Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

dar-pr-2

Diseño y administración de redes

Autor 1:	Toral Pallás, Héctor - 798095
Autor 2:	Lahoz Bernad, Fernando - 800989
Autor 3:	Martínez Lahoz, Sergio - 801621
Grado:	Ingeniería Informática
Curso:	2023-2024

9 de octubre de 2023

Índice

1. Esquema de diagrama de red 2
2. Pregunta 1 3
3. Pregunta 2 4
4. Pregunta 3 5
5. Pregunta 5 6
6. Pregunta 6 7
7. Pregunta 7 8
8. Pregunta 8 10
9. Pregunta 9 11

01

Esquema de diagrama de red

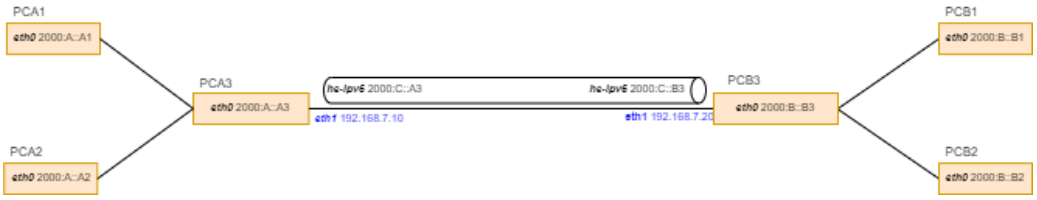

Figura 1: Esquema del tunel ipv6.

02

Pregunta 1

Capturar en la interfaz eth0 del router (no en la interfaz túnel creada) y verificar que, efectivamente, el tráfico desde el router hacia el exterior es IPv6 sobre IPv4. Muéstralo indicando las direcciones IPv6 e IPv4 que aparecen en los paquetes

---

En la captura de red podemos observar que en la pila de protocolos aparecen ambos protocolos de red: IPv4 por debajo de IPv6. Tomamos como ejemplo el paquete 7. El origen del mensaje es PCB3, que a nivel IPv6 utiliza la dirección de la interfaz del túnel (2000:c::b3) y a nivel IPv4 usa la de eth1 (192.168.7.20). El destino es PCA3, que al igual que PCB3, emplea la dirección del túnel (2000:c::a3) a nivel IPv6 y la de la interfaz eth1 para el encapsulamiento en IPv4 (192.168.7.10).

03

Pregunta 2

Observa el valor MTU configurado en la interfaz tipo túnel (ip 6 link show dev [nombre_tunel] / ifconfig [nombre_tunel]). Justifica dicho valor teniendo en cuenta la información capturada previamente.

Ejecutando el comando *ip -6 link show dev he-ipv6* en PCA3 y PCB3 obtenemos que el MTU es de 1480 bytes:

```
7: he-ipv6: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1480 qdisc noqueue state UNKNOWN
    link/sit 192.168.7.20 peer 192.168.7.10
```

```
8: he-ipv6: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1480 qdisc noqueue state UNKNOWN
    link/sit 192.168.7.10 peer 192.168.7.20
```

En la captura de la anterior pregunta hemos visto que IPv6 viaja encapsulado en IPv4. Considerando que el MTU por defecto para las interfaces IPv4 es 1500, se puede deducir fácilmente que esos 20 bytes de diferencia corresponden al tamaño de la cabecera IPv4.

04

Pregunta 3

Una vez configurado el escenario completo, se comprobará la conectividad del mismo verificando la comunicación entre los equipos de las redes LAN A y LAN B mediante comandos como ping6, o traceroute6

Se ha realizado un ping de PCA1 hasta PCB1 mediante el comando *ping6 -c 3 2000:B::B1* y otro de PCA2 a PCB2 *ping6 -c 3 2000:B::B2*, mientras capturamos en todas las redes. Hemos comprobado que todos los paquetes ICMPv6 echo request reciben su correspondiente response.

