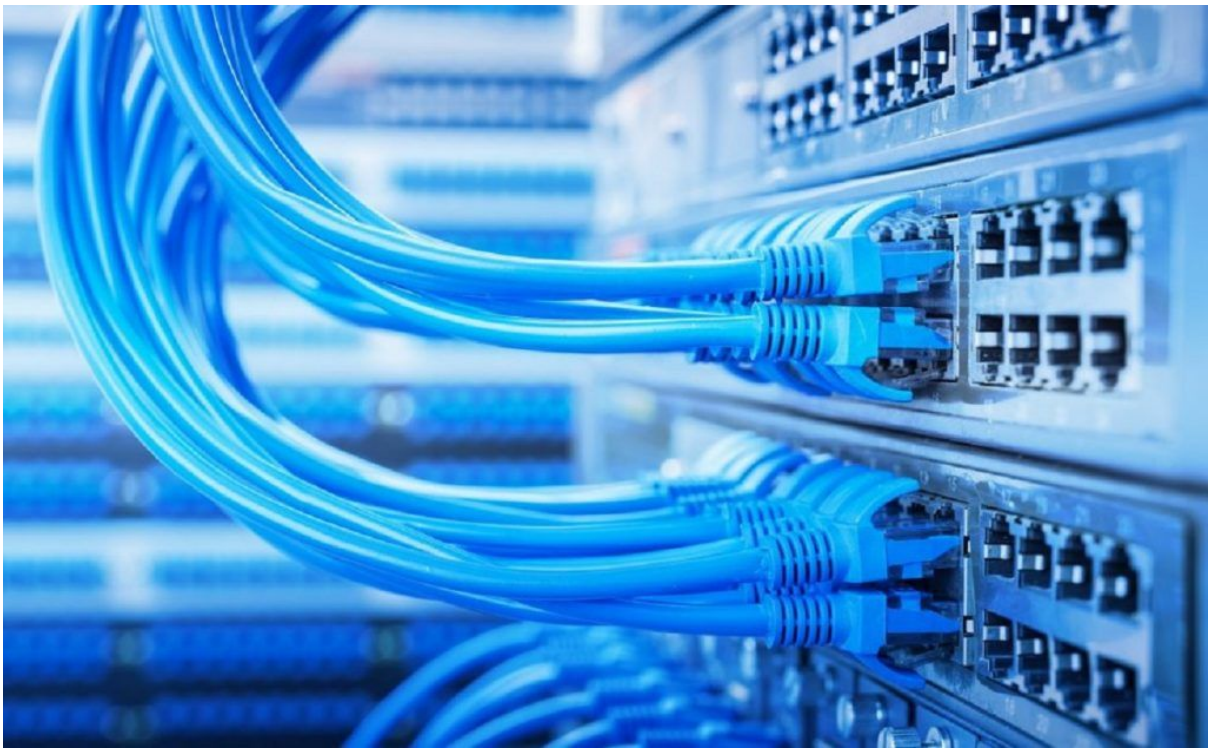
Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Laboratorio de Redes en Ingeniería de Computadores
Ingeniería Informática
Curso 2021-2022
Grupo 02

Informe Práctica 6

Border Gateway protocol (BGP)



Realizado por:
Anabel Díaz Labrador
Cheuk Kelly Ng Pante
Jaime Pablo Pérez Moro
Carmen Clara Rocío Machado

Índice

1. Introducción.

1.1. Sistemas autónomos

Un sistema autónomo se define como “Un grupo de conectado de uno o más prefijos IP promovidos por uno o más operadores de red con una política de enrutamiento **única** y **claramente definida.**

Tipos de sistemas autónomos

- ***Multihomed*:** Son los que tienen conexión con más de un sistema autónomo y no permiten el tránsito del tráfico a través del sistema autónomo.
- ***Stub*:** Es un sistema autónomo que está únicamente conectado con un sistema autónomo.
- ***Tránsito*:** Permite la interconexión de otros sistemas autónomos pasando por dentro del propio sistema autónomo.
- ***Internet Exchange Point (IXP)*:** Es una infraestructura física a través de la cual los proveedores de servicio o de contenido intercambian tráfico entre sus sistemas autónomos.

