

ARQUITECTURAS DE REDES AVANZADAS

QUAGGA

José Luis Cánovas Sánchez

20 de noviembre de 2015



Open Source Routing

Resumen

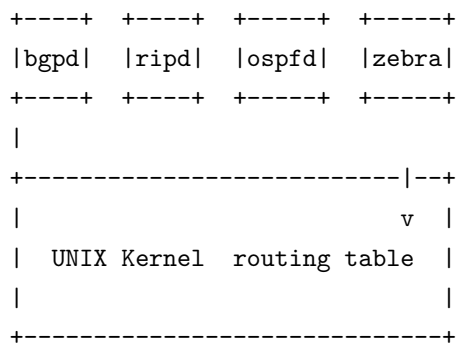
En este informe se redacta el despliegue de un escenario de red IPv6 usando la herramienta Quagga.

Índice

1. Introducción	2
2. Topología de trabajo	2
3. Instalar Quagga	4
4. Configuración de OSPF	4
5. Configuración de RIPng	5
6. Configuración de BGP	6

1. Introducción

Quagga es un proyecto de software que provee de utilidades para encaminamiento de red en sistemas UNIX, proveyendo el demonio zebra originario de otro proyecto y que permite aplicar configuraciones de algoritmos de encaminamiento a las tablas de red de la máquina. Los algoritmos se ejecutan como demonios que toman la información de la red y aplican protocolos conocidos, tales como OSPF, RIP, IS-IS y BGP, en múltiples de sus versiones, y por medio de zebra los aplican a la red.



Quagga System Architecture

Vamos a configurar tres de estos protocolos con Quagga en cuatro máquinas que harán las veces de routers. En la sección 2 se muestra la topología que se va a implementar. En las secciones 3 a 6 se explicará cómo instalar y configurar Quagga con los mecanismos OSPF, RIP y BGP para IPv6.

La topología usada en este estudio de Quagga se asemeja mucho a la de la práctica 1 de ARA, pero se puede llevar a otros proyectos, como LEGO, fácilmente. Más adelante se muestra la topología en Figura 1, y con hacer corresponder R1 y R2 con los routers de las organizaciones de LEGO, las redes XXXX:0:1::/64 con las subredes de cada organización, y las subredes XXXX::/64 con una subred un nivel más interna, tenemos la topología de LEGO ampliada. La salida a internet se debería realizar con tunelado a IPv4 en R1 y R2, si no se dispone de IPv6 pública. En caso de tener un rango de direcciones IPv6 públicas, habría que cambiar los valores de cada subred para que sean válidos, pero queda fuera de este estudio.

2. Topología de trabajo

Partimos de dos Sistemas Autónomos con números 17 y 71 (curiosidad, son primos permutables entre sí) y mecanismo de encaminamiento interno por OSPF, con un área backbone y un área 1 stub para el AS17, un mecanismo basado en RIPng para el AS1 y BGP entre ambos AS, como se muestra en la Figura 1.

Esta topología lógica la llevamos a la práctica con Virtual Box, con cuatro máquinas Debian sin interfaz gráfica, y con tarjetas de red virtuales configuradas como redes internas del siguiente modo:

- R1: Adaptador 1 Red Interna *AS17* - Adaptador 2 Red Interna *interAS* - Adaptador 3 NAT.
- R2: Adaptador 1 Red Interna *AS71* - Adaptador 2 Red Interna *interAS* - Adaptador 3 NAT.

