***Trabajo Práctico*** *4*

*Ruteo dinámico*

*Autores*

*Esteban Nicolas Andrada:*

[*nico86gnr@gmail.com*](mailto:nico86gnr@gmail.com)

*Federico Sepulveda:* [*federico.sepulveda@alumnos.unc.edu.ar*](mailto:federico.sepulveda@alumnos.unc.edu.ar)

# Diagrama

# 

# Tabla de asignación de direcciones IPv6

# Crear la tabla de asignación de direcciones IP

# 

|  |  |  |
| --- | --- | --- |
| **Computadora** | **Interfaz de Red** | **Dirección IP** |
| r1 | eth0 | 2001:1111:2222::10 |
| eth1 | 2001:1111:3333::10 |
| eth2 | 2001:1111:5555::10 |
| eth3 | 2001:1111:7777::10 |
| r2 | eth0 | 2001:1111:2222::20 |
| eth1 | 2001:2222:4444::20 |
| eth2 | 2001:2222:6666::20 |
| eth3 | 2001:2222:8888::20 |
| r3 | eth0 | 2001:1111:3333::30 |
| eth1 | 2001:3333:3333::31 |
| eth2 | 2001:3333:3333::32 |
| r4 | eth0 | 2001:2222:4444::40 |
| eth1 | 2001:4444:4444::41 |
| eth2 | 2001:4444:4444::42 |
| r5 | eth0 | 2001:1111:5555::50 |
| eth1 | 2001:5555:7777::50 |
| eth2 | 2001:5555:cccc::50 |
| r6 | eth0 | 2001:2222:6666::60 |
| eth1 | 2001:6666:8888::60 |
| eth2 | 2001:6666:cccc::60 |
| r7 | eth0 | 2001:1111:7777::70 |
| eth1 | 2001:5555:7777::70 |
| eth2 | 2001:7777:cccc::70 |
| r8 | eth0 | 2001:2222:8888::80 |
| eth1 | 2001:6666:8888::80 |
| eth2 | 2001:8888:cccc::80 |
| h11 | eth0 | 2001:5555:cccc::11 |
| h12 | eth0 | 2001:6666:cccc::12 |
| h13 | eth0 | 2001:7777:cccc::13 |
| h14 | eth0 | 2001:8888:cccc::14 |

# 

# Consignas

# Preparación de entorno

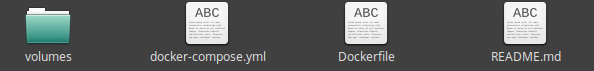
# 1.- Sobre Desktop instalar Docker CE, docker-compose y git



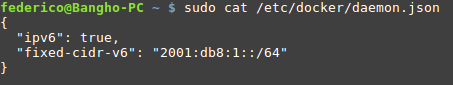




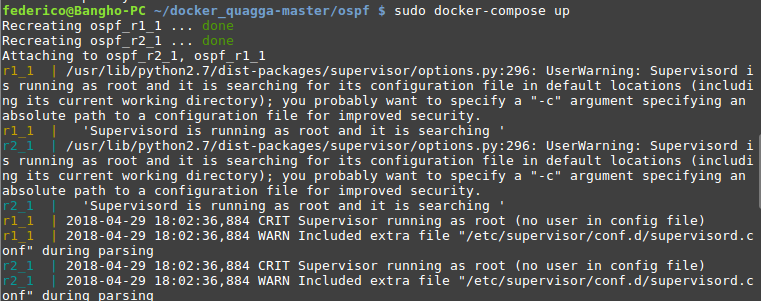
# 2.- Clonar el siguiente repositorio: ​<https://github.com/maticue/docker_quagga>



