

# Memoria

Pedro Gómez Martín y Mario Aguilera Alcalde

1 de junio de 2021

## Índice

<b>1. Tarea 1: Configuración del direccionamiento IP</b>	<b>3</b>
1.1. Indique los comandos necesarios para añadir dichas direcciones IP y para eliminarlas: . . .	3
<b>2. Tarea 2: Configuración de rutas estáticas</b>	<b>4</b>
2.1. Indique y explique brevemente los comandos introducidos para realizar este apartado . . .	4
2.2. Explique cómo ha comprobado la conectividad y muestre el resultado de los comandos que demuestran dicha conectividad. . . . .	4
2.3. Indique los comandos necesarios para eliminar las rutas creadas y muestre cómo dichas rutas se han eliminado. . . . .	5
<b>3. Tarea 3: Configuración de protocolos de encaminamiento</b>	<b>5</b>
3.1. Explique detalladamente los pasos que siga para realizar este apartado, así como el contenido de los ficheros de configuración que haya creado o modificado . . . . .	5
3.1.1. DAEMONS: . . . . .	5
3.1.2. OSPFD.CONF: . . . . .	5
3.1.3. ZEBRA.CONF . . . . .	5
3.2. Explique cómo ha comprobado la conectividad y muestre el resultado de los comandos que demuestran dicha conectividad . . . . .	5
3.3. Empleando Wireshark, analice los paquetes OSPF observando cual es el intervalo de tiempo entre dos paquetes hello consecutivos. Cambie dicho intervalo de tiempo e indique los pasos seguidos, así como muestre mediante capturas de Wireshark que dicho intervalo ha sido cambiado con éxito. . . . .	6
<b>4. Tarea 4: Configuración de túneles, protocolos de encaminamiento e IPv6</b>	<b>7</b>
4.1. Tarea 4.1: Definición de túnel IPv6 sobre IPv4 . . . . .	7
4.1.1. Túnel VM1 a VM2 . . . . .	7
4.1.2. Túnel VM2 a VM1 . . . . .	7
4.1.3. Comprobación . . . . .	7
4.1.4. ¿Debe emplearse el mismo nombre de túnel en VM1 y en VM2? Justifique su respuesta. . . . .	8
4.2. Tarea 4.2: Conectividad IPv6 mediante RIPng . . . . .	8
4.2.1. Explique detalladamente los pasos que siga para realizar este apartado, así como el contenido de los ficheros de configuración que haya modificado. . . . .	8
4.2.2. Explique cómo ha comprobado la conectividad y muestre el resultado de los comandos que demuestran dicha conectividad . . . . .	8
4.2.3. Desde VM1 haga un ping a la dirección 200::2:1 y, capturando uno de dichos pings (en sentido de ida) con Wireshark, rellene la siguiente tabla: . . . . .	9