PCA1 → PCB1

```
[root@localhost ~]# ping6 -c 3 2000:b::b1
PING 2000:b::b1(2000:b::b1) 56 data bytes
64 bytes from 2000:b::b1: icmp_seq=1 ttl=62 time=23.8 ms
64 bytes from 2000:b::b1: icmp_seq=2 ttl=62 time=5.88 ms
64 bytes from 2000:b::b1: icmp_seq=3 ttl=62 time=7.77 ms

--- 2000:b::b1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4049ms
rtt min/avg/max/mdev = 5.888/11.824/23.867/6.588 ms
```

05

Pregunta 5

Observa las direcciones IPv6 existentes en las interfaces de los PCs y pon un ejemplo de dirección local de enlace y dirección global. Muestra, en una de ellas, cómo se ha obtenido el identificador de interfaz de 64 bits (EUI-64).

Podemos comprobar el valor de las direcciones utilizando el comando ifconfig. Por ejemplo, para la interfaz eth0 de PCA1 obtenemos esta salida:

```
[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 0C:34:77:66:00:00
          inet6 addr: 2000:a::e34:77ff:fe66:0/64 Scope:Global
          inet6 addr: fe80::e34:77ff:fe66:0/64 Scope:Link
          inet6 addr: 2000:a::a1/64 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:115 errors:0 dropped:0 overruns:0 frame:0
          TX packets:448 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10932 (10.6 KiB)  TX bytes:38400 (37.5 KiB)
```

Podemos ver las dos direcciones globales asignadas:

- La dirección fijada por nosotros: 2000:a::a1.
- La dirección generada aleatoriamente: 2000:a::e34:77ff:fe66:0.

También aparece la dirección local de enlace: fe80::e34:77ff:fe66:0.

Tanto la dirección local de enlace como la dirección global aleatoria comparten los últimos 64 bits, el identificador de dispositivo. Este valor se obtiene siguiendo el procedimiento EUI-64, que se basa en la dirección MAC de la interfaz de red (HWaddr en la salida de ifconfig). El proceso consiste en separar la MAC en dos mitades, añadir los bytes 0xff y 0xfe entre las dos y poner a 1 el bit número 7, tal y como se muestra en la figura 2.

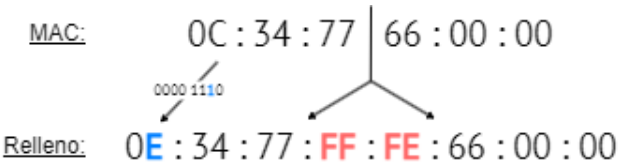

Figura 2: Descomposición de la dirección MAC en los 64 bits de EUI.

06

Pregunta 6

A partir de la captura indica los dos casos de procedimiento DAD descritos especificando en ambos: dirección multicast (Solicited Node Address) a la que se dirige el mensaje de ND (Neighbor Discovery, ICMPv6) y dirección unicast por la que se pregunta (target) Verifica la correspondencia entre las direcciones multicast y unicast identificadas. Muestra los paquetes de petición y respuesta (por parte del equipo y el router) de los parámetros necesarios para la autoconfiguración. Resalta dichos parámetros en el mensaje correspondiente.

Para esta tarea hemos desactivado y reactivador la interfaz de red de PCA1. Lo primero que hace es enviar dos mensajes (paquetes 1 y 2) a todos los routers que soporten el protocolo MLDv2 (PCA3 en este caso). El protocolo lo indica el tipo de ICMPv6: 143. Con estos mensajes, PCA1 se une al grupo de difusión del router, indicando la dirección multicast a la que deben escuchar dentro del campo records: **ff02::1:ff66:0**. Esta dirección es la Solicited Node Address que utiliza el siguiente paquete (3) para efectuar el primer procedimiento DAD. Se trata de un paquete ICMPv6 de tipo Neighbor Solicitation, en el que PCA1 incluye su dirección local de enlace en el campo target address para ver si alguien más la tiene: **fe80::e34:77ff:fe66:0**. Al no haber ningún mensaje Neighbor Advertisement se confirma que es única.

Con la dirección de enlace asegurada, toca configurar la global. Para ello PCA1 pregunta a todos los routers (ff02::2) los parámetros necesarios con un paquete de tipo Router Solicitation (paquete 5 en la captura). PCA3 le responde con un Router Advertisement (paquete 7), aunque no ese el que utiliza PCA1 para configurar su IP, sino uno enviado anteriormente de forma espontánea. Ambos contienen la misma información:

- ICMPv6 flags:
  - Flag M = 0 → la configuración de dirección es sin estado.
  - Flag O = 0 → la configuración de otros parámetros también es sin estado.
- Prefix information:
  - Flag L = 0 → toda dirección sin este prefijo es ajena a la red local y se debe considerar así a no ser que pasen 86400 s (valid lifetime).
  - Flag A = 0 → este prefijo sirve para autoconfigurar tu IP, y será válida al menos durante 14400 s (preferred lifetime).
  - Prefix → corresponde a los bits de red en la que se encuentran PCA1 y PCA3 y es el que debe se debe usar para autoconfigurar una dirección IPv6 local: **2000:a::**
- Link layer address → dirección MAC del router, PCA3, para que PCA1 la guarde en su Neighbor Cache: 0c:b1:38:19:00:00.