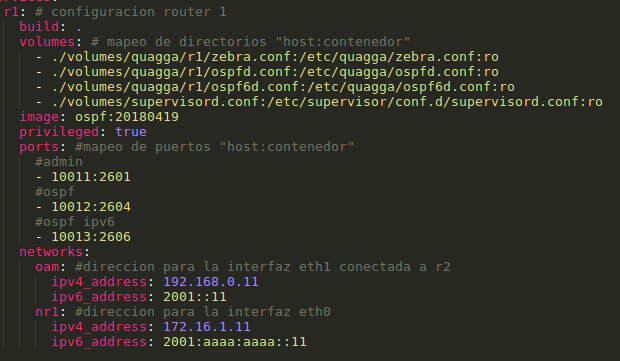
# 3.- Siguiendo las instrucciones del repositorio, configurar Docker CE con soporte para IPv6



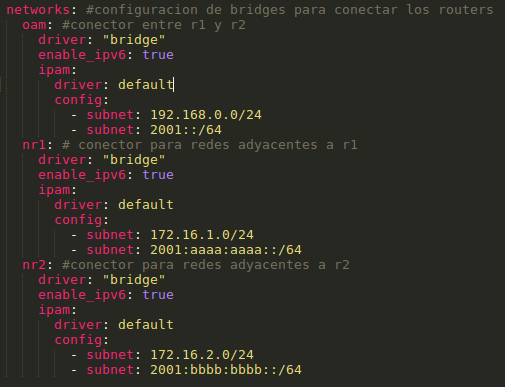
# 4.- Siguiendo las instrucciones del repositorio, probar de iniciar el entorno de pruebas.



# 5.- Leer el archivo docker-compose.yml e identificar cada sección.







# 5.1.- ¿En que puerto escucha el servicio OSPFv3 para IPv6? Conectarse usando telnet

Para el caso de r1 es 10013 y para r2 es 10023.

Conexión a al servicio OSPFv6 en r1:

sudo telnet localhost 10013

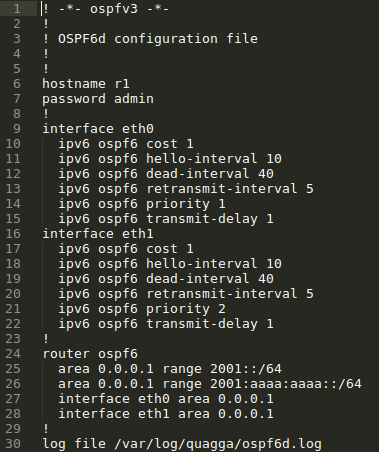
Conexión a al servicio OSPFv6 en r2:

sudo telnet localhost 10023

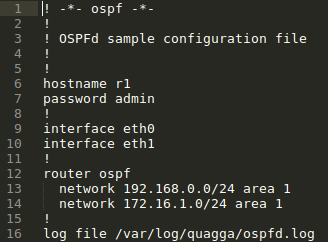
# 6.- Analizar los archivos de configuración de los servicios.

Configuración de servicios para r1:

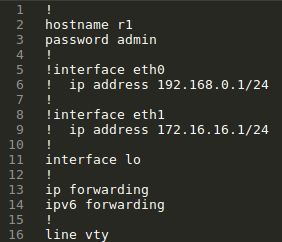
Servicio OSPFv6:



Servicio OSPF:

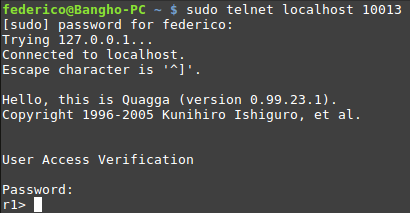


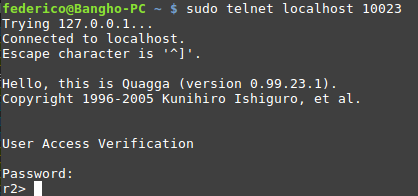
Servicio Zebra:



# 6.1.- Identificar el password de los servicios y utilizarlo para autenticarse en la conexión telnet creada en el punto anterior.