**Nota:**

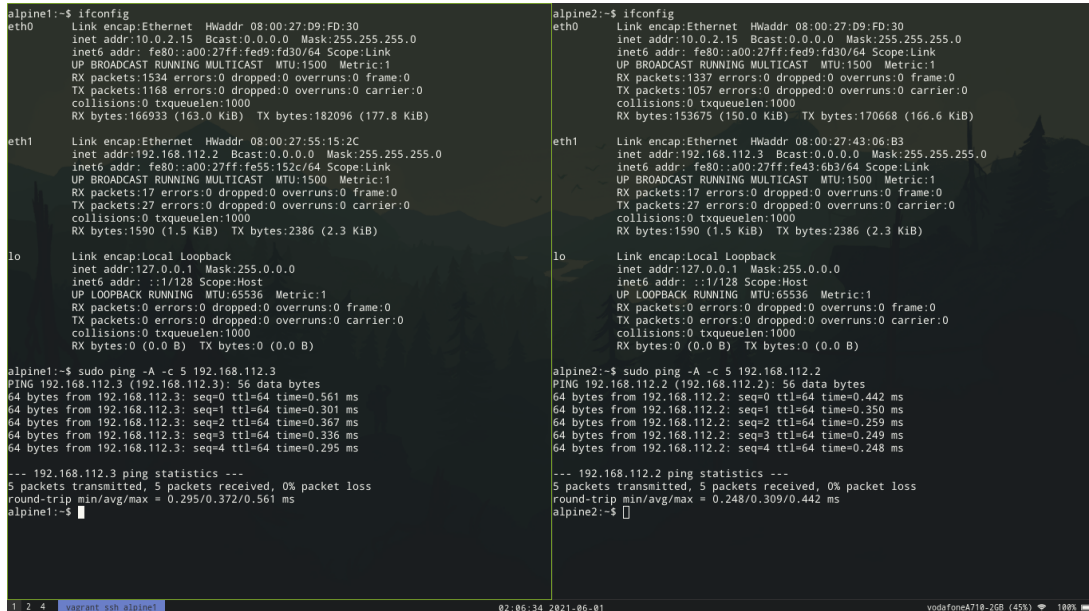
Se ha utilizado Vagrant con el backend de virtualbox para la creación de las máquinas virtuales. Como podemos ver en el archivo **Vagrantfile**, la distribución utilizada es Alpine Linux por su ligereza y velocidad a la hora de iniciarse.

Se ha utilizado Vagrant para hacer más fácil la reproducción de esta práctica ya que configura las máquinas virtuales automáticamente en base al archivo de configuración, asegurando así un entorno idéntico.

# 1. Tarea 1: Configuración del direccionamiento IP

En esta primera parte de la práctica, únicamente se pide que configure una de las VMs que está ejecutando para asignarle la dirección IP 10.10.5.5, con una máscara de 32 bits en la interfaz loopback y la dirección IPv6 3ff::5 con una máscara de 122 bits en la interfaz que conecta las VMs entre sí. Una vez haya realizado estos pasos, elimine dichas asignaciones para volver al estado inicial. NOTA: no elimine las direcciones IP ya existentes en su equipo.

En la primera foto se observa la prueba de conectividad al empezar las máquinas:



The image shows two terminal windows side-by-side. The left window is for 'alpine1' and the right for 'alpine2'. Both show the output of the 'ifconfig' command for interfaces eth0, eth1, and lo. In alpine1, eth0 has IP 10.0.2.15, eth1 has IP 192.168.112.2, and lo has IP 127.0.0.1. In alpine2, eth0 has IP 10.0.2.15, eth1 has IP 192.168.112.3, and lo has IP 127.0.0.1. Below the ifconfig output, both terminals show a successful ping test from 192.168.112.3 to 192.168.112.2, with 5 packets transmitted and received, and 0% packet loss.

## 1.1. Indique los comandos necesarios para añadir dichas direcciones IP y para eliminarlas:

```
sudo ip addr add 10.10.5.5 dev lo
sudo ip addr del 127.0.0.1 dev lo
sudo ip addr add 3ff::5/122 dev lo
```

```

alpine1:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:D9:FD:30
          inet addr:10.0.2.15  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fed9:fd30/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1633 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1257 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:173327 (169.2 KiB)  TX bytes:190224 (185.7 KiB)

eth1      Link encap:Ethernet  HWaddr 08:00:27:55:15:2C
          inet addr:192.168.112.2  Bcast:0.0.0.0  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe55:152c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29 errors:0 dropped:0 overruns:0 frame:0
          TX packets:42 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2690 (2.6 KiB)  TX bytes:3696 (3.6 KiB)

lo        Link encap:Local Loopback
          inet addr:10.10.5.5  Mask:255.255.255.255
          inet6 addr: 3ff::5/122 Scope:Global
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

```

```
sudo ip addr del ::1 dev lo
```

## 2. Tarea 2: Configuración de rutas estáticas

Para dotar de conectividad a un conjunto de redes que no están directamente conectadas (en cuyo caso Linux lo hace automáticamente) se va a comenzar haciendo uso de la herramienta ip para instalar rutas de manera manual. Puesto que la maqueta compuesta de dos VMs de la que se dispone no tiene redes que no se encuentren directamente conectadas, se va a emplear la interfaz de loopback de dichas VMs para definir en cada una de ellas una red diferente, las cuales no están directamente conectadas.

