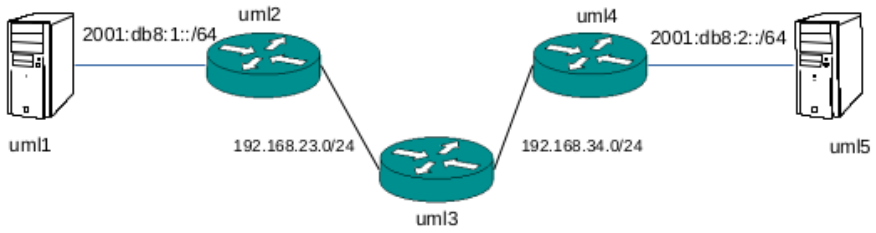
Índice

1. Túneles 6in4

Dado el siguiente escenario:




```
# net.conf
defsw br12 uml1.0 uml2.0
defsw br23 uml2.1 uml3.0
defsw br34 uml3.1 uml4.0
defsw br23 uml4.1 uml5.0

! UML1 y UML5
configure terminal
int eth0
no shutdown
end
write

! UML2
configure terminal
int eth0
```

```
ipv6 address 2001:db8:1::1/64
ipv6 nd prefix 2001:db8:1::/64 ! Para que anuncie su prefijo a UML1
no ipv6 nd suppress-ra
quit
int eth1
ip address 192.168.23.1/24
quit
ip route 0.0.0.0/0 192.168.23.2
ipv6 forwarding
ip forwarding
end
write
```

```
! UML3
configure terminal
int eth0
ip address 192.168.23.2/24
quit
int eth1
ip address 192.168.34.2/24
quit
ip forwarding
ip route 192.168.23.0/24 192.168.34.2
ip route 192.168.34.0/24 192.168.23.2
end
write
```

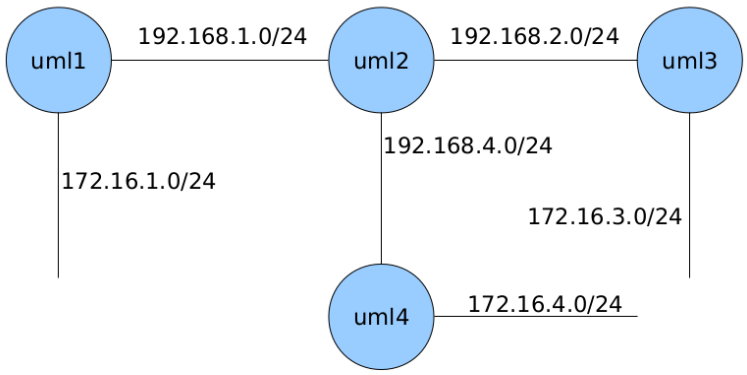
```
! UML4
configure terminal
int eth0
ip address 192.168.34.1/24
quit
int eth1
ipv6 address 2001:db8:2::1/64
ipv6 nd prefix 2001:db8:2::/64 ! Para que anuncie su prefijo a UML5
no ipv6 nd suppress-ra
quit
ip route 0.0.0.0/0 192.168.34.2
ipv6 forwarding
ip forwarding
end
write
```

```
# UML2 (bash)
ip tunnel add tunnel1 mode sit remote 192.168.34.1
ip link set dev tunnel1 up mtu 1400
ip route add 2001:db8:2::/64 dev tunnel1
```

```
# UML4 (bash)
ip tunnel add tunnel2 mode sit remote 192.168.23.1
ip link set dev tunnel2 up mtu 1400
ip route add 2001:db8:1::/64 dev tunnel2

# TEST
# probar desde vtysh en UML1 y UML2 que se anunció correctamente el prefijo
show ipv6 route

# TEST
# Desde UML2, probar un ping a UML4 y viceversa
ping -c 5 192.168.(34|23).1
# Desde UML1, probar un ping6 a UML5 y viceversa
ping6 -c 5 2001:db8:(1|2):ff:fe00:5f0
```