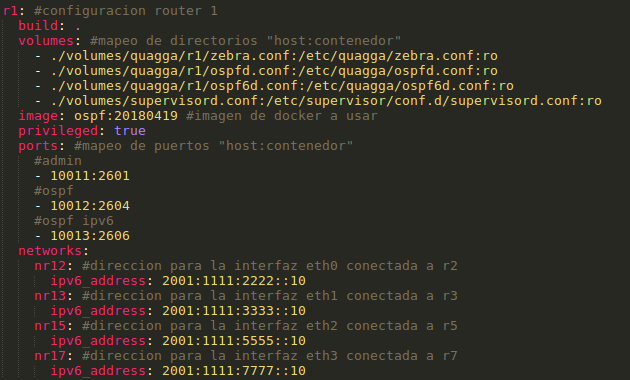
Para cualquiera de los tres servicios el password es admin

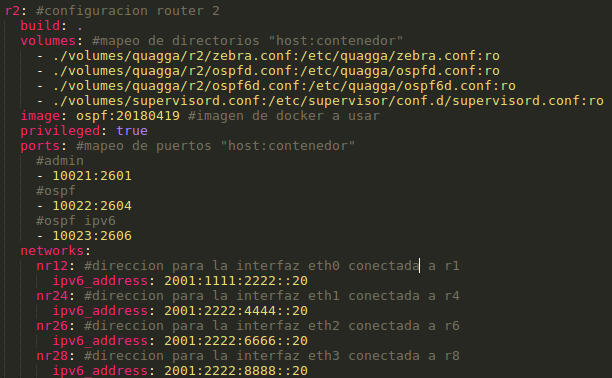


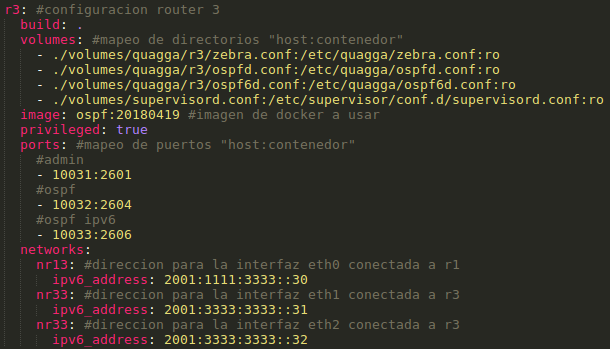


# Creación de entorno

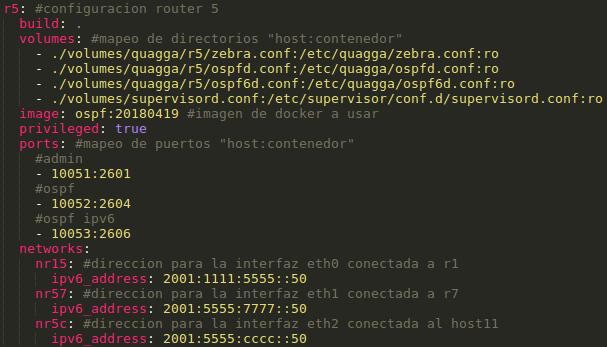
# 7.- Modificar el archivo docker-compose para replicar la topología definida en el diagrama.