### 2.1. Indique y explique brevemente los comandos introducidos para realizar este apartado

```

# sudo ip route add {RED} via {IP} dev {DISPOSITIVO}
sudo sysctl -w net.ipv4.ip_forward=1
sudo ip route add 10.10.2.0/24 via 192.168.112.3
sudo ip route add 10.10.1.0/24 via 192.168.112.2

```

### 2.2. Explique cómo ha comprobado la conectividad y muestre el resultado de los comandos que demuestran dicha conectividad.

```
sudo ping -A -c 10 10.10.2.1
```

```

alpine1:~$ ip route show
default via 10.0.2.2 dev eth0 metric 202
10.0.2.0/24 dev eth0 scope link src 10.0.2.15
10.10.2.0/24 via 192.168.112.3 dev eth1
192.168.112.0/24 dev eth1 scope link src 192.168.112.2
alpine1:~$ sudo ping -A -c 10 10.10.2.1
PING 10.10.2.1 (10.10.2.1): 56 data bytes
64 bytes from 10.10.2.1: seq=0 ttl=64 time=0.455 ms
64 bytes from 10.10.2.1: seq=1 ttl=64 time=0.294 ms
64 bytes from 10.10.2.1: seq=2 ttl=64 time=0.364 ms
64 bytes from 10.10.2.1: seq=3 ttl=64 time=0.300 ms
64 bytes from 10.10.2.1: seq=4 ttl=64 time=0.259 ms
64 bytes from 10.10.2.1: seq=5 ttl=64 time=0.346 ms
64 bytes from 10.10.2.1: seq=6 ttl=64 time=0.347 ms
64 bytes from 10.10.2.1: seq=7 ttl=64 time=0.351 ms
64 bytes from 10.10.2.1: seq=8 ttl=64 time=0.267 ms
64 bytes from 10.10.2.1: seq=9 ttl=64 time=0.269 ms
--- 10.10.2.1 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.259/0.325/0.455 ms

alpine2:~$ ip route show
default via 10.0.2.2 dev eth0 metric 202
10.0.2.0/24 dev eth0 scope link src 10.0.2.15
10.10.1.0/24 via 192.168.112.2 dev eth1
192.168.112.0/24 dev eth1 scope link src 192.168.112.3
alpine2:~$ sudo ping -A -c 10 10.10.1.1
PING 10.10.1.1 (10.10.1.1): 56 data bytes
64 bytes from 10.10.1.1: seq=0 ttl=64 time=0.493 ms
64 bytes from 10.10.1.1: seq=1 ttl=64 time=0.325 ms
64 bytes from 10.10.1.1: seq=2 ttl=64 time=0.283 ms
64 bytes from 10.10.1.1: seq=3 ttl=64 time=0.277 ms
64 bytes from 10.10.1.1: seq=4 ttl=64 time=0.305 ms
64 bytes from 10.10.1.1: seq=5 ttl=64 time=0.280 ms
64 bytes from 10.10.1.1: seq=6 ttl=64 time=0.322 ms
64 bytes from 10.10.1.1: seq=7 ttl=64 time=0.289 ms
64 bytes from 10.10.1.1: seq=8 ttl=64 time=0.357 ms
64 bytes from 10.10.1.1: seq=9 ttl=64 time=0.287 ms
--- 10.10.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.277/0.321/0.493 ms

```

## 2.3. Indique los comandos necesarios para eliminar las rutas creadas y muestre cómo dichas rutas se han eliminado.

```

sudo ip route del 10.10.2.0/24 via 192.168.112.3
sudo ip route del 10.10.1.0/24 via 192.168.112.2