2. RIPv1

Crear la siguiente configuración de red y utilizar RIP en las máquinas uml
Comprobar las tablas de rutas y los mensajes intercambiados




```
  # net.conf
  defsw br12 uml1.0 uml2.1
  defsw net1 uml1.1
  defsw br23 uml2.0 uml3.0
  defsw br24 uml2.2 uml4.0
  defsw net4 uml4.1
  defsw net3 uml3.1

# En cuanto se inician las UML, editar el fichero /etc/quagga/daemons la línea
ripd=no
# Por
ripd=yes
```

```
# A continuación restart del servicio
systemctl restart quagga
# Verificar que se ospfd está corriendo
systemctl status quagga
```

```
! UML1 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip addr[ess] 192.168.1.1/24
no s[hutdown]
q[uit]
int[erface] eth1
ip addr[ess] 172.16.1.1/24
no s[hutdown]
q[uit]
ip f[orwarding]
end
w[rite]
```

```
! UML1 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip addr[ess] 192.168.1.1/24
no s[hutdown]
q[uit]
int[erface] eth1
ip addr[ess] 172.16.1.1/24
no s[hutdown]
q[uit]
ip f[orwarding]
end
w[rite]
```

```
! UML2 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip addr[ess] 192.168.2.1/24
no s[hutdown]
q[uit]
int[erface] eth1
ip addr[ess] 192.168.1.2/24
no s[hutdown]
q[uit]
int[erface] eth2
ip addr[ess] 192.168.4.2/24
no s[hutdown]
q[uit]
```

```
ip f[orwarding]
end
w[rite]

! UML3 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip addr[ess] 192.168.2.2/24
no s[hutdown]
q[uit]
int[erface] eth1
ip addr[ess] 172.16.3.1/24
no s[hutdown]
q[uit]
ip f[orwarding]
end
w[rite]

! UML4 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip addr[ess] 192.168.4.1/24
no s[hutdown]
q[uit]
int[erface] eth1
ip addr[ess] 172.16.4.1/24
no s[hutdown]
q[uit]
ip f[orwarding]
end
w[rite]

! UML1 (ripd.conf)
con[figure] t[erminal]
router rip
  network eth0
  network eth1
  neighbor 192.168.1.2
  passive-interface eth0
  passive-interface eth1
  do w[rite]

! UML2 (ripd.conf)
con[figure] t[erminal]
router rip
  network eth0
  network eth1
```

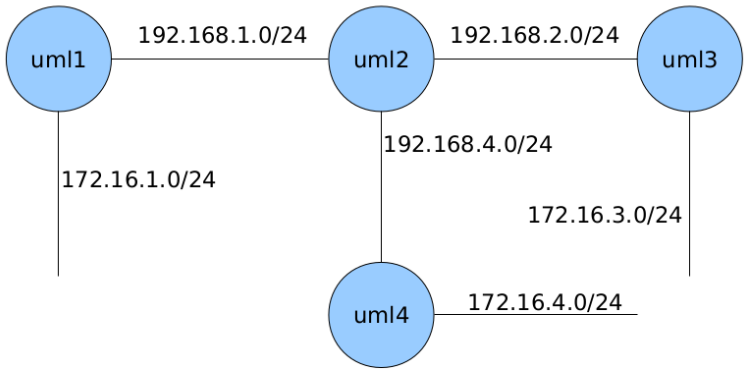
```
  network eth2
  neighbor 192.168.1.1
  neighbor 192.168.2.2
  neighbor 192.168.4.1
  passive-interface eth0
  passive-interface eth1
  passive-interface eth2
  do w[rite]

! UML3 (ripd.conf)
con[figure] t[erminal]
router rip
  network eth0
  network eth1
  neighbor 192.168.2.1
  passive-interface eth0
  passive-interface eth1
  do w[rite]

! UML4 (ripd.conf)
con[figure] t[erminal]
router rip
  network eth0
  network eth1
  neighbor 192.168.4.2
  passive-interface eth0
  passive-interface eth1
  do w[rite]
```

Pruebas:

```
sh[ow] ip rip
sh[ow] ip route
```

3. RIPng

Crear la siguiente configuración de red y utilizar RIP en las máquinas uml
Comprobar las tablas de rutas y los mensajes intercambiados




```
# net.conf
defsw br12 uml1.1 uml2.0
defsw net1 uml1.0
defsw br23 uml2.1 uml3.1
defsw br24 uml2.2 uml4.1
defsw net4 uml4.0
defsw net3 uml3.0
```

```
# En cuanto se inician las UML, editar el fichero /etc/quagga/daemons la línea
ripngd=no
# Por
ripngd=yes
# A continuación restart del servicio
systemctl restart quagga
# Verificar que se ospfd está corriendo
systemctl status quagga
```

```
! UML1 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ipv6 address 2001:db8:1::1/64
no s[hutdown]
q[uit]
int[erface] eth0
no s[hutdown]
q[uit]
ipv6 f[orwarding]
w[rite]
```

```
! UML2 (zebra.conf)
int[erface] eth0
no s[hutdown]
q[uit]
int[erface] eth1
no s[hutdown]
q[uit]
int[erface] eth2
no s[hutdown]
q[uit]

! UML3 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ipv6 address 2001:db8:3::1/64
no s[hutdown]
q[uit]
int[erface] eth1
no s[hutdown]
q[uit]
ipv6 f[orwarding]
w[rite]

! UML4 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ipv6 address 2001:db8:4::1/64
no s[hutdown]
q[uit]
int[erface] eth1
no s[hutdown]
q[uit]
ipv6 f[orwarding]
w[rite]

! UML1 (ripngd.conf)
con[figure] t[erminal]
router ripng
  network fe80::/64
  network eth0
  passive-interface eth0
  passive-interface fe80::/64
  do w[rite]

! UML2 (ripngd.conf)
con[figure] t[erminal]
router ripng
```