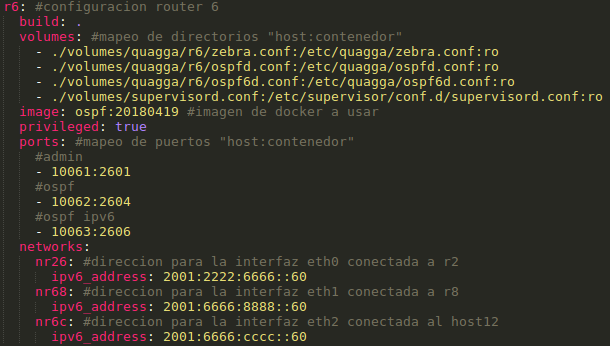


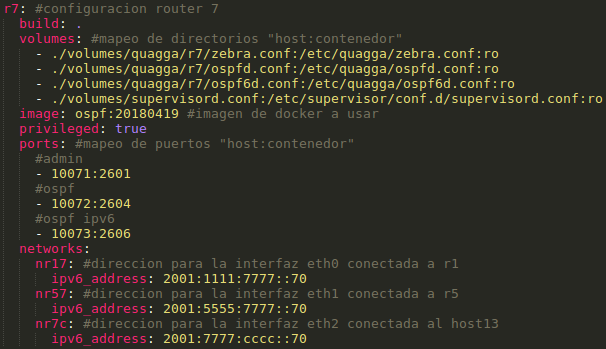


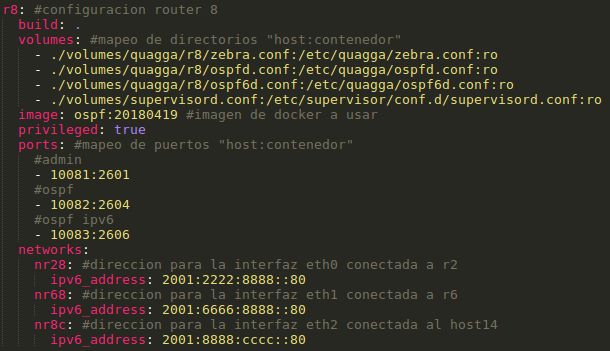


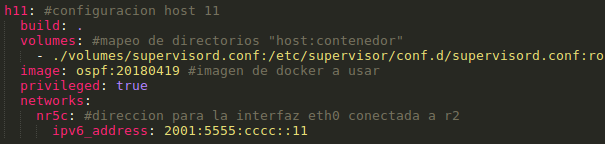


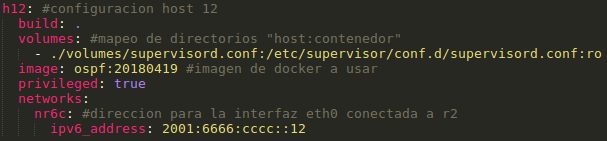


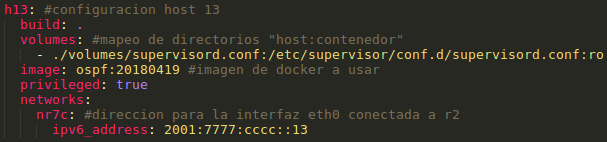


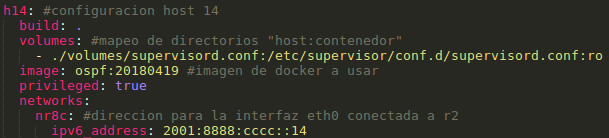










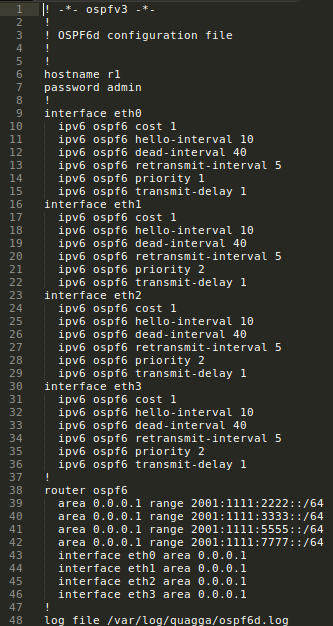


# 8.- Crear y modificar los archivos de configuración para cada router.



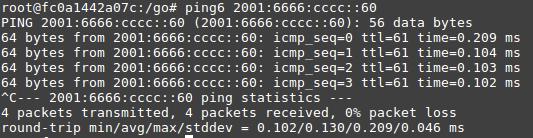
# 8.1.- Configurar cada router para que funcione OSPF.

Ejemplo para r1:



# 9.- Probar interconexión entre los distintos puntos y verificar que las tablas de ruteo de los routers muestran las rutas OSPF.