1.2. Enrutamiento entre sistemas autónomos: BGP (Border Gateway Protocol)

El protocolo de puerta de enlace de frontera o BGP es un protocolo mediante el cual se intercambia información de encaminamiento entre sistemas autónomos. Emplea el protocolo TCP junto con el puerto 179. La conexión entre dos routers BGP se denomina *peers*. Existen varios mensajes que pueden ser intercambiados entre 2 routers *peers*:

- ***OPEN*:** Se emplea en la negociación de los términos de la comunicación después de abrir la comunicación tcp. Ejemplo Versión de BGP.
- ***UPDATE*:** Es un mensaje de actualización con los nuevos prefijos, se generan cada vez que se determina una nueva ruta óptima o una modificación de una existente.
- ***KEEPALIVE*:** Se emplea para comunicar al resto de equipos que el nodo está funcionando correctamente y se emplea. Se envía periódicamente.
- ***NOTIFICATION*:** Se envía cuando cierra una sesión de BGP por un error.

1.3. Atributos de paquete BGP

- ***ORIGIN*:** Identifica el mecanismo por el cual se anunció el prefijo IP por primera vez. Se puede especificar como IGP (0), EGP (1) o INCOMPLETE (2).
- ***AS-PATH*:** Secuencia de números AS que permiten identificar el camino recorrido por un paquete. Cada vez que entra en un sistema autónomo diferente el router bgp de entrada, modifica el atributo para identificar que ha atravesado el sistema autónomo, de esta forma se puede determinar la ruta más corta.
- ***NEXT-HOP*:** Dirección IP del router correspondiente al siguiente salto hacia el destino. Se modifica cuando se anuncia una ruta fuera del sistema autónomo o

cuando se desea redirigir tráfico a otro interlocutor. La información contenida en este campo sirve para incluir los prefijos IP contenidos en el anuncio en la tabla de enrutamiento.

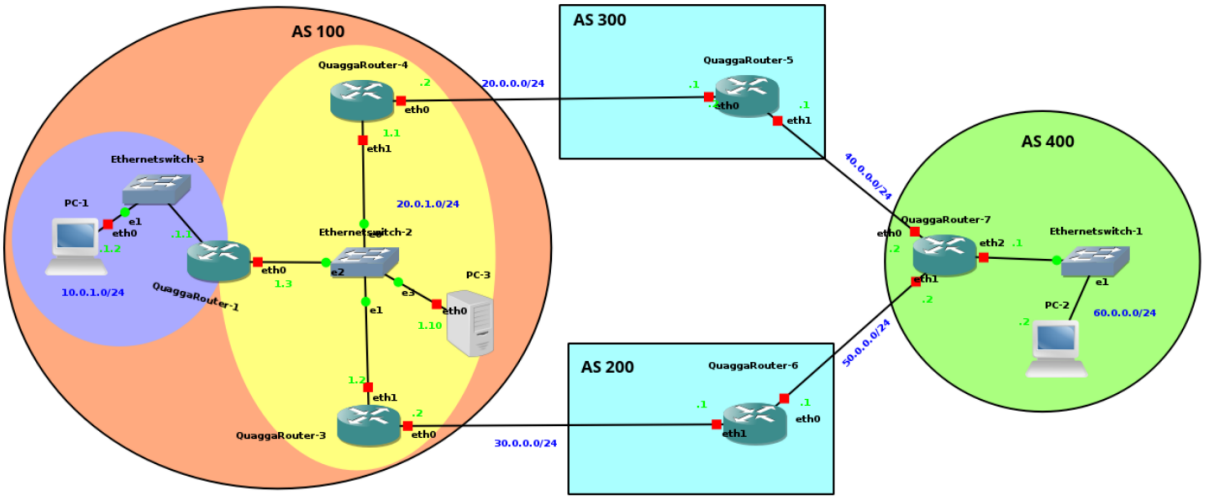
- ***LOCAL-PREF*:** Representa el grado de preferencia que el operador de red tiene por una determinada ruta dentro del sistema autónomo. El valor más alto indica una preferencia mayor. Por defecto, tiene el valor 100. El valor de este atributo es local al AS.
- ***Otros*:** Existen algunos atributos más que no se van a tratar aquí.

1.4. El proceso de enrutamiento: iBGP y eBGP

Existen dos casos de BGP:

- **iBGP:** Se produce cuando se comunican routers del mismo sistema autónomo intercambiando información BGP.
- **eBGP:** Se produce cuando se comunican routers fronterizos de sistemas autónomos diferentes.

2. Topología.



3. BGP.

3.1. Activar las sesiones BGP entre los peers.

Una vez se han asignado los nombres de los routers (R3, R4, R5, R6, R7 que corresponden con QuaggaRouter-3, QuaggaRouter-4 y así sucesivamente) se debe activar una sesión eBGP entre los siguientes peers: R4 - R5, R3 - R6, R5 - R7, R6 - R7.

