

Diseño y administración de redes

Práctica 1.1

Diseño y gestión de escenarios IPv4. Configuración básica.

Informe

Dorian Boleslaw Wozniak - 817570@unizar.es

Marcos Pérez Guillén - 820532@unizar.es

Índice

Índice	2
Cuestión 1	3
Cuestión 2	4
Cuestión 3	6
Cuestión 4	7
Cuestión 5	8
Cuestión 6	9
Cuestión 7	9
Cuestión 8	9
Cuestión 9	10
Cuestión 10	10
Cuestión 11	11
Cuestión 12	12

Cuestión 1

Cuando el escenario esté correctamente configurado, verificarlo en PCA1 y PCA2 mediante las herramientas *tracert* (en la máquina *tinycore*) y *ping -R* (en *Centos*) y comprobándolo mediante las capturas de *Wireshark* correspondientes a los tres saltos de la comunicación. Indica sobre el ejemplo concreto de comunicación desde PCA1 y PCA2 hacia PCB1, qué información proporciona cada una de estas herramientas y relacionala con lo capturado.

La herramienta *ping -R* permite añadir un campo extra en el paquete ICMP donde se registran los saltos realizados. Se puede ver su funcionamiento en las capturas de *Wireshark* para un *ping* entre PCA1 y PCB1:

- Al enviarse un paquete, en la LAN A, se puede observar en el paquete 1 (*echo request* a 192.168.20.1) que contiene una opción *Record Route* donde tiene grabada la IP 192.168.10.1 (PCA1), es decir, la dirección de salida del paquete.
- En el paquete 1 de la captura LAN C (*echo request*) se puede observar que ha registrado un nuevo salto: 192.168.7.10 (PCA3 - interfaz a LAN C), la interfaz por la que se ha reexpedido el paquete.
- Lo mismo se puede observar en el paquete 1 de la LAN B: se registra el salto 192.168.20.3 (PCB3 - LAN B).
- Los *replies* también graban las IPs: en el paquete 2 de la captura LAN B se registra dos veces la dirección 192.168.20.1 (una de cuando llega el *request*, la otra de cuando sale el *reply*).
- Lo mismo pasa en la captura 2 de la LAN C: se añade la dirección 192.168.20.7 (PCB3 - LAN C)
- Finalmente llega el paquete a su origen en la LAN A: se registra las IP 192.168.10.3 (PCA3 - LAN A) y alcanza de nuevo a PCA1, que informa de la ruta tomada.

tracert funciona de forma diferente: en vez de registrar las direcciones de salida del paquete, utiliza el protocolo UDP e ICMP para detectar direcciones “visibles” modificando el valor *Time-to-live* (número de saltos máximo antes de expirar el paquete) e ICMP (para saber si se ha recibido el paquete por el destinatario).

Se puede observar en las capturas un *tracert* de PCA2 a PCB1:

- En los datagramas 13, 15, y 17 de la LAN A, se observa que se envían tres paquetes UDP con un TTL 1. Es decir, no realizará ningún salto. PCA3, en los paquetes 14, 16 y 18, responde a PCA2 con un ICMP *Time-to-live exceeded*, informando que los datagramas han expirado.
- En los datagramas 19, 21, 23, vuelve a intentarlo con un TTL=2. Esta vez se puede observar que llega a la LAN C (datagramas 15, 17 y 19 de captura LAN C) con un TTL reducido (porque ya ha hecho un salto). Se vuelve a recibir un ICMP *Time-to-live exceeded*, pero esta vez contesta PCB3 (LAN A: paquetes 20, 22, 24, LAN C: 16, 18, 20).

- Finalmente, los datagramas 26, 28 y 30 de la LAN A se envían con un TTL=3, suficiente para alcanzar la LAN B. Los datagramas pasan por LAN C con TTL=2 (21, 23, 25) y alcanzan LAN B (15, 17, 19). Por fin alcanzan LAN B con un TTL=1, y PCB1 responde con un ICMP *Destination unreachable* (16, 18, 20), que regresa a PCA2.

