Haciendo un Prepend, es decir, vamos a aumentar el AS\_PATH: incluye información sobre los AS que deben atravesarse para alcanzar el destino. Haremos pensar que la ruta hasta el destino es mucho más larga.

Procedimiento a seguir en R4:

end

configure terminal

route-map ruta-preferente-rm permit 10

set local-preference 200

exit

route-map ruta-secundaria-rm permit 10

set as-path prepend 65513 65513 65513 65513 65513

do write

do show running-config → Nos mostrará los dos route-map que hemos creado.

exit

router bgp 65513

neighbor 10.0.34.3 route-map ruta-preferente in → Esto se le aplica a los prefijos de entrada.

neighbor 10.0.24.2 route-map ruta-secundaria out → Esto se aplica a los anuncios de salida.

neighbor 10.0.45.5 route-map ruta-secundaria out → Esto se aplica a los anuncios de salida.

# Las rutas son las mismas (no hay cambios), esto es porque los anuncios al principio no  
# contaban con esta información.

do show ip route

do show ip bgp

address-family ipv6

neighbor 10.0.34.3 route-map ruta-preferente in

neighbor 10.0.45.5 route-map ruta-secundaria out

neighbor 10.0.24.2 route-map ruta-secundaria out

do write

# Cerramos sesión BGP con todos los vecinos, cuando se restablezca y se vuelva a  
# anunciar, se tendrán en cuenta los filtros.

do clear bgp ip \*

do clear bgp ipv6 \*

# Revisamos que los cambios han tenido lugar.

do show ip route

do show ip bgp

Procedimiento a seguir en R3:

# Veamos ahora si cortando el enlace primario entre R1 y R3 ocurre algo …

end

configure terminal

interface eth2

shutdown → Esperar 2 min (aprox) a que los cambios topológicos se completen.

Procedimiento a seguir en R1:

# Comprobar que ya no se hace uso del enlace que une R1 y R3.

do show ip route

do show ip bgp

Procedimiento a seguir en R3:

# Al levantar la interfaz nos encontramos con una *Wedgie*, es decir, un comportamiento

# inesperado, en este caso, el interfaz se ha restaurado pero no recuperamos las rutas que # teníamos anteriormente. Para volver a la situación anterior habríamos de apagar el

# interfaz que une R2 con R4, esto es porque R3 está encaminando por R4, entonces no

# anuncia a R1, y como este no recibe anuncios nuevos y no tiene forma de recuperarse.

end

configure terminal

interface eth2

no shutdown