```

```

alpine1:~$ sudo ip route del 10.10.2.0/24 via 192.168.112.3
alpine1:~$ ip route show
default via 10.0.2.2 dev eth0 metric 202
10.0.2.0/24 dev eth0 scope link src 10.0.2.15
192.168.112.0/24 dev eth1 scope link src 192.168.112.2

alpine2:~$ sudo ip route del 10.10.1.0/24 via 192.168.112.2
alpine2:~$ ip route show
default via 10.0.2.2 dev eth0 metric 202
10.0.2.0/24 dev eth0 scope link src 10.0.2.15
192.168.112.0/24 dev eth1 scope link src 192.168.112.3

```

## 3. Tarea 3: Configuración de protocolos de encaminamiento

Esta tarea tiene el mismo objetivo que la tarea 2, pero ahora el encaminamiento no va a realizarse de manera estática sino dinámica, para lo cual va a hacerse uso de protocolos de encaminamiento. Más concretamente, se va a emplear el protocolo OSPF para tener conectividad entre la red 10.10.1.1/32 y 10.10.2.1/32, definidas en las interfaces de loopback de la VM1 y VM2, respectivamente.

### 3.1. Explique detalladamente los pasos que siga para realizar este apartado, así como el contenido de los ficheros de configuración que haya creado o modificado

#### 3.1.1. DAEMONS:

```

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

```

#### 3.1.2. OSPFD.CONF:

```

interface eth1
ip ospf hello-interval 5
router ospf
redistribute connected
network 192.168.112.0/24 area 0.0.0.1

```

#### 3.1.3. ZEBRA.CONF

```

interface eth1
interface lo

```

### 3.2. Explique cómo ha comprobado la conectividad y muestre el resultado de los comandos que demuestran dicha conectividad

Mediante un ping desde una VM y luego desde la otra.

```

alpine1:/etc/quagga$ ip r
default via 10.0.2.2 dev eth0 metric 202
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15
10.10.2.1 via 192.168.112.3 dev eth1 proto zebra metric 20
192.168.112.0/24 dev eth1 proto kernel scope link src 192.168.112.2
alpine1:/etc/quagga$ sudo ping -A -c 10 10.10.1.1
PING 10.10.1.1 (10.10.1.1): 56 data bytes
64 bytes from 10.10.1.1: seq=0 ttl=64 time=0.459 ms
64 bytes from 10.10.1.1: seq=1 ttl=64 time=0.359 ms
64 bytes from 10.10.1.1: seq=2 ttl=64 time=0.334 ms
64 bytes from 10.10.1.1: seq=3 ttl=64 time=0.280 ms
64 bytes from 10.10.1.1: seq=4 ttl=64 time=0.366 ms
64 bytes from 10.10.1.1: seq=5 ttl=64 time=0.295 ms
64 bytes from 10.10.1.1: seq=6 ttl=64 time=0.277 ms
64 bytes from 10.10.1.1: seq=7 ttl=64 time=0.248 ms
64 bytes from 10.10.1.1: seq=8 ttl=64 time=0.288 ms
64 bytes from 10.10.1.1: seq=9 ttl=64 time=0.282 ms

--- 10.10.2.1 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.248/0.318/0.459 ms

alpine2:/etc/quagga$ ip r
default via 10.0.2.2 dev eth0 metric 202
10.0.2.0/24 dev eth0 proto kernel scope link src 10.0.2.15
10.10.1.1 via 192.168.112.2 dev eth1 proto zebra metric 20
192.168.112.0/24 dev eth1 proto kernel scope link src 192.168.112.3
alpine2:/etc/quagga$ sudo ping -A -c 10 10.10.1.1
PING 10.10.1.1 (10.10.1.1): 56 data bytes
64 bytes from 10.10.1.1: seq=0 ttl=64 time=0.526 ms
64 bytes from 10.10.1.1: seq=1 ttl=64 time=0.353 ms
64 bytes from 10.10.1.1: seq=2 ttl=64 time=0.340 ms
64 bytes from 10.10.1.1: seq=3 ttl=64 time=0.282 ms
64 bytes from 10.10.1.1: seq=4 ttl=64 time=0.284 ms
64 bytes from 10.10.1.1: seq=5 ttl=64 time=0.314 ms
64 bytes from 10.10.1.1: seq=6 ttl=64 time=0.246 ms
64 bytes from 10.10.1.1: seq=7 ttl=64 time=0.265 ms
64 bytes from 10.10.1.1: seq=8 ttl=64 time=0.261 ms
64 bytes from 10.10.1.1: seq=9 ttl=64 time=0.273 ms

--- 10.10.1.1 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.246/0.314/0.526 ms