Con esta información PCA1 puede realizar el segundo DAD enviando otro Neighbor Solicitation (paquete 6) con la misma dirección multicast que el anterior (**ff02::1:ff66:0**), y que ahora incluye su dirección IPv6 global dentro del campo target address: **2000:a::e34:77ff:fe66:0**. Como no recibe ningún Neighbor Advertisement, se confirma que esa dirección es única dentro de la red local.

Un tiempo después se ejecuta un ping6 desde PCA1 a PCA3. Los paquetes 9 y 10 son el equivalente a una pregunta y respuesta ARP con el fin de obtener la dirección MAC destino. El flag Override activado del Neighbor Advertisement indica a PCA1 que actualice la entrada correspondiente de su Neighbor Cache. Una vez conocida la dirección MAC, PCA1 ya es capaz de enviar un paquete ICMPv6 echo request.

07

Pregunta 7

Realiza las siguientes conexiones, identificando la correspondencia entre dirección MAC destino (dirección de nivel de enlace – link layer – lladdr) y dirección IPv6 destino del paquete capturado en el equipo origen del ping:

PCA1 → PCA2

```
[root@localhost ~]# ping6 -c 5 2000:A::A2
PING 2000:A::A2(2000:a::a2) 56 data bytes
64 bytes from 2000:a::a2: icmp_seq=1 ttl=64 time=2.88 ms
64 bytes from 2000:a::a2: icmp_seq=2 ttl=64 time=2.10 ms
64 bytes from 2000:a::a2: icmp_seq=3 ttl=64 time=3.67 ms
64 bytes from 2000:a::a2: icmp_seq=4 ttl=64 time=3.30 ms
64 bytes from 2000:a::a2: icmp_seq=5 ttl=64 time=4.75 ms

--- 2000:A::A2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4031ms
rtt min/avg/max/mdev = 2.104/3.342/4.751/0.876 ms
```

Esta conexión es local a la red A, por lo que sólo aparecen mensajes ICMP en la captura de esa lan. Si nos fijamos en el paquete 2 (request) y el paquete 3 (response) de esa captura, podemos ver que el direccionamiento a nivel ethernet corresponde al direccionamiento a nivel IP, ya que no aparece ningún paquete en ninguna captura de otra red con las mismas direcciones IP de origen y destino.

PCB1 → PCB2

```
[root@localhost ~]# ping6 -c 5 2000:B::B2
PING 2000:B::B2(2000:b::b2) 56 data bytes
64 bytes from 2000:b::b2: icmp_seq=1 ttl=64 time=3.88 ms
64 bytes from 2000:b::b2: icmp_seq=2 ttl=64 time=4.75 ms
64 bytes from 2000:b::b2: icmp_seq=3 ttl=64 time=2.00 ms
64 bytes from 2000:b::b2: icmp_seq=4 ttl=64 time=5.81 ms
64 bytes from 2000:b::b2: icmp_seq=5 ttl=64 time=4.67 ms

--- 2000:B::B2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4037ms
rtt min/avg/max/mdev = 2.002/4.226/5.814/1.272 ms
```

Al igual que pasaba en el caso anterior, la conexión es local, y sólo aparecen mensajes ICMP en la captura de la red, por lo que el direccionamiento a nivel ethernet corresponde al de nivel IP. Los paquetes 13 y 14 de esa captura corresponden al primer ping.