Para ello se ejecutarán los siguientes comandos siguiendo la Tabla 1.

```
RX(config)#router bgp <AS Number>
RX(config-router)#neighbor <Peer IP address> remote-as <Peer AS Number>
```

Router (RX)	AS Number	Peer IP address	Peer AS Number
R3	100	30.0.0.1	200
		20.0.1.1(iBGP)	100
R4	100	20.0.0.1	300
		20.0.1.2 (iBGP)	100
R5	300	20.0.0.2	100
		40.0.0.2	400
R6	200	30.0.0.2	100
		50.0.0.2	400
R7	400	40.0.0.1	300
		50.0.0.1	200

Tabla 1. Información para la activación de sesiones

A continuación se muestran los comandos ejecutados en cada uno de los routers.

R3:

```
R3(config)# router bgp 100
R3(config-router)# neighbor 30.0.0.1 remote-as 200
R3(config-router)# neighbor 20.0.1.1 remote-as 100
R3(config-router)# exit
```

R4:

```
R4(config)# router bgp 100
R4(config-router)# neighbor 20.0.0.1 remote-as 300
R4(config-router)# neighbor 20.0.1.2 remote-as 100
R4(config-router)# exit
```

R5:

```
R5(config)# router bgp 300
R5(config-router)# neighbor 20.0.0.2 remote-as 100
R5(config-router)# neighbor 40.0.0.2 remote-as 400
R5(config-router)# exit
```

R6:

```
R6(config)# router bgp 200
R6(config-router)# neighbor 30.0.0.2 remote-as 100
R6(config-router)# neighbor 50.0.0.2 remote-as 400
R6(config-router)# exit
```

R7:

```
Debian(config)# router bgp 400
Debian(config-router)# neighbor 40.0.0.1 remote-as 300
Debian(config-router)# neighbor 50.0.0.1 remote-as 200
Debian(config-router)# exit
```

Para este caso concreto se debe indicar que el valor del atributo NEXT-HOP sea la dirección IP del router que propaga el mensaje por iBGP, a˜añadiendo el siguiente a los anteriores en las sesión iBGP (R3-R4) :

```
RX(config-router)#neighbor <Peer IP address> next-hop-self
```

R3:

```
R3(config)# router bgp 100
R3(config-router)# neighbor 20.0.1.1 next-hop-self
R3(config-router)# exit
R3(config)# exit
```

R4:

```
R4(config)# router bgp 100
R4(config-router)# neighbor 20.0.1.2 next-hop-self
R4(config-router)# exit
R4(config)# exit
```

Después de ejecutar estos comandos se comprueba que realmente se han establecido las relaciones de peering entre los routers mediante el comando **show bgp neighbors.** Esto se hace en cada uno de los Routers.

R3:

```
BGP neighbor is 20.0.1.1, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 20.0.1.1
  BGP state = Established, up for 00:18:37
  Last read 00:00:37, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

```
BGP neighbor is 30.0.0.1, remote AS 200, local AS 100, external link
  BGP version 4, remote router ID 50.0.0.1
  BGP state = Established, up for 00:11:23
  Last read 00:00:23, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

R4:

```
BGP neighbor is 20.0.0.1, remote AS 300, local AS 100, external link
  BGP version 4, remote router ID 40.0.0.1
  BGP state = Established, up for 00:08:48
  Last read 00:00:48, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

```
BGP neighbor is 20.0.1.2, remote AS 100, local AS 100, internal link
  BGP version 4, remote router ID 30.0.0.2
  BGP state = Established, up for 00:14:32
  Last read 00:00:32, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

R5:

```
BGP neighbor is 20.0.0.2, remote AS 100, local AS 300, external link
  BGP version 4, remote router ID 20.0.1.1
  BGP state = Established, up for 00:14:18
  Last read 00:00:18, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

```
BGP neighbor is 40.0.0.2, remote AS 400, local AS 300, external link
  BGP version 4, remote router ID 60.0.0.1
  BGP state = Established, up for 00:11:43
  Last read 00:00:43, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

R6:

```
BGP neighbor is 30.0.0.2, remote AS 100, local AS 200, external link
  BGP version 4, remote router ID 30.0.0.2
  BGP state = Established, up for 00:14:19
  Last read 00:00:19, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

```
BGP neighbor is 50.0.0.2, remote AS 400, local AS 200, external link
  BGP version 4, remote router ID 60.0.0.1
  BGP state = Established, up for 00:13:05
  Last read 00:00:05, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