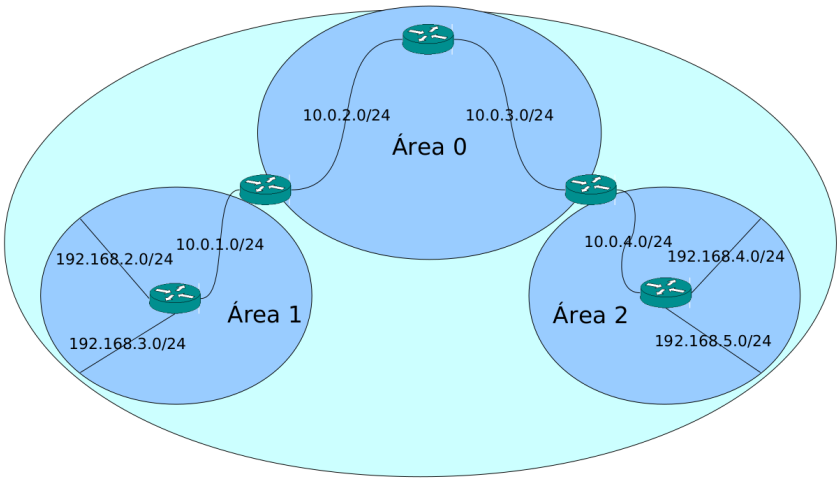
```
  network fe80::/64º
  passive-interface fe80::/64
  do w[rite]

! UML3 (ripngd.conf)
con[figure] t[erminal]
router ripng
  network fe80::/64
  network eth0
  passive-interface eth0
  passive-interface fe80::/64
  do w[rite]

! UML4 (ripngd.conf)
con[figure] t[erminal]
router ripng
  network fe80::/64
  network eth0
  passive-interface eth0
  passive-interface fe80::/64
  do w[rite]
```

Pruebas:

```
sh[ow] ipv6 rip
sh[ow] ipv6 route
```

4. OSPFv2

Configurar 5 máquinas virtuales para crear el siguiente AS:




```
# net.conf
defsw br12 uml1.0 uml2.0
defsw net1 uml1.1
defsw net2 uml1.2
defsw br23 uml2.1 uml3.0
defsw br34 uml3.1 uml4.0
defsw br45 uml4.1 uml5.0
defsw net3 uml5.1
defsw net4 uml5.2

# En cuanto se inician las UML, editar el fichero /etc/quagga/daemons la línea
ospfd=no
# Por
ospfd=yes
# A continuación restart del servicio
systemctl restart quagga
# Verificar que se ospfd está corriendo
systemctl status quagga

! UML1 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.1.1/24
no shutdown
quit
int[erface] eth1
ip address 192.168.3.1/24
```

```
no shutdown
quit
int[erface] eth2
ip address 192.168.2.1/24
no shutdown
quit
ip forwarding
exit
write

! UML2 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.1.2/24
no shutdown
quit
int[erface] eth1
ip address 10.0.2.2/24
no shutdown
quit
ip forwarding
exit
write

! UML3 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.2.1/24
no shutdown
quit
int[erface] eth1
ip address 10.0.3.1/24
no shutdown
quit
ip forwarding
exit
write

! UML4 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.3.2/24
no shutdown
quit
int[erface] eth1
ip address 10.0.4.2/24
no shutdown
```

```
quit
ip forwarding
exit
write

! UML5 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.4.1/24
no shutdown
quit
int[erface] eth1
ip address 192.168.4.1/24
no shutdown
quit
int[erface] eth2
ip address 192.168.5.1/24
no shutdown
quit
ip forwarding
exit
write

! UML1 ospf.conf
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.1
network 10.0.1.0/24 area 1
network 192.168.2.0/24 area 1
network 192.168.3.0/24 area 1
area 1 stub
passive-interface eth1 ! Las redes de usuario irán siempre como passive-interface
passive-interface eth2
end
write

! UML2 ospf.conf
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.2
network 10.0.1.0/24 area 1
network 10.0.2.0/24 area 0
! No advierte de este area en otras areas por ser ABR
area 1 range 10.0.0.0/8 not-advertise
end
write
```