PCA1 → PCB1

```
[root@localhost ~]# ping6 -c 5 2000:b::b1
PING 2000:b::b1(2000:b::b1) 56 data bytes
64 bytes from 2000:b::b1: icmp_seq=1 ttl=62 time=7.77 ms
64 bytes from 2000:b::b1: icmp_seq=2 ttl=62 time=23.8 ms
64 bytes from 2000:b::b1: icmp_seq=3 ttl=62 time=5.88 ms
64 bytes from 2000:b::b1: icmp_seq=4 ttl=62 time=13.8 ms
64 bytes from 2000:b::b1: icmp_seq=5 ttl=62 time=7.77 ms

--- 2000:b::b1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4049ms
rtt min/avg/max/mdev = 5.888/11.824/23.867/6.588 ms
```

En este caso, la comunicación se lleva a cabo entre máquinas de distinta red. Si observamos los paquetes 27 y 28 de la captura de la red A, 25 y 26 de la red C, y 30 y 31 de la red B podemos comprobar por el número de identificación IP que se tratan de mensajes request y response de la misma conexión (0x0c14). En cada red las direcciones MAC de origen y destino del paquete request son alteradas: en la red A la MAC destino es 0c:b1:38:19:00:01 que pasa a ser la MAC origen en la red C, y en esta red la MAC destino es 0c:ca:0c:e9:00:01, que pasa a ser la MAC origen en la red B. Estas direcciones corresponden a las máquinas PCA3 y PCB3, que están actuando de encaminadores. Para el paquete response ocurre el camino inverso.

PCB1 → PCA1

```
[root@localhost ~]# ping6 -c 5 2000:a::a1
PING 2000:a::a1(2000:a::a1) 56 data bytes
64 bytes from 2000:a::a1: icmp_seq=1 ttl=62 time=8.76 ms
64 bytes from 2000:a::a1: icmp_seq=2 ttl=62 time=23.4 ms
64 bytes from 2000:a::a1: icmp_seq=3 ttl=62 time=7.43 ms
64 bytes from 2000:a::a1: icmp_seq=4 ttl=62 time=27.6 ms
64 bytes from 2000:a::a1: icmp_seq=5 ttl=62 time=7.48 ms

--- 2000:a::a1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4027ms
rtt min/avg/max/mdev = 7.437/14.958/27.669/8.766 ms
```

Este caso es el camino inverso al anterior. A las direcciones MAC del paquete request les ocurre lo mismo que al paquete response del ping de PCA1 a PCB1, y para el paquete response es análogo. El primer ping de esta conexión se encuentra en los paquetes 40 y 41 de la captura de la red A, 45 y 46 de la red C, y paquetes 46 y 47 de la red B. El identificador de conexión IP en todos es 0xd909.

08

Pregunta 8

Comprueba las conexiones (mediante varios ping, paso a paso) entre PCA1/PCB1 y PCB1/PCA1, es decir comprobando que funciona el ping a cada una de las direcciones intermedias del camino entre los extremos.

En la captura del ejercicio anterior aparecen ambas conexiones:

PCA1 → PCB1

```
[root@localhost ~]# ping6 -c 5 2000:b::b1
PING 2000:b::b1(2000:b::b1) 56 data bytes
64 bytes from 2000:b::b1: icmp_seq=1 ttl=62 time=7.77 ms
64 bytes from 2000:b::b1: icmp_seq=2 ttl=62 time=23.8 ms
64 bytes from 2000:b::b1: icmp_seq=3 ttl=62 time=5.88 ms
64 bytes from 2000:b::b1: icmp_seq=4 ttl=62 time=13.8 ms
64 bytes from 2000:b::b1: icmp_seq=5 ttl=62 time=7.77 ms

--- 2000:b::b1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4049ms
rtt min/avg/max/mdev = 5.888/11.824/23.867/6.588 ms
```

PCB1 → PCA1

```
[root@localhost ~]# ping6 -c 5 2000:a::a1
PING 2000:a::a1(2000:a::a1) 56 data bytes
64 bytes from 2000:a::a1: icmp_seq=1 ttl=62 time=8.76 ms
64 bytes from 2000:a::a1: icmp_seq=2 ttl=62 time=23.4 ms
64 bytes from 2000:a::a1: icmp_seq=3 ttl=62 time=7.43 ms
64 bytes from 2000:a::a1: icmp_seq=4 ttl=62 time=27.6 ms
64 bytes from 2000:a::a1: icmp_seq=5 ttl=62 time=7.48 ms

--- 2000:a::a1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4027ms
rtt min/avg/max/mdev = 7.437/14.958/27.669/8.766 ms
```

09

Pregunta 9

Modificar el MTU del interfaz eth0 de PCB3 a un valor de 1300 y el del interfaz eth1 de PCA3 a un valor de 1350. Observa qué sucede si se realiza un ping6 desde PCA1 hacia PCB1 con un tamaño de 1400 bytes. ¿Quién realiza la fragmentación? ¿Qué tramas se intercambian entre los equipos? ¿Qué diferencia habría si la red fuera totalmente IPv4?