R7:

```
BGP neighbor is 40.0.0.1, remote AS 300, local AS 400, external link
  BGP version 4, remote router ID 40.0.0.1
  BGP state = Established, up for 00:14:04
  Last read 00:00:04, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

```
BGP neighbor is 50.0.0.1, remote AS 200, local AS 400, external link
  BGP version 4, remote router ID 50.0.0.1
  BGP state = Established, up for 00:13:55
  Last read 00:00:55, hold time is 180, keepalive interval is 60 seconds
  Neighbor capabilities:
    4 Byte AS: advertised and received
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    Inq depth is 0
    Outq depth is 0
```

Como podemos ver en las capturas anteriores, las relaciones de vecindad se han establecido de manera correcta. Cabe destacar que todas las relaciones de peering son eBGP (external link) excepto la relación entre el R3 y R4 (iBGP).

3.2. Añadir anuncios de prefijo a los routers.

Con todo lo anterior tenemos ya la red configurada bajo el protocolo BGP pero este todavía no hace nada. Es decir, hay conexión entre los dispositivos pero no son capaces de compartir sus redes. Esto se debe a que no hemos indicado ningún anuncio de prefijo en las configuraciones de los routers.

Pues bien, esto se hace con los siguientes comandos:

```
RX(config)#router bgp 400                    //para el R7
RX(config-router)#network 60.0.0.0/24
```

Luego comprobamos los prefijos conocidos por el router con el comando **show bgp ipv4 unicast:**

R7: Vemos que el R7 conoce el prefijo que hemos añadido

```
Debian# show bgp ipv4 unicast
BGP table version is 0, local router ID is 60.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 60.0.0.0/24      0.0.0.0                  0         32768 i

Total number of prefixes 1
```

Ahora comprobamos los prefijos conocidos por R3 y R4 y vemos que 60.0.0.0/24 se conoce y además es alcanzable por dos caminos,es decir, se tiene una conexión redundante a internet.

R3:

```
Debian# show bgp ipv4 unicast
BGP table version is 0, local router ID is 30.0.0.2
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i60.0.0.0/24      20.0.1.1                 100      0 300 400 i
*>                  30.0.0.1                          0 200 400 i

Total number of prefixes 1
```

R4:

```
Debian# show bgp ipv4 unicast
BGP table version is 0, local router ID is 20.0.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i60.0.0.0/24      20.0.1.2                 100      0 200 400 i
*>                  20.0.0.1                          0 300 400 i

Total number of prefixes 1
```

Por último, podemos revisar las tablas de enrutamiento para comprobar que el prefijo anunciado por AS 400 se encuentra en las tablas y efectivamente se ha anunciado.

R3:

```
Debian#
Debian# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

R>* 10.0.1.0/24 [120/2] via 20.0.1.3, eth1, 01:04:36
R>* 20.0.0.0/24 [120/2] via 20.0.1.1, eth1, 01:04:35
C>* 20.0.1.0/24 is directly connected, eth1
C>* 30.0.0.0/24 is directly connected, eth0
B>* 60.0.0.0/24 [20/0] via 30.0.0.1, eth0, 00:05:50
C>* 127.0.0.0/8 is directly connected, lo
```

R4:

Con todo lo anterior tenemos ya la red configurada bajo el protocolo BGP pero este todavía no hace nada. Es decir, hay conexión entre los dispositivos pero no son capaces de compartir sus redes. Esto se debe a que no hemos indicado ningún anuncio de prefijo en las configuraciones de los routers.

Pues bien, esto se hace con los siguientes comandos:

```
RX(config)#router bgp 400                    //para el R7
RX(config-router)#network 60.0.0.0/24
```

Luego comprobamos los prefijos conocidos por el router con el comando **show bgp ipv4 unicast:**

R7: Vemos que el R7 conoce el prefijo que hemos añadido

```
Debian# show bgp ipv4 unicast
BGP table version is 0, local router ID is 60.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 60.0.0.0/24      0.0.0.0                  0         32768 i

Total number of prefixes 1
```

Ahora comprobamos los prefijos conocidos por R3 y R4 y vemos que 60.0.0.0/24 se conoce y además es alcanzable por dos caminos,es decir, se tiene una conexión redundante a internet.

R3:

```
Debian# show bgp ipv4 unicast
BGP table version is 0, local router ID is 30.0.0.2
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i60.0.0.0/24      20.0.1.1                 100      0 300 400 i
*>                  30.0.0.1                          0 200 400 i