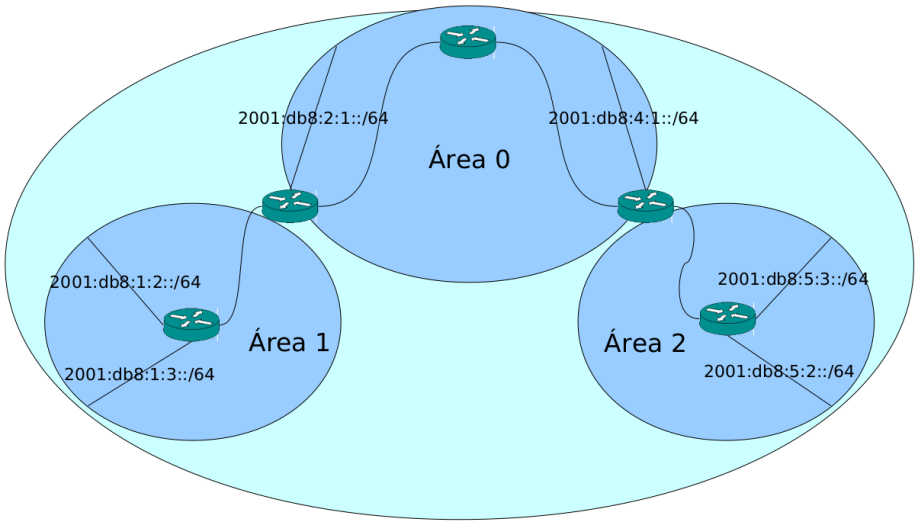
```
! UML3 ospf.conf
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.3
network 10.0.2.0/24 area 0
network 10.0.3.0/24 area 0

! UML4 ospf.conf
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.4
network 10.0.3.0/24 area 0
network 10.0.4.0/24 area 2
! No advierte de este area en otras areas por ser ABR
area 2 range 10.0.0.0/8 not-advertise
end
write

! UML5 ospf.conf
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.5
network 10.0.4.0/24 area 2
network 192.168.4.0/24 area 2
network 192.168.5.0/24 area 2
area 2 stub
passive-interface eth1 ! Las redes de usuario irán siempre como passive-interface
passive-interface eth2
end
write
```

Comprobar las tablas de ospf con el comando

```
show ip ospf database
```

5. OSPFv3

Configurar 5 máquinas virtuales para crear el siguiente AS:




```
# net.conf
defsw br12 uml1.0 uml2.0
defsw net1 uml1.1
defsw net2 uml1.2
defsw br23 uml2.2 uml3.0
defsw net3 uml2.1
defsw br34 uml3.1 uml4.0
defsw net4 uml4.2
defsw br45 uml4.1 uml5.0
defsw net5 uml5.1
defsw net6 uml5.2
```

```
 # En cuanto se inician las UML, editar el fichero /etc/quagga/daemons la línea
 ospf6d=no
 # Por
 ospf6d=yes
 # A continuación restart del servicio
 systemctl restart quagga
 # Verificar que se ospfd está corriendo
 systemctl status quagga
```

```
! UML1 zebra.conf
conf[igure] term[inal]
int[erface] eth0
no shutdown
quit
```

```
int[erface] eth1
ipv6 address 2001:db8:1:3::1/64
no shutdown
quit
int[erface] eth2
ipv6 address 2001:db8:1:2::1/64
no shutdown
quit
ipv6 forwarding
exit
write
```

```
! UML2 zebra.conf
conf[igure] term[inal]