Prueba ping desde h11 a h12:



Prueba ping desde h13 a h14:

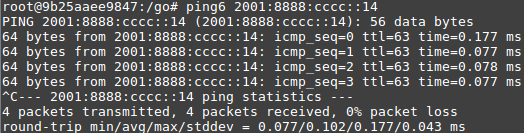


Tabla de ruteo de r1:

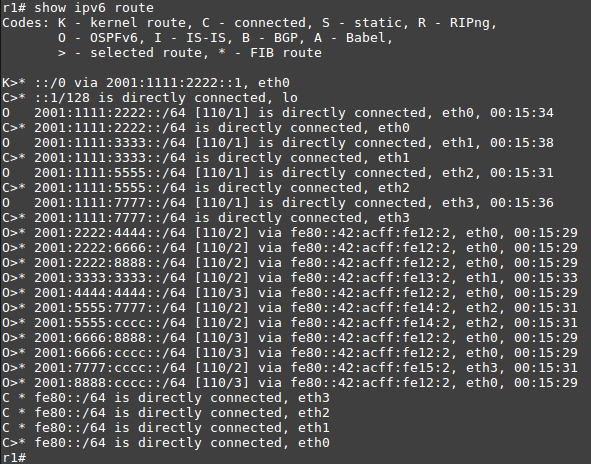


Tabla de ruteo de r2:

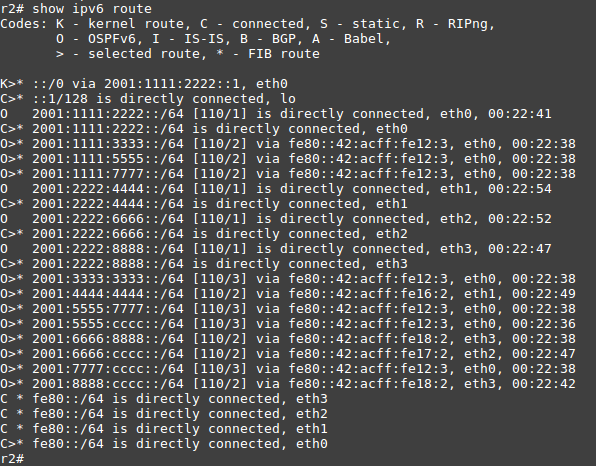


Tabla de ruteo de r5:

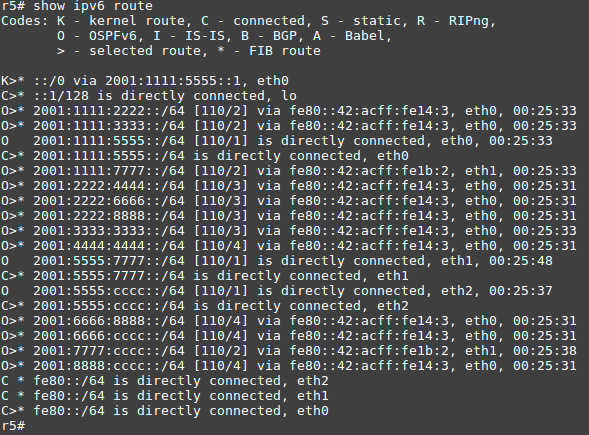
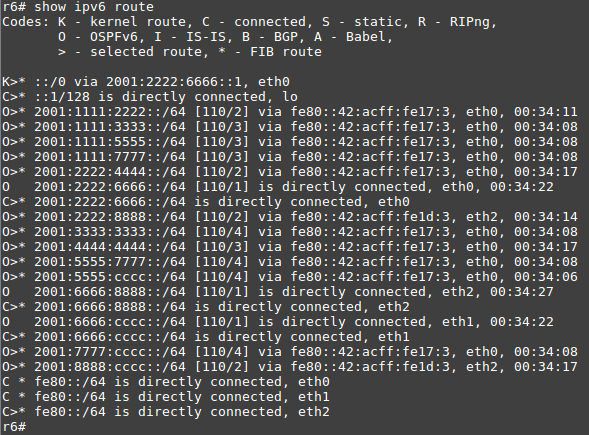
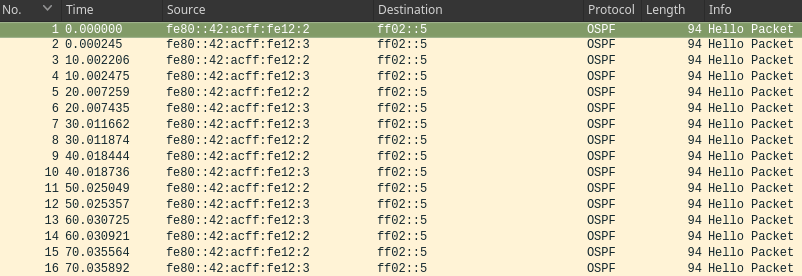