Total number of prefixes 1
```

R4:

```
Debian# show bgp ipv4 unicast
BGP table version is 0, local router ID is 20.0.1.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i60.0.0.0/24      20.0.1.2                 100      0 200 400 i
*>                  20.0.0.1                          0 300 400 i

Total number of prefixes 1
```

Por último, podemos revisar las tablas de enrutamiento para comprobar que el prefijo anunciado por AS 400 se encuentra en las tablas y efectivamente se ha anunciado.

R3:

```
Debian#
Debian# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

R>* 10.0.1.0/24 [120/2] via 20.0.1.3, eth1, 01:04:36
R>* 20.0.0.0/24 [120/2] via 20.0.1.1, eth1, 01:04:35
C>* 20.0.1.0/24 is directly connected, eth1
C>* 30.0.0.0/24 is directly connected, eth0
B>* 60.0.0.0/24 [20/0] via 30.0.0.1, eth0, 00:05:50
C>* 127.0.0.0/8 is directly connected, lo
```

R4:

```
R4# show ip r
rip    route
R4# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

R>* 10.0.1.0/24 [120/2] via 20.0.1.3, eth1, 00:40:24
C>* 20.0.0.0/24 is directly connected, eth0
C>* 20.0.1.0/24 is directly connected, eth1
B>* 60.0.0.0/24 [20/0] via 20.0.0.1, eth0, 00:03:45
C>* 127.0.0.0/8 is directly connected, lo
R4#
```

3.3. Propagar los prefijos internos del AS 100.

Como en el anterior paso se han añadido los prefijos de red del AS 400, ahora también es necesario propagar el prefijo de red del AS 100, porque el AS 200 y el AS 300 no tiene equipos ni más prefijos que compartir. El problema es que no nos conviene que el exterior conozca el prefijo correspondiente a la red interna de este AS, por lo que solo tenemos que publicar su parte pública, la red 20.0.0.0/24 y 20.0.1.0/24.

Una manera de conseguir esto es mediante su ruta sumarizada 20.0.0.0/16. Quedando la 10.0.1.0/24 invisible al exterior.

Para propagar estos prefijos necesitamos repetir los pasos del apartado anterior en ambos routers frontera, siendo lo siguiente.

R3:

```
QuaggaRouter-3# conf ter
QuaggaRouter-3(config)# router bgp 100
QuaggaRouter-3(config-router)# network 20.0.0.0/16
QuaggaRouter-3(config-router)# exit
QuaggaRouter-3(config)# exit
```

R4:

```
QuaggaRouter-4# configure terminal
QuaggaRouter-4(config)# router bgp 100
QuaggaRouter-4(config-router)# network 20.0.0.0/16
QuaggaRouter-4(config-router)# exit
```

Comprobamos que se ha propagado mirándolo en R6.

```
show bgp ipv4 unicast
BGP table version is 0, local router ID is 50.0.0.1
Status codes: s suppressed, d damped, h history, * valid, > best, = multipath,
              i internal, r RIB-failure, S Stale, R Removed
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  20.0.0.0/16      50.0.0.2                               0 400 300 100 i
*>                  30.0.0.2                 0             0 100 i
*> 60.0.0.0/24      50.0.0.2                 0             0 400 i

Total number of prefixes 2
QuaggaRouter-6#
```

Y la conectividad entre el PC-3 y el PC-2 da el siguiente resultado.

```
PC-3 — Konsole
File   Edit   View   Bookmarks   Plugins   Settings   Help
root@Debian:~# traceroute 60.0.0.2
traceroute to 60.0.0.2 (60.0.0.2), 30 hops max, 60 byte packets
 1  20.0.1.1 (20.0.1.1)  0.757 ms  0.641 ms  0.648 ms
 2  20.0.0.1 (20.0.0.1)  0.872 ms  0.872 ms  0.869 ms
 3  40.0.0.2 (40.0.0.2)  1.126 ms  1.125 ms  1.122 ms
 4  60.0.0.2 (60.0.0.2)  1.721 ms  1.722 ms  1.722 ms
```

Como puede comprobar es un resultado diferente al del guión de la práctica y es que el de este último carece de sentido ya que en el camino que traza, iría pasando por el R4 y luego saltaría a la red que está entre R6 y R7, no siendo alcanzable directamente desde R4.

Imagen del guión de este apartado.

```
traceroute to 60.0.0.2 (60.0.0.2), 30 hops max, 60 byte packets
1  20.0.1.1 (20.0.1.1)  0.720 ms  0.579 ms  0.539 ms
2  20.0.0.1 (20.0.0.1)  1.203 ms  2.187 ms  2.302 ms
3  50.0.0.2 (50.0.0.2)  1.977 ms  1.852 ms  1.793 ms
4  60.0.0.2 (60.0.0.2)  2.980 ms  3.273 ms  3.174 ms
```

3.4. Propagar una ruta por defecto hacia los routers frontera.

Para evitar la propagación de la tabla de enrutamiento de Internet hacia los routers internos del AS 100 (en este caso existe como router interno el R1), es necesario establecer a uno de los dos routers fronterizos como salida por defecto dentro del protocolo de enrutamiento de pasarela interior RIP.

Se ha elegido al R4 para la configuración.

```
router rip
default-information originate
```

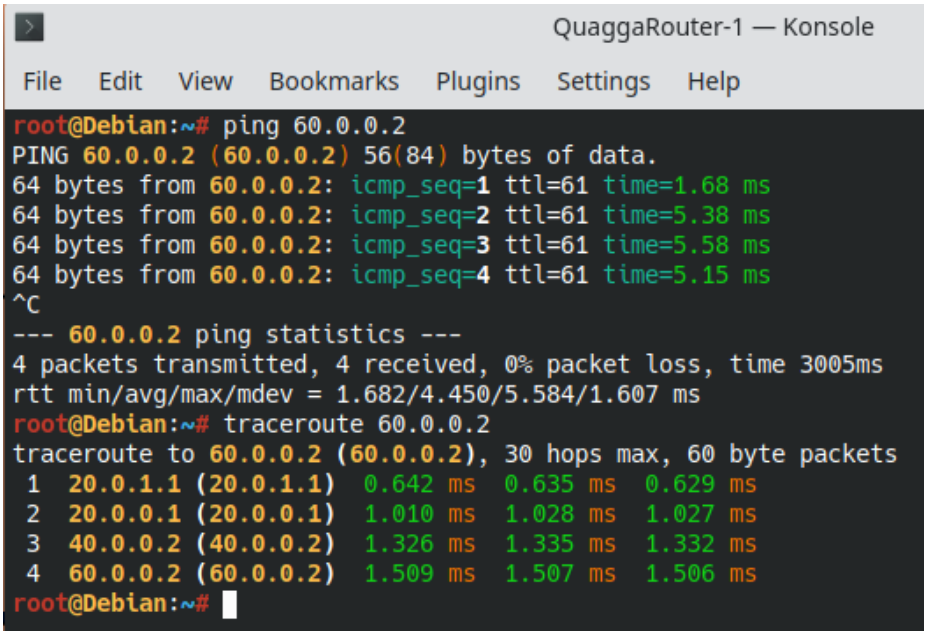
```
QuaggaRouter-4(config)# router rip
QuaggaRouter-4(config-router)# default-information originate
QuaggaRouter-4(config-router)# exit
```

Ahora comprobamos en el router interno si aparece una ruta por defecto hacia el R4.

```
QuaggaRouter-1# show ip route
Codes: K - kernel route, C - connected, S - static, R - RIP,
       O - OSPF, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route

R>* 0.0.0.0/0 [120/2] via 20.0.1.1, eth0, 00:03:23
C>* 10.0.1.0/24 is directly connected, eth1
R>* 20.0.0.0/24 [120/2] via 20.0.1.1, eth0, 00:26:52
C>* 20.0.1.0/24 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
```

Y hacemos ping desde R1 al PC-2 para comprobar que existe conexión al exterior. Pero sin embargo desde el PC-1 no es posible porque no hemos configurado ninguna NAT.



3.4.1. Comprobar funcionamiento de la salida redundante a Internet.

Para comprobar el funcionamiento vamos a seguir los siguientes pasos:

Primero es necesario saber por dónde van los paquetes, por lo que hacemos un traceroute desde R1 para comprobarlo.

```
QuaggaRouter-1# traceroute 60.0.0.2
traceroute to 60.0.0.2 (60.0.0.2), 30 hops max, 60 byte packets
 1  20.0.1.1 (20.0.1.1)  1.010 ms  0.975 ms  0.955 ms
 2  20.0.0.1 (20.0.0.1)  1.458 ms  1.446 ms  1.479 ms
 3  40.0.0.2 (40.0.0.2)  2.122 ms  2.090 ms  2.088 ms
 4  60.0.0.2 (60.0.0.2)  2.407 ms  2.387 ms  2.386 ms
```

En esta captura se puede ver que hace el siguiente recorrido: R1-R4-R5-R7-(PC-2).
Ahora debemos inhabilitar una parte del camino, por ejemplo eth0 de R5 y volvemos a hacer un traceroute.

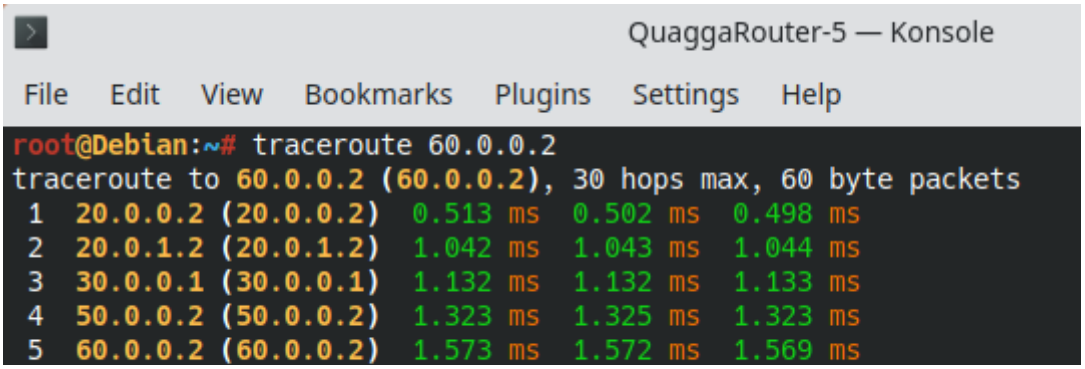
```
QuaggaRouter-1# traceroute 60.0.0.2
traceroute to 60.0.0.2 (60.0.0.2), 30 hops max, 60 byte packets
 1  20.0.1.1 (20.0.1.1)  0.728 ms  0.698 ms  0.697 ms
 2  20.0.1.2 (20.0.1.2)  1.599 ms  1.601 ms  1.599 ms
 3  30.0.0.1 (30.0.0.1)  1.597 ms  1.593 ms  1.590 ms
 4  50.0.0.2 (50.0.0.2)  1.588 ms  1.585 ms  1.556 ms
 5  60.0.0.2 (60.0.0.2)  1.831 ms  1.842 ms  1.899 ms
```

Ahora podemos ver que coge el camino de abajo, haciendo lo siguiente: R1-R4-R3-R6-R7-(PC-2).

Hay un salto más porque siempre tiene que pasar por R4 porque es la que pusimos como salida por defecto en el apartado anterior. Se podría solucionar con VRRP englobando a los dos routers R3 y R4 en un solo router virtual pero esto no se abordará en esta práctica.

3.5. Políticas de enrutamiento.

El propósito de tener dos conexiones a otros sistemas autónomos en AS 100 era tener salida redundante a Internet por si uno de los ISP falla. Esto lo hemos resuelto, como hemos podido comprobar en el paso anterior. Sin embargo, existe otro problema que debemos solucionar. Aún con la interfaz eth1 del QuaggaRouter-5 apagada ejecutamos un traceroute hacia 60.0.0.2.



Se puede observar que el camino que toma va a través del AS100 para llegar hasta el destino y por lo tanto el sistema autónomo es un sistema autónomo de tránsito, cosa que no queremos ya que esto implica costes en ancho de banda, que no queremos pagar.

Para solucionar esto tenemos que establecer una política de enrutamiento:

Tráfico de salida:

- Debe preferir la ruta por defecto a través de QuaggaRouter-4.
- Si el enlace con el AS300 falla, entonces debemos recurrir a la segunda salida a través de QuaggaRouter-3.

Tráfico de entrada:

- Si uno de los dos ISP falla deberá devolver el tráfico a través del otro.

Para establecer estas políticas se hace uso de los mapas de rutas. Primero nos centramos en el tráfico de salida. Creamos un mapa de rutas para modificar el atributo de preferencia local de las rutas que vienen a través de QuaggaRouter-4.

```
(config)# route-map AS300-entrada permit 10
(config-route-map)# set local-preference 200
(exit)#
```

Gracias a esto todas las rutas filtradas por el mapa de rutas se les fija el atributo de preferencia local (dentro del sistema autónomo) a 200. El valor por defecto de la preferencia local es 100, por lo que las rutas que se aprendan desde QuaggaRouter-3 tienen menos prioridad que las aprendidas a través del QuaggaRouter-4.

Ahora se aplica el mapa de rutas a los mensajes BGP entrantes desde QuaggaRouter-5.

```
(config)# router bgp 100
(config-router)# neighbor 20.0.0.1 route-map AS300-entrada in
```

Activamos de nuevo el eth1 del QuaggaRouter-5 y en el QuaggaRouter-3 vamos a comprobar el prefijo 60.0.0.0/24.

```
Debian# sh ip bgp 60.0.0.0/24
BGP routing table entry for 60.0.0.0/24
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
  30.0.0.1
  300 400
    20.0.1.1 from 20.0.1.1 (20.0.1.1)
      Origin IGP, localpref 200, valid, internal, best
      Last update: Wed Apr  6 17:20:54 2022

  200 400
    30.0.0.1 from 30.0.0.1 (50.0.0.1)
      Origin IGP, localpref 100, valid, external
      Last update: Wed Apr  6 16:19:51 2022
```

Se puede observar como hay dos rutas alternativas pero hay una en la que localpref tiene valor 200 y es más prioritaria.

Pasamos a ocuparnos del tráfico de salida, para ello filtramos los prefijos que se propagan desde AS100 hacia AS200 y AS 300 estableciendo un mapa de rutas para las rutas salientes ejecutando en el router 4 los siguientes comandos:

```
(config)# ip as-path access-list 1 permit ^$
(config)# route-map AS300-salida permit 10
(config-route-map)# match as-path 1
```

Aplicamos el mapa de rutas recién creado al tráfico de salida:

```
(config-router)# router bgp 100
(config-router)# neighbor 20.0.0.1 route-map AS300-salida out
```

Estas políticas de filtrado no se aplican inmediatamente por lo que debemos forzar su aplicación enviando mensajes de UPDATE a los vecinos mediante el comando:

```
# clear ip bgp * soft
```

Ahora observamos el efecto en el QuaggaRouter-5 comprobando el prefijo 60.0.0.0/24 que solo es alcanzable por un único camino.

```
Debian# sh ip bgp 60.0.0.0/24
BGP routing table entry for 60.0.0.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
  20.0.0.2
  400
    40.0.0.2 from 40.0.0.2 (60.0.0.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Last update: Wed Apr  6 17:20:24 2022
```

Si comprobamos el prefijo 20.0.0.0/16 sí que se consideran los dos caminos alternativos:

```
Debian# sh ip bgp 20.0.0.0/16
BGP routing table entry for 20.0.0.0/16
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
  40.0.0.2
  400 200 100
    40.0.0.2 from 40.0.0.2 (60.0.0.1)
      Origin IGP, localpref 100, valid, external
      Last update: Wed Apr  6 17:20:24 2022

  100
    20.0.0.2 from 20.0.0.2 (20.0.1.1)
      Origin IGP, metric 0, localpref 100, valid, external, best
      Last update: Wed Apr  6 17:35:17 2022
```

Aplicamos el mismo principio al router 3:

```
(config)# ip as-path access-list 1 permit ^$
(config)# route-map AS400-salida permit 10
(config-route-map)# match as-path 1
(config-route-map)# exit
(config)# router bgp 100
(config-router)# neighbor 30.0.0.1 route-map AS400-salida out
```

Para terminar vamos a comprobar qué pasa si paramos nuevamente la interfaz eth1 del router 5 y hacemos ping desde éste hacia 60.0.0.2. Observamos que no se puede hacer ping.

```
QuaggaRouter-5# ping 60.0.0.2
connect: Network is unreachable
```

Comprobamos la conectividad desde el router 1 hacia 60.0.0.2.

```
R1# ping 60.0.0.2
PING 60.0.0.2 (60.0.0.2) 56(84) bytes of data.
From 20.0.1.1: icmp_seq=1 Redirect Host(New nexthop: 20.0.1.2)
64 bytes from 60.0.0.2: icmp_seq=1 ttl=61 time=1.75 ms
From 20.0.1.1: icmp_seq=2 Redirect Host(New nexthop: 20.0.1.2)
64 bytes from 60.0.0.2: icmp_seq=2 ttl=61 time=1.95 ms
From 20.0.1.1: icmp_seq=3 Redirect Host(New nexthop: 20.0.1.2)
64 bytes from 60.0.0.2: icmp_seq=3 ttl=61 time=1.81 ms
^C
--- 60.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 1.756/1.840/1.956/0.097 ms
```

4. Referencias.

Apuntes de clase.

Guión Práctica 6 .- Border Gateway protocol (BGP)