int[erface] eth0
no shutdown
quit
int[erface] eth1
ipv6 address 2001:db8:2:1::1/64
no shutdown
quit
int[erface] eth2
no shutdown
quit
ipv6 f[orwarding]
exit
write
```

```
! UML3 zebra.conf
conf[igure] term[inal]
int[erface] eth0
no shutdown
quit
int[erface] eth1
no shutdown
quit
ipv6 f[orwarding]
exit
write
```

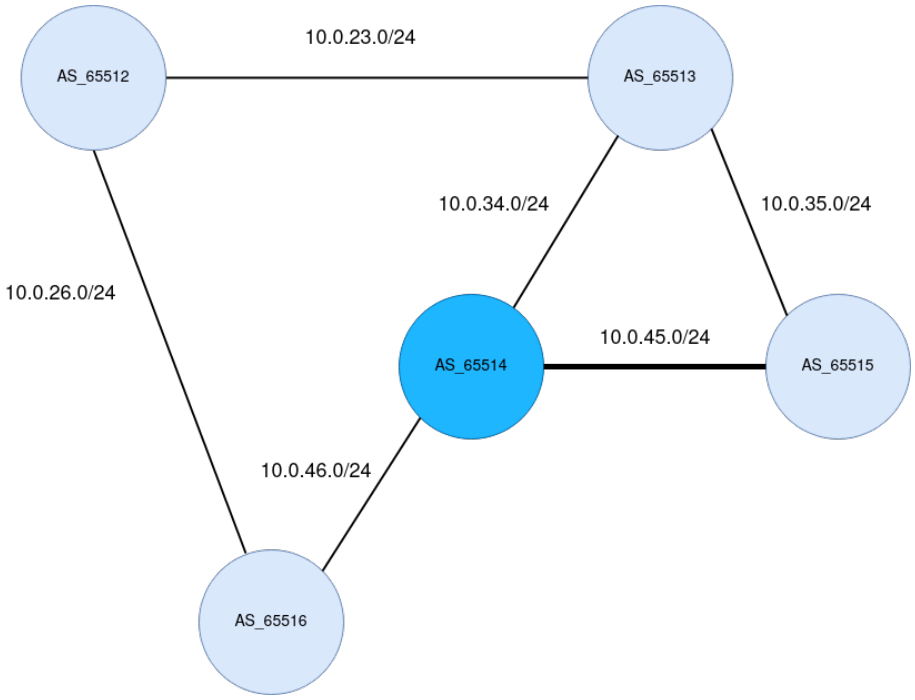
```
! UML4 zebra.conf
conf[igure] term[inal]
int[erface] eth0
no shutdown
quit
int[erface] eth1
no shutdown
```

```
quit
int[erface] eth2
ipv6 address 2001:db8:4:1::1/64
no shutdown
quit
ipv6 f[orwarding]
exit
write

! UML5 zebra.conf
conf[igure] term[inal]
int[erface] eth0
no shutdown
quit
int[erface] eth1
ipv6 address 2001:db8:5:3::1/64
no shutdown
quit
int[erface] eth2
ipv6 address 2001:db8:5:2::1/64
no shutdown
quit
ipv6 forwarding
exit
write

! UML1 ospf6d.conf
conf[igure] term[inal]
router ospf6
router-id 0.0.0.1
interface eth0 area 0.0.0.1
interface eth1 area 0.0.0.1
interface eth2 area 0.0.0.1
end
write

! UML2 ospf6d.conf
conf[igure] term[inal]
router ospf6
router-id 0.0.0.2
interface eth0 area 0.0.0.1
interface eth1 area 0.0.0.0
interface eth2 area 0.0.0.0
end
write

! UML3 ospf6d.conf
conf[igure] term[inal]
```

```
router ospf6
router-id 0.0.0.3
interface eth0 area 0.0.0.0
interface eth1 area 0.0.0.0
end
write

! UML4 ospf6d.conf
conf[igure] term[inal]
router ospf6
router-id 0.0.0.4
interface eth0 area 0.0.0.0
interface eth1 area 0.0.0.2
interface eth2 area 0.0.0.0
end
write

! UML5 ospf6d.conf
conf[igure] term[inal]
router ospf6
router-id 0.0.0.5
interface eth0 area 0.0.0.2
interface eth1 area 0.0.0.2
interface eth2 area 0.0.0.2
```