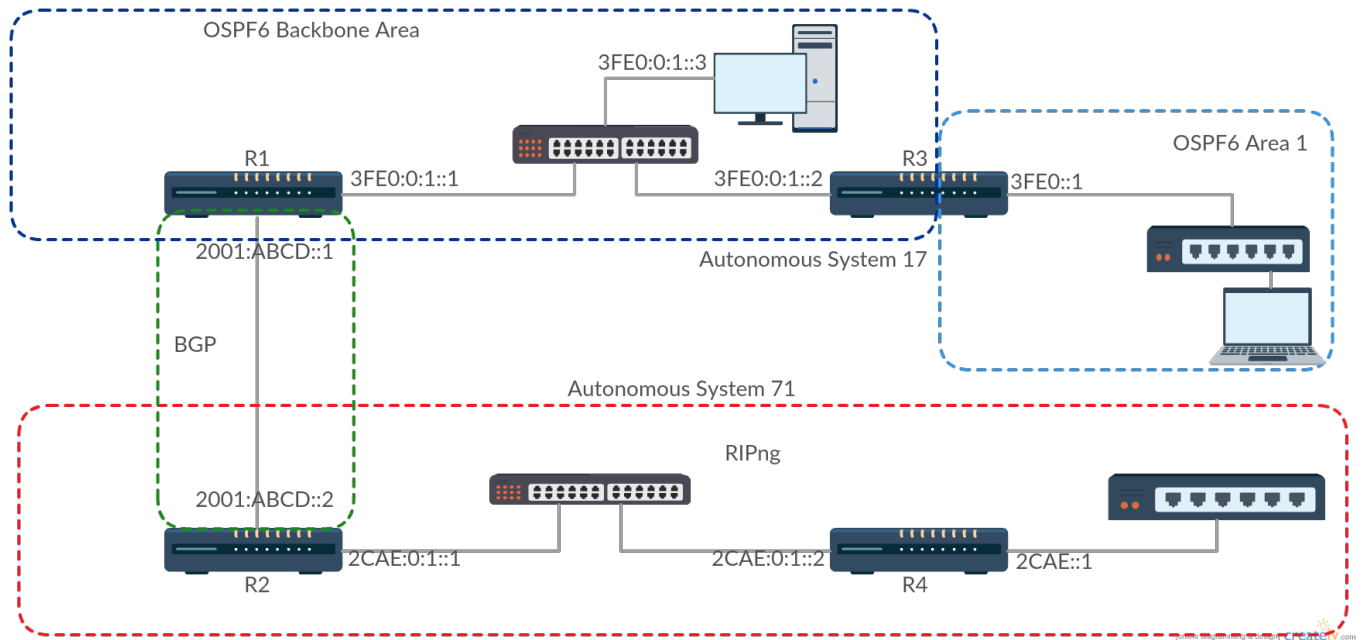


Figura 1: Topología

- R3: Adaptador 1 Red Interna *AS17* - Adaptador 2 Red Interna *AS17Area1* - Adaptador 3 NAT.
- R4: Adaptador 1 Red Interna *AS71* - Adaptador 2 Red Interna *AS71Subred* - Adaptador 3 NAT.

Con el NAT conseguimos salida a internet para poder descargar los ficheros de configuración. Con las otras conseguimos simular los switches y conectar cada router por las interfaces correspondientes.

La configuración de `/etc/network/interfaces` sería, por ejemplo para R1:

```
auto lo
```

```
iface lo inet loopback
```

```
auto eth0
```

```
iface eth0 inet6 static
```

```
address 3FE0:0:1::1
```

```
netmask 64
```

```
auto eth1
```

```
iface eth1 inet6 static
```

```
address 3FE0::1
```

```
netmask 64
```

```
auto eth2
```

```
iface eth2 inet dhcp
```

Desde Quagga también se podrían configurar usando las órdenes:

```
interface eth0
no shutdown
ipv6 address 3FE0:0:1::1/64
```

Y sustituyendo eth0 y la dirección ipv6 por cada interfaz correspondiente y su dirección.

Como trabajamos con máquinas virtuales podemos trabajar con */etc/network/interfaces* para una topología inicial, y en caso de que ya no nos valga, modificar la configuración mediante Quagga para ir adaptándola, en vez de editar */etc/network/interfaces*.

3. Instalar Quagga

Para facilitar las pruebas, hay preparado un fichero Makefile que recibe como parámetro el router a configurar (r1, r2, r3 o r4), y ejecuta un script *install.sh* que instala por aptitude el paquete *quagga*. Con esto no es suficiente para que quagga funcione, ni tampoco para poder iniciar su consola de configuración. Para ello se crean en */etc/quagga/* los ficheros *zebra.conf*, *bgpd.conf*, *ospf6d.conf*, *ripngd.conf* y *vttysh.conf* con usuario quagga y grupo quaggavty.

A todos los ficheros menos a *vttysh.conf* se les añade la línea *password pwd* donde pwd será la contraseña para poder configurar el demonio por su propia terminal. El fichero *vttysh.conf* configura la terminal vtysh, que permite controlar todos los demonios, al estilo de la terminal de un router Cisco, y para mayor comodidad, se escribe *username root nopassword* para no necesitar iniciar sesión cada vez.

Con esto los demonios podrían iniciarse sin problemas, pero falta decirle a Quagga qué demonios iniciar. Para ello el fichero */etc/quagga/daemons* se reescribe con:

```
zebra=yes
bgpd=yes
ospfd=no
ospf6d=yes
ripd=no
ripngd=no
isisd=no
babeld=no
```

Cambiando por yes o no según queramos activar o no cada demonio. El de zebra es obligatorio si queremos que se apliquen las tablas de rutas de los demonios al sistema real, pero si sólo se quiere ejecutar el protocolo sin afectar a la máquina, para pruebas por ejemplo, se puede desactivar.

Finalmente habría que reiniciar Quagga con */etc/init.d/quagga restart*, lo cual hace el Makefile automáticamente y facilita al añadir una regla extra de r1-r4 llamada restart.

4. Configuración de OSPF

La configuración por comandos de OSPFv3 partiendo del punto de entrada de *sudo vtysh* sería:

```
1 configure terminal
2 router ospf6
3 router-id a.b.c.d
4 interface ethx area e.f.g.h
5 redistribute connected
6 redistribute prot
7 exit
8 exit
9 write
```

Donde a.b.c.d se sustituye por un ID para el router en forma de dirección ipv4. Para R1 elegimos 1.1.1.1 y para R3 2.2.2.2.

Las áreas se identifican también con direcciones tipo ipv4 sustituyendo el valor en e.f.g.h, y nótese que no es necesario ir a la configuración de la interfaz y activar ahí ospf, quagga lo hace automáticamente desde la configuración de ospf al indicar la interfaz, aquí sustituyendo ethx por la interfaz correspondiente. Para las áreas la 0.0.0.0 será la backbone, y la 0.0.0.1 la segunda área de R3.

Para anunciar las redes que no están incluidas en las áreas de OSPF, se debe indicar con redistribute, en el caso de R1 connected y bgp.

Para R3 la línea de *interface* habrá que ejecutarla una vez por cada interfaz y su área, y no haría falta el redistribute pues todas sus redes están en el AS gestionado con OSPF.

Los ficheros de configuración generados al final por Quagga son casi idénticos a los de Cisco, y podrían ser la lista de comandos a ejecutar, excepto que añaden la información de las interfaces que deben usar ospf.

5. Configuración de RIPng

La configuración por comandos de RIPng partiendo del punto de entrada de *sudo vtysh* sería:

```
1 configure terminal
2 router ripng
3 network ethx
4 redistribute connected
5 redistribute prot
6 exit
7 exit
8 write
```

La configuración de RIPng es aún más sencilla que OSPF, donde además Quagga nos facilita el definir las redes a distribuir indicando directamente la interfaz donde antes pone network ethx.

Para que R2 anuncie las rutas a redes que quedan fuera del AS administrado por RIPng, al igual que en OSPF, se debe añadir la orden redistribute, anunciando las redes directamente conectadas y las aprendidas por bgp.

En R4 no hacen falta las líneas *redistribute* y habría que poner la orden *network ethx* por cada una de las dos interfaces que tiene y quedan dentro del AS.

Los ficheros de configuración del demonio *ripngd* generados son también parecidos a los de Cisco y en este caso son los comandos ejecutados antes:

6. Configuración de BGP