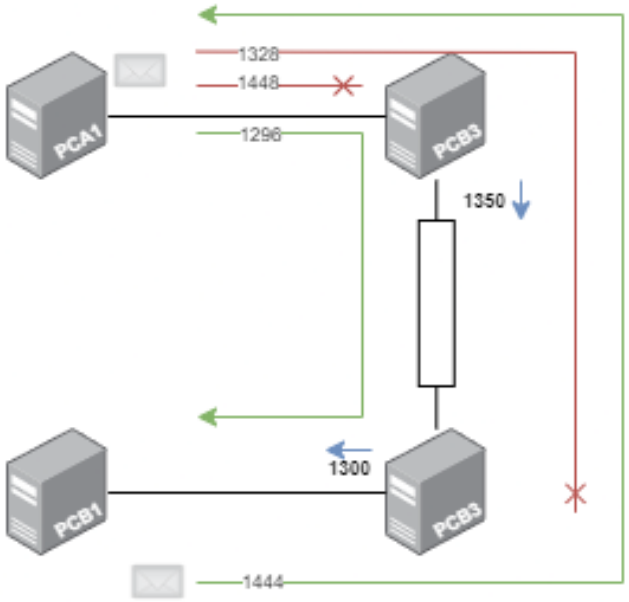

Figura 3: Esquema MTU.

Observando la captura de la red A podemos ver que el primer paquete ICMPv6 echo request (el número 52) es enviado con un tamaño de payload de 1400 bytes, que sumado a los 8 bytes de cabecera ICMPv6 y 40 bytes de cabecera IPv6 da un paquete IPv6 de 1448 bytes de longitud a nivel IP. Cuando PCA3 lo recibe, se da cuenta de que es demasiado grande como para reenviarlo con un MTU de 1330 (descontando cabecera IPv4 del túnel), por lo que interrumpe el envío y responde a PCA1 con un paquete ICMPv6 Packet Too Big. Este paquete es el que aparece inmediatamente después del request, y entre sus campos contiene el MTU, con el fin de que PCA1 fragmente.

PCA1 reintenta el envío (paquetes 54 y 55) con un payload ICMP echo request de 1272 bytes, que sumado a los 48 bytes de antes, más 8 bytes de cabecera IPv6 de fragmentación, da un paquete de 1328 bytes a nivel IP. El resto del payload lo envía separado en otro paquete IPv6, sin cabecera ICMP. Ahora PCA3 es capaz de reenviar el paquete a PCB3, pero PCB3 se encuentra con que es demasiado grande como para transmitirlo por la red B, por lo que lo vuelve a tumbar y en su lugar envia un ICMP Packet Too Big a PCA1. Es decir, es el origen de la comunicación, PCA1, el que debe fragmentar los paquetes, y no los routers.

PCA1 recibe el mensaje de control con el nuevo MTU más bajo, 1300, y vuelve a intentar el envío, esta vez con un payload de 1240. Podemos comprobar que en los dos casos el tamaño del primer fragmento IPv6 es múltiplo de 8, ya que, al igual que en IPv4, el offset es un campo de 13 bits con 2^16 posibles valores. Ahora el mensaje request sí que llega a PCB1. El mensaje reply no tiene problemas de fragmentación porque no hemos limitado el MTU en el sentido contrario.

Si la red operara exclusivamente con IPv4, los routers serían los encargados de fragmentar los paquetes de acuerdo con su propio MTU, siempre que no esté activo el flag *Don't Fragment*, en cuyo caso el paquete no sería retransmitido. Para un caso IPv4 con fragmentación internet los tamaños de los paquetes enviados por cada encaminador serían los siguientes:

PCA1	**PCA3**	**PCB3**
1428 bytes (1400 + 8 + 20)	1344 bytes (1316 + 8 + 20)	1296 bytes (1268 + 8 + 20)
	112 bytes (84 + 8 + 20)	76 bytes (48 + 8 + 20)
		112 bytes (84 + 8 + 20)