6. BGP. Filtrado de rutas

Dada la siguiente topología:



- El AS65514 desea usar preferentemente su enlace con AS65515, y mantener los otros dos como respaldo (backup)
- Implementar esa política mediante el uso de `route-map` .

Detalles a tener en cuenta:

- Redes de usuario:
  - IPv4: 172.16.X.0/24
  - IPv6: 2001:db8:X::/64

$$\forall X \in \{12, 13, 14, 15, 16\}$$

```
# net.conf
defsw br23 uml2.0 uml3.0
defsw br34 uml3.1 uml4.0
defsw br35 uml3.2 uml5.0
defsw br45 uml4.1 uml5.1
defsw br26 uml2.1 uml6.0
defsw br46 uml4.2 uml6.1
```

```
# En cuanto se inician las UML, editar el fichero /etc/quagga/daemons la línea
bgpd=no
# Por
bgpd=yes
# A continuación restart del servicio
systemctl restart quagga
# Verificar que se bgpd está corriendo
systemctl status quagga
```

```
! UML2 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip addr 10.0.23.1/24
no s[hutdown]
q[uit]
int[erface] eth1
ip addr 10.0.26.1/24
no s[hutdown]
quit
ip f[orwarding]
write
```

```
! UML3 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip address 10.0.23.2/24
no s[hutdown]
q[uit]
int[erface] eth1
ip address 10.0.34.1/24
no s[hutdown]
q[uit]
int[erface] eth2
ip address 10.0.35.1/24
no s[hutdown]
q[uit]
ip f[orwarding]
write
```

```
! UML4 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip address 10.0.34.2/24
no s[hutdown]
q[uit]
int[erface] eth1
ip address 10.0.45.1/24
```

```
no s[hutdown]
q[uit]
int[erface] eth2
ip address 10.0.46.1/24
no s[hutdown]
q[uit]
ip f[orwarding]
write

! UML5 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip address 10.0.35.2/24
no s[hutdown]
q[uit]
int[erface] eth1
ip address 10.0.45.2/24
no s[hutdown]
q[uit]
ip f[orwarding]
write

! UML6 (zebra.conf)
con[figure] t[erminal]
int[erface] eth0
ip address 10.0.26.2/24
int[erface] eth1
ip address 10.0.46.2/24
no s[hutdown]
q[uit]
ip f[orwarding]
write

! UML2 (bgpd.conf)
con[figure] t[erminal]
router bgp 65512
  bgp router-id 10.0.26.1
  ! La red pública que se anuncia en los mensajes BGP que se intercambian
  network 172.16.2.0/24
  ! Establecimiento de los vecinos
  neighbor 10.0.23.2 remote-as 65513
  neighbor 10.0.26.2 remote-as 65516
  ! To enter address family configuration mode for configuring routing sessions,
  ! such as BGP, that use standard IPv6 address prefixes,
  ! use the address-family ipv6 command in router configuration mode
  address-family ipv6
  ! La red IPv4 que se anuncia en los mensajes BGP que se intercambian
```

```
  network 2001:db8:2::/64
  neighbor 10.0.23.2 activate
  neighbor 10.0.26.2 activate
  exit-address-family
  do w[rite]

! UML3 (bgpd.conf)
con[figure] t[erminal]
router bgp 65513
  bgp router-id 10.0.35.1
  network 172.16.3.0/24
  neighbor 10.0.23.1 remote-as 65512
  neighbor 10.0.34.2 remote-as 65514
  neighbor 10.0.35.2 remote-as 65515
  address-family ipv6
  network 2001:db8:3::/64
  neighbor 10.0.23.1 activate
  neighbor 10.0.34.2 activate
  neighbor 10.0.35.2 activate
  exit-address-family
  do w[rite]

! UML4 (bgpd.conf)
con[figure] t[erminal]
router bgp 65514
  bgp router-id 10.0.46.1
  network 172.16.4.0/24
  neighbor 10.0.34.1 remote-as 65513
  ! Añade una politica de route-map de salida
  neighbor 10.0.34.1 route-map backup out
  neighbor 10.0.45.2 remote-as 65515
  ! Añade una politica de route-map de entrada
  neighbor 10.0.45.2 route-map preferente in
  neighbor 10.0.46.2 remote-as 65516
  ! Añade una politica de route-map de salida
  neighbor 10.0.46.2 route-map backup out
!
  address-family ipv6
  network 2001:db8:4::/64
  neighbor 10.0.34.1 activate
  neighbor 10.0.45.2 activate
  neighbor 10.0.46.2 activate
  exit-address-family
!
  route-map preferente permit 10
  ! Establece un local-preference para
```

```
  ! que la ruta tenga mayor preferencia que otra
  set local-preference 200
!
  route-map backup permit 10
  ! Realiza un prepend del identificador
  ! para asegurar que la ruta tiene menor
  ! preferencia que la anterior
  set as-path prepend 65514 65514 65514 65514 65514 65514 65514
!
  do w[rite]

! UML5 (bgpd.conf)
con[figure] t[erminal]
router bgp 65515
  bgp router-id 10.0.35.2
  network 172.16.5.0/24
  neighbor 10.0.35.1 remote-as 65513
  neighbor 10.0.45.1 remote-as 65514
!
  address-family ipv6
  network 2001:db8:5::/64
  neighbor 10.0.35.1 activate
  neighbor 10.0.45.1 activate
  exit-address-family
!
  do w[rite]

! UML6 (bgpd.conf)
con[figure] t[erminal]
router bgp 65516
  bgp router-id 10.0.46.2
  network 172.16.6.0/24
  neighbor 10.0.26.1 remote-as 65512
  neighbor 10.0.46.1 remote-as 65514
!
  address-family ipv6
  network 2001:db8:6::/64
  neighbor 10.0.26.1 activate
  neighbor 10.0.46.1 activate
  exit-address-family
!
  do w[rite]
```

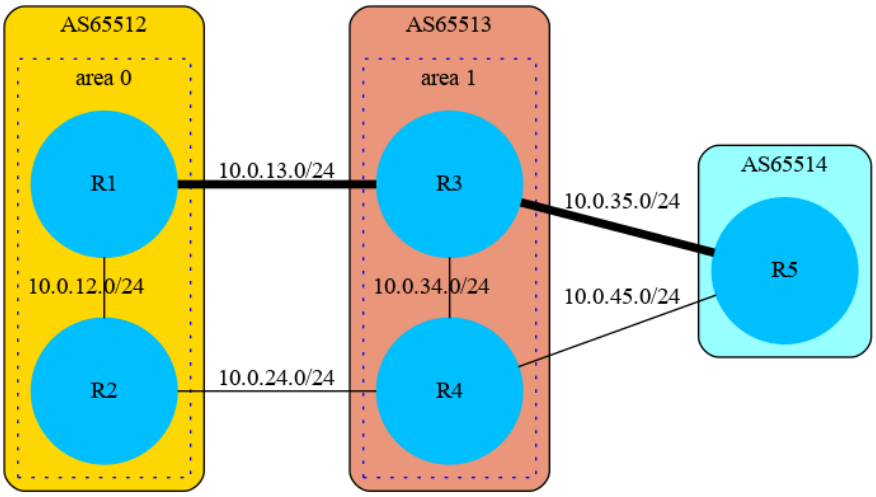
Pruebas:

```
! En una UML
clear bgp * ! se borra toda la información BGP de las tablas de todos los routeres
```

```
! En cada uno de los nodos
! Comprobar que pasar por el AS 65514 tiene un sobrecoste adicional por el prepend
sh[ow] ip bgp
```

7. BGP y OSPFv2

Configurar como principales los enlaces marcados:



Detalles a tener en cuenta:

- Las interfaces entre los nodos R1 - R3 y R2 - R4 han de ser pasivas
- La comunicación entre routeres del mismo AS es mediante iBGP/OSPF
- La comunicación entre los AS es mediante eBGP
- En el AS 65514 la comunicación con el router R5 es exclusivamente BGP
- Anuncios BGP explícitos:
  - AS 65512: 172.12.0.0/16
  - AS 65513: 172.13.0.0/16
  - AS 65514: 172.14.0.0/16

```
# net.conf
defsw br12 uml1.0 uml2.0
defsw br13 uml1.1 uml3.0
defsw br24 uml2.1 uml4.0
defsw br34 uml3.1 uml4.1
defsw br45 uml4.2 uml5.0
defsw br35 uml3.2 uml5.1
```

```
# En cuanto se inician las UML1-4, editar el fichero /etc/quagga/daemons la línea
bgpd=no
ospfd=no
# Por
bgpd=yes
ospfd=yes
# Para UML5, solo cambiar el demonio bgpd=no por bgpd=yes
# A continuación restart del servicio para todas las UML
systemctl restart quagga
# Verificar que se ospfd está corriendo
systemctl status quagga
```

```
! UML1 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.12.1/24
no shutdown
quit
int[erface] eth1
ip address 10.0.13.1/24
no shutdown
quit
ip forwarding
exit
write
```

```
! UML2 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.12.2/24
no shutdown
quit
int[erface] eth1
ip address 10.0.24.1/24
no shutdown
quit
ip forwarding
exit
write
```

```
! UML3 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.13.1/24
no shutdown
quit
int[erface] eth1
```

```
ip address 10.0.34.1/24
no shutdown
int[erface] eth2
ip address 10.0.35.1/24
no shutdown
quit
ip forwarding
exit
write

! UML4 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.24.2/24
no shutdown
quit
int[erface] eth1
ip address 10.0.34.2/24
no shutdown
int[erface] eth2
ip address 10.0.45.1/24
no shutdown
quit
ip forwarding
exit
write

! UML5 (zebra.conf)
conf[igure] term[inal]
int[erface] eth0
ip address 10.0.45.2/24
no shutdown
quit
int[erface] eth1
ip address 10.0.35.2/24
no shutdown
quit
ip forwarding
exit
write

! UML1 (ospf.conf)
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.1
network 10.0.12.0/24 area 0
network 10.0.13.0/24 area 0
```

```
p[assive-interface] eth1 ! El interfaz que comunica con el AS65513
end
write

! UML2 (ospf.conf)
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.2
network 10.0.12.0/24 area 0
network 10.0.24.0/24 area 0
p[assive-interface] eth1 ! El interfaz que comunica con el AS65513
end
write

! UML3 (ospf.conf)
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.3
network 10.0.13.0/24 area 1
network 10.0.34.0/24 area 1
network 10.0.35.0/24 area 1
p[assive-interface] eth0 ! El interfaz que comunica con el AS65512
end
write

! UML4 (ospf.conf)
conf[igure] term[inal]
router ospf
ospf router-id 0.0.0.4
network 10.0.24.0/24 area 1
network 10.0.34.0/24 area 1
network 10.0.35.0/24 area 1
p[assive-interface] eth0 ! El interfaz que comunica con el AS65512
end
write

! UML1 (bgp.conf)
con[figure] t[erminal]
router bgp 65512
nei[ghbor] 10.0.13.2 remote-as 65513
nei[ghbor] 10.0.12.2 remote-as 65512
network 172.12.0.0/16
nei[ghbor] 10.0.13.2 route-m[ap] FILTRO in
route-map FILTRO permit 10
  se[t] l[ocal-preference] 200
do write
```

```
! UML2 (bgp.conf)
con[figure] t[erminal]
router bgp 65512
nei[ghbor] 10.0.24.2 remote-as 65513
nei[ghbor] 10.0.12.1 remote-as 65512
do write

! UML3 (bgp.conf)
con[figure] t[erminal]
router bgp 65513
nei[ghbor] 10.0.13.1 remote-as 65512
nei[ghbor] 10.0.35.2 remote-as 65514
nei[ghbor] 10.0.34.2 remote-as 65513
network 172.13.0.0/16
nei[ghbor] 10.0.35.2 route-m[ap] FILTRO in
route-map FILTRO permit 10
  se[t] l[ocal-preference] 200
do write

! UML4 (bgp.conf)
con[figure] t[erminal]
router bgp 65513
nei[ghbor] 10.0.24.1 remote-as 65512
nei[ghbor] 10.0.45.2 remote-as 65514
nei[ghbor] 10.0.34.1 remote-as 65513
do write

! UML5 (bgp.conf)
con[figure] t[erminal]
router bgp 65514
nei[ghbor] 10.0.35.1 remote-as 65513
nei[ghbor] 10.0.45.1 remote-as 65513
network 172.14.0.0/16
do write
```

Pruebas:

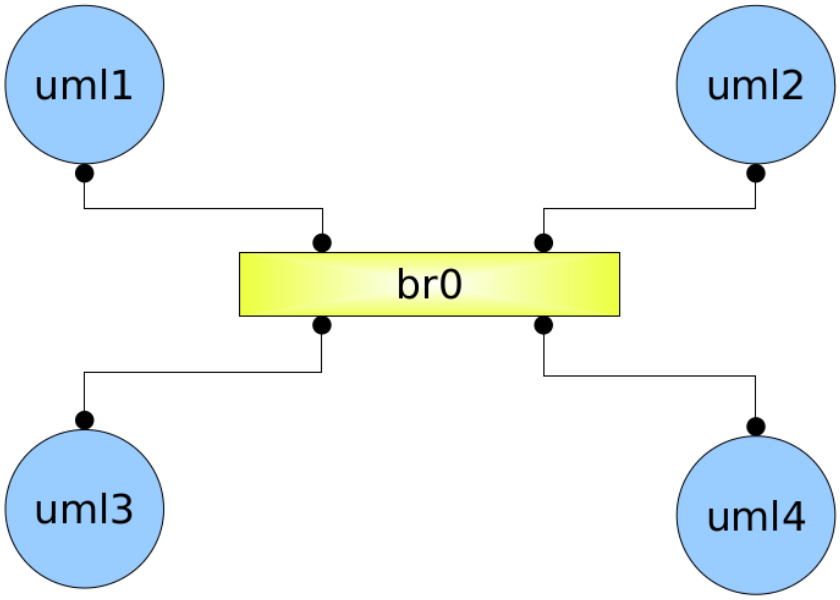
```
! En una UML
clear bgp * ! se borra toda la información BGP de las tablas de todos los routeres
! En cada uno de los nodos
sh[ow] ip bgp
```

- Probar a cambiar el router que anuncia la network para comprobar que las rutas BGP son acordes a las políticas aplicadas

8. OpenFlow: ovs-ofctl

Dada la siguiente topología:

- La máquina UML1 realiza un anuncio de prefijos, y es la única autorizada
- La máquina UML3 también intentará hacerlo, pero OpenFlow lo impedirá



Anuncios de prefijos:

- UML1: 2001:db8:1::/64
- UML3: 2001:db8:3::/64

```
# net.conf
defsw br0 uml1.0 uml2.0 uml3.0 uml4.0
```

```
! UML1
con[figure] t[erminal]
int eth0
no [shutdown]
ipv6 nd prefix 2001:db8:1::/64
no ipv6 nd suppres-ra
q[uit]
ipv6 f[orwarding]
end
w[rite]
```

```
! UML1
con[figure] t[erminal]
int eth0
no [shutdown]
ipv6 nd prefix 2001:db8:3::/64
no ipv6 nd suppres-ra
q[uit]
ipv6 f[orwarding]
end
w[rite]

! UML2 y UML4
con[figure] t[erminal]
int eth0
no [shutdown]
end
w[rite]
```

En este momento, tanto UML1 como UML3 están anunciando sus prefijos. La manera de comprobarlo es ejecutar el comando `ifconfig` en UML2 y UML4, como se muestra a continuación:

```
root@uml2:~# ifconfig
eth0    Link encap:Ethernet  HWaddr 02:00:00:00:02:f0
        inet6 addr: 2001:db8:3::ff:fe00:2f0/64 Scope:Global
        inet6 addr: fe80::ff:fe00:2f0/64 Scope:Link
        inet6 addr: 2001:db8:1::ff:fe00:2f0/64 Scope:Global
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:59 errors:0 dropped:0 overruns:0 frame:0
        TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:6589 (6.4 KiB)  TX bytes:734 (734.0 B)
        Interrupt:5

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:4 errors:0 dropped:0 overruns:0 frame:0
        TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:200 (200.0 B)  TX bytes:200 (200.0 B)
root@uml2:~#
```

En el shell maestro debemos ejecutar las siguientes líneas:

```
redes@RED:~$ sudo su
# Borrar los posibles flujos que puedan existir sobre br0
redes@RED:~# ovs-ofctl del-flows br0
# Añadir un flujo a br0 que dropee los mensajes ICMPv6 del tipo
# ROUTER_ADVERTISEMENT que provengan de UML3
redes@RED:~# ovs-ofctl  add-flow br0 "table=0, priority=200 \
> ipv6_src=fe80::ff:fe00:3f0/128 icmp6 icmp_type=134 actions=drop"
redes@RED:~#
```

Una vez hecho esto, podemos comprobar apagando la int `eth0` de UML2 o UML4 y volviendo a levantar, que ahora los anuncios de UML3 están restringidos, y por tanto, no llegarán anuncios a través del bridge:

```
root@uml2:~# ifconfig
eth0    Link encap:Ethernet  HWaddr 02:00:00:00:02:f0
        inet6 addr: fe80::ff:fe00:2f0/64 Scope:Link
        inet6 addr: 2001:db8:1::ff:fe00:2f0/64 Scope:Global
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:61 errors:0 dropped:0 overruns:0 frame:0
        TX packets:17 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:6781 (6.6 KiB)  TX bytes:1390 (1.3 KiB)
        Interrupt:5

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:4 errors:0 dropped:0 overruns:0 frame:0
        TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:200 (200.0 B)  TX bytes:200 (200.0 B)
root@uml2:~#
```