```

- 3.3. Empleando Wireshark, analice los paquetes OSPF observando cual es el intervalo de tiempo entre dos paquetes hello consecutivos. Cambie dicho intervalo de tiempo e indique los pasos seguidos, así como muestre mediante capturas de Wireshark que dicho intervalo ha sido cambiado con éxito.

#### INTERVALO DE 2 SEC EN EL WIRESHARK EN EL HELLO-INTERVAL

The screenshot shows a Wireshark capture of OSPF Hello packets. The packet list on the left shows 17 packets, all of which are OSPF Hello packets. The packet details pane on the right shows the 'Hello Interval [sec]: 2' field under the 'Options' section. The packet bytes pane at the bottom shows the raw data of the selected packet, which is an OSPF Hello packet.

#### INTERVALO DE 5 SEC EN EL WIRESHARK EN EL HELLO-INTERVAL

The screenshot shows a Wireshark capture of OSPF Hello packets. The packet list on the left shows 17 packets, all of which are OSPF Hello packets. The packet details pane on the right shows the 'Hello Interval [sec]: 5' field under the 'Options' section. The packet bytes pane at the bottom shows the raw data of the selected packet, which is an OSPF Hello packet.

## 4. Tarea 4: Configuración de túneles, protocolos de encaminamiento e IPv6

Partiendo del resultado de la tarea 3, donde se dispone de tres redes y se emplea OSPF para dotar de conectividad a las redes que no están directamente conectadas (las definidas en las interfaces loopback de las VMs), en esta tarea se va a configurar un túnel IPv6 sobre IPv4 que interconectará dos encaminadores, utilizando OSPF como protocolo de encaminamiento de IPv4 y RIPng como protocolo de encaminamiento de IPv6. Para la realización de esta tarea, se requieren dos subtareas:

### 4.1. Tarea 4.1: Definición de túnel IPv6 sobre IPv4

Definición del túnel IPv6 sobre IPv4:

#### 4.1.1. Túnel VM1 a VM2

```
sudo ip tunnel add mape mode sit local 192.168.112.2 remote 192.168.112.3
ip tunnel show
```

```
alpine1:~# sudo ip tunnel add mape mode sit local 192.168.112.2 remote 192.168.112.3
alpine1:~# ip tunnel show
sit0: ipv6/ip remote any local any ttl 64 nopmtudisc 6rd-prefix 2002::/16
mape: ipv6/ip remote 192.168.112.3 local 192.168.112.2 ttl inherit 6rd-prefix 2002::/16
```

#### 4.1.2. Túnel VM2 a VM1

```
sudo ip tunnel add mape mode sit local 192.168.112.3 remote 192.168.112.2
ip tunnel show
```

```
alpine2:~$ sudo ip tunnel add mape mode sit local 192.168.112.3 remote 192.168.112.2
alpine2:~$ ip tunnel show
sit0: ipv6/ip remote any local any ttl 64 nopmtudisc 6rd-prefix 2002::/16
mape: ipv6/ip remote 192.168.112.2 local 192.168.112.3 ttl inherit 6rd-prefix 2002::/16
```

#### 4.1.3. Comprobación

1. Comprobamos el túnel en VM1:

```
sudo ip addr add 200::2:1/128 dev lo
ip addr show
```

```
alpine1:~# sudo ip addr add 200::2:1/128 dev lo
alpine1:~# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 200::2:1/128 scope global
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:d9:fd:30 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed9:fd30/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:55:15:2c brd ff:ff:ff:ff:ff:ff
    inet 192.168.112.2/24 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe55:152c/64 scope link
        valid_lft forever preferred_lft forever
4: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
5: mape@NONE: <POINTOPOINT,NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 192.168.112.2 peer 192.168.112.3
```

2. Comprobamos el túnel en VM2:

```
sudo ip addr add 200::1:1/128 dev lo
ip addr show
```



```

alpine2:~$ sudo ip addr add 200::1:1/128 dev lo
alpine2:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 200::1:1/128 scope global
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:d9:fd:30 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fed9:fd30/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:43:06:b3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.112.3/24 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe43:6b3/64 scope link
        valid_lft forever preferred_lft forever
4: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
6: mape@NONE: <POINTOPOINT,NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 192.168.112.3 peer 192.168.112.2

```

#### 4.1.4. ¿Debe emplearse el mismo nombre de túnel en VM1 y en VM2? Justifique su respuesta.

No tiene por qué, ya que lo que importa son las direcciones.

## 4.2. Tarea 4.2: Conectividad IPv6 mediante RIPng

### 4.2.1. Explique detalladamente los pasos que siga para realizar este apartado, así como el contenido de los ficheros de configuración que haya modificado.

Para usar RIPng editaremos el archivo zebra:

```

interface eth1
interface mape
interface lo

```

Y crearemos un archivo nuevo llamado ripngd:

```

router ripng
network mape
redistribute connected

```

Y hacemos un restart antes de comprobar la conectividad:

```

alpine1:~$ sudo rc-service zebra restart
* Stopping ripngd ... [ ok ]
* Stopping ospfd ... [ ok ]
* Stopping zebra ... [ ok ]
* Cleaning up stale zebra routes ... [ ok ]
* Starting zebra ... [ ok ]
alpine1:~$ * Starting ospfd ...
* Starting ripngd ... [ ok ]

```

### 4.2.2. Explique cómo ha comprobado la conectividad y muestre el resultado de los comandos que demuestran dicha conectividad

Mediante un ping en ambas maquinas:



```

alpine1:/etc/quagga$ sudo ping -A -c 10 200::2:1
PING 200::2:1 (200::2:1): 56 data bytes
64 bytes from 200::2:1: seq=0 ttl=64 time=0.146 ms
64 bytes from 200::2:1: seq=1 ttl=64 time=0.028 ms
64 bytes from 200::2:1: seq=2 ttl=64 time=0.144 ms
64 bytes from 200::2:1: seq=3 ttl=64 time=0.185 ms
64 bytes from 200::2:1: seq=4 ttl=64 time=0.154 ms
64 bytes from 200::2:1: seq=5 ttl=64 time=0.025 ms
64 bytes from 200::2:1: seq=6 ttl=64 time=0.138 ms
64 bytes from 200::2:1: seq=7 ttl=64 time=0.024 ms
64 bytes from 200::2:1: seq=8 ttl=64 time=0.133 ms
64 bytes from 200::2:1: seq=9 ttl=64 time=0.125 ms

--- 200::2:1 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 0.024/0.110/0.185 ms

```

- 4.2.3. Desde VM1 haga un ping a la dirección 200::2:1 y, capturando uno de dichos pings (en sentido de ida) con Wireshark, rellene la siguiente tabla:

Pila de protocolos completa	IPv6 y IPv4
Dirección IP origen IPv4	10.0.2.15
Dirección IP destino IPv4	10.0.2.4
Dirección IP origen IPv6	200::1:1
Dirección IP destino IPv6	200::2:1
TTL	64
Hop limit	64
TOS (o DSCP)	0