ARQUITECTURAS DE REDES AVANZADAS
QUAGGA

José Luis Cánovas Sánchez

23 de noviembre de 2015

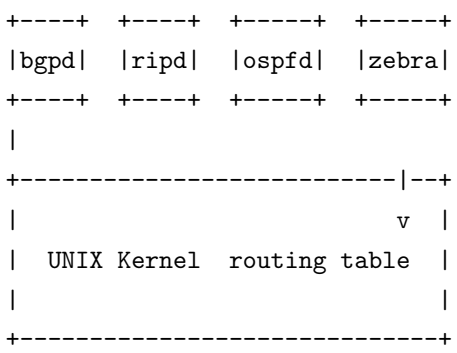
Resumen

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

Índice

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 OSPFv3 y RIPng 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 gŕafica, y con tarjetas de red virtuales configuradas como redes internas del siguiente modo:

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

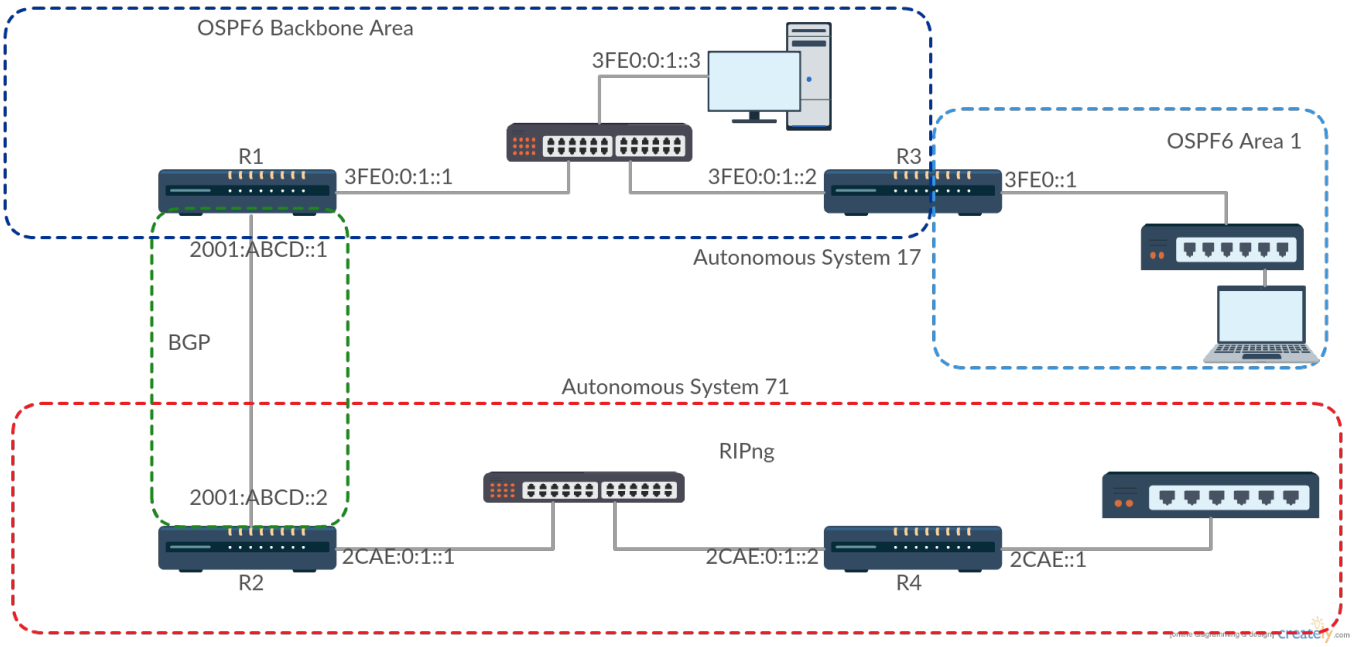


Figura 1: Topología

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

Con las otras conseguimos simular los switchs y conectar cada router por las interfaces correspondientes.

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*, *ospf6d.conf*, *ripngd.conf* y *vtysh.conf* con usuario quagga y grupo quaggavty.

A todos los ficheros menos a *vtysh.conf* se les añade la línea *password pwd* donde pwd será la contraseña para poder configurar el demonio por su propia terminal. El fichero *vtysh.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=no
ospfd=no
```

```
ospf6d=yes
ripd=no
ripngd=yes
isisd=no
babeld=no
```

Cambiando *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.

En cuanto a que la máquina se quiera usar como router, se debe activar el forwarding de paquetes ipv6 a nivel de sistema operativo. En nuestro caso usamos la orden *sysctl -w net.ipv6.conf.all.forwarding=1*. Para facilitar su ejecución en cada reinicio, existe una regla en el Makefile llamada *ipv6forward* que ejecuta el comando. Si se quiere evitar ejecutarlo cada vez que se inicia la máquina, basta modificar la entrada correspondiente de */etc/sysctl*

Finalmente habría que reiniciar Quagga con */etc/init.d/quagga restart*, y existe en Makefile una regla extra llamada *restart* para facilitarlo.

4. Configuración de interfaces

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
```

Pero nosotros vamos a configurarlo desde Quagga usando las órdenes:

```
configure terminal
        interface eth0
                no shutdown
                ipv6 address 3FE0:0:1::1/64
                no ipv6 nd suppress-ra
                ipv6 nd prefix 3fe0:0:1::/64 300 250 router-address
```

Y sustituyendo eth0 y la dirección ipv6 por cada interfaz correspondiente y su dirección. Las líneas que incluyen *nd* (neighbour discovery) se incluyen para enviar Router Advertisements (no suppress-ra) y definir la información del Prefix Information Option del router advertisement (nd prefix). Con esto la interfaz queda configurada y actúa como router.

En los ficheros que usa el Makefile de instalación se incluyen las interfaces ya configuradas en zebra.conf (fichero que configura lo relativo a interfaces y la red en general, independiente del protocolo de enrutamiento).

En ese mismo fichero también se encuentran las rutas estáticas que deben conocer R1 y R2 para conocer la red del otro AS. Una ruta estática se indica en el estado de *configure terminal* con *ipv6 route X:X::X:X/X Y:Y::Y:Y*, donde la primera dirección indica una subred con prefijo, y la segunda el gateway. Hay que tener cuidado con la ayuda de la terminal, pues también indica que se puede poner una interfaz como gateway, pero quagga tomará esa información como que la interfaz está conectada a dicha subred directamente.

El fichero de configuración *zebra.conf* de R1, como ejemplo, es:

```
!
! Zebra configuration saved from vty
!   2015/11/22 19:55:56
!
password manyhue
!
interface eth0
ipv6 address 3fe0:0:1::1/64
no ipv6 nd suppress-ra
ipv6 nd prefix 3fe0:0:1::/64 300 250 router-address
!
interface eth1
ipv6 address 2001:ABCD::1/64
no ipv6 nd suppress-ra
ipv6 nd prefix 2001:ABCD::/64 300 250 router-address
!
interface lo
!
ipv6 route 2cae::/42 2001:abcd::2
!
!
!
line vty
!
```

5. Configuración de OSPF

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

```
configure terminal
        router ospf6
                router-id a.b.c.d
                interface ethx area e.f.g.h
                redistribute connected
                redistribute static
                end
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 static.

Para R3 la línea de *interface* habrá que ejecutarla una vez por cada una de las dos interfaces y sus áreas, 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.

El fichero de configuración *ospf6d.conf* de R3, como ejemplo, es:

```
!
! Zebra configuration saved from vty
!   2015/11/17 18:45:32
!
password manyhue
!
debug ospf6 lsa unknown
!
interface eth0
ipv6 ospf6 network broadcast
!
interface eth1
ipv6 ospf6 network broadcast
!
router ospf6
router-id 2.2.2.2
interface eth0 area 0.0.0.0
interface eth1 area 0.0.0.1
!
line vty
!
```

6. Configuración de RIPng

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

```
configure terminal
        router ripng
                network ethx
                redistribute connected
                redistribute static
                exit
        exit
write
```

La configuración de RIPng es aún más sencilla que OSPF, donde además Quagga nos facilita el definir las redes a distribuir inticando 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 indicadas por rutas estáticas.

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:

El fichero de configuración *ripngd.conf* de R2, como ejemplo, es:

```
1  !
2  ! Zebra configuration saved from vty
3  !   2015/11/17 20:15:56
4  !
5  password manyhue
6  !
7  router ripng
8  network eth0
9  redistribute connected
10 redistribute static
11 !
12 line vty
13 !
```

Referencias

[1] Documentación HTML de Quagga. Link

[2] Tips & tricks de Quagga. Link