Tabla de ruteo de r6:



# 10.- Identificar y Analizar los mensajes de OSPF.

Router 1 (eth0):



En la figura se puede ver direcciones IPv6 que son local-link.

local-link r2= fe80::42:acff:fe12:2

local-link r1= fe80::42:acff:fe12:3

El destino ff02::5 es la dirección multicast de OSPFv3

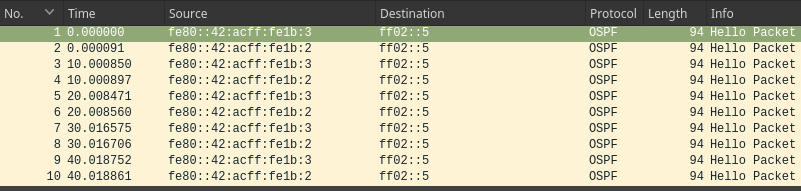
Router 2 (eth2):



local-link r2= fe80::42:acff:fe17:3

local-link r6= fe80::42:acff:fe17:2

Router 5 (eth1):



local-link r5= fe80::42:acff:fe1b:3

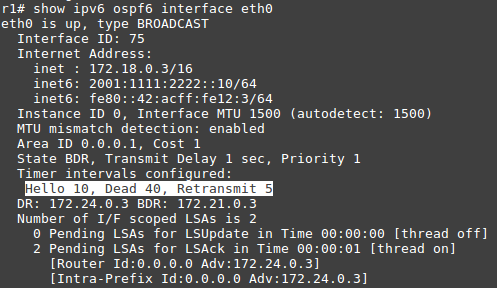
local-link r8= fe80::42:acff:fe1b:2

# 11.- Modificar al menos 2 parámetros de configuración de OSPF y demostrar que cambios se producen.

En el router 1 se eligió cambiar el parámetro hello-interval en la interfaz eth0. Este parámetro regula (en segundos) el intervalo en el que el router envía los mensajes hello por la interfaz que se desee configurar.

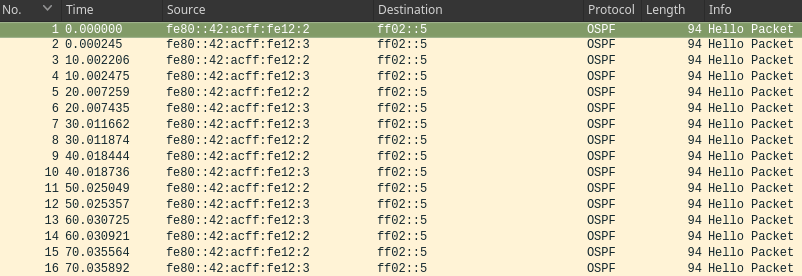
El segundo parámetro que se eligió cambiar es cost en el router 6 en la interfaz eth1 del router. Esto cambia el la forma en la que los routers van a reenviar los paquetes, la prueba se realizará haciendo ping entre los host 12 y 14.

Parámetro hello-interval antes del cambio:



Se puede apreciar en la parte resaltada de la figura que en parámetro Hello vale 10 segundos.