Cuestión 2

Mientras se va realizando la configuración del encaminamiento dinámico verifica en las capturas adecuadas, que aparecen los paquetes RIP y cuál es su contenido. También comprueba en los router los valores de las tablas de encaminamiento (telnet localhost ripd >> tiempo de vida de las rutas y métricas) relacionándolo con la aparición de los paquetes RIP. Cuando el escenario esté correctamente configurado, verificarlo en PCA1 (Centos) mediante la herramienta ping -R y explica los resultados. No hace falta captura de wireshark. Después, parar el servicio ripd en uno de los router y comprobar que transcurridos 180 segundos la línea de la tabla de encaminamiento del otro router (telnet localhost ripd >> tiempo de vida de las rutas y métricas) pasa a tener métrica 16 y que transcurridos otros 120 segundos, esta línea se borra. Para comprobarlo, hay que entregar una captura de pantalla con los parámetros de la tabla de encaminamiento.

Se puede observar la comunicación entre los routers para obtener una tabla de encaminamiento mediante RIP en la captura de la LAN C.

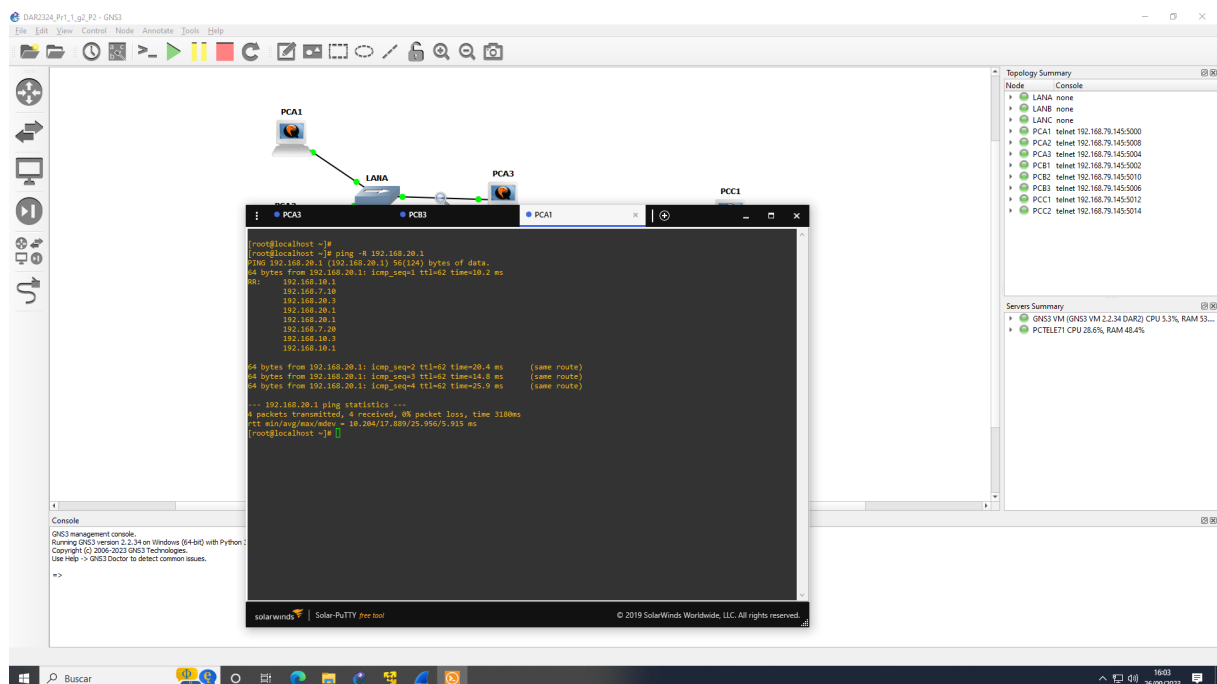


Figura 1: Ejecución de un ping -R donde se muestra que PCA1 alcanza PCB1 en el escenario donde los routers PCA3 y PCB3 utilizan encaminamiento