Hello Packet análisis de intervalo:



Con wireshark se podemos verificar en la columna Time que efectivamente ese es el intervalo que se envían los mensajes.

Se cambia los el parámetro hello-interval de la siguiente manera:

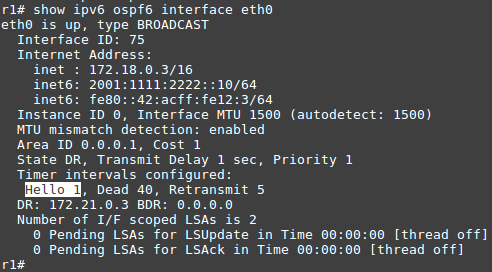
r1# configure terminal

r1(config)# interface eth0

r1(config-if)# ipv6 ospf6 hello-interval 1

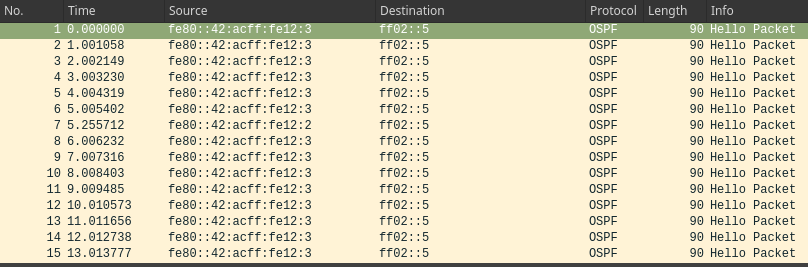
Para este caso se le especificó al router que cambie a un segundo el intervalo de envío.

Comprobamos nuevamente los valores de parámetros para la interfaz eth0:



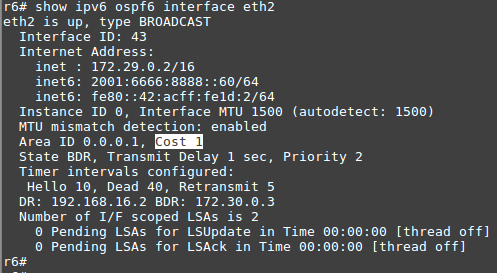
Se puede ver que el intervalo bajo a 1.

Nuevamente vemos el intervalo con el que se envía los mensajes Hello:



Se puede ver que ahora el intervalo cambio a un segundo.

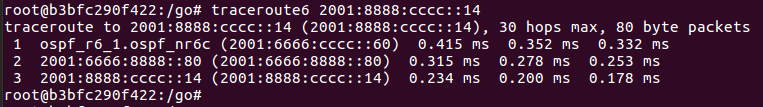
Ahora se modificará el parámetro cost en la interfaz eth1 de r6. Primero vemos cual es su valor por defecto



Se puede ver resaltado que Cost vale 1.

Con traceroute se puede ver la ruta que toma hacer para enviar una paquete desde h12 a h14:

\*a partir de ahora se cambio de SO por eso las capturas tomada a los terminales pueden diferir a las anteriores



Se lo modificamos cost a 200 mediante los comandos:

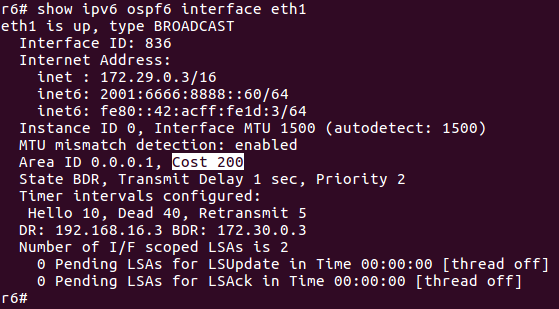
Se cambia los el parámetro hello-interval de la siguiente manera:

r6# configure terminal

r6(config)# interface eth1

r6(config-if)# ipv6 ospf6 cost 200

Se observan los cambios:



Nuevamente mediante el comando traceroute vemos la ruta que toman los paquetes de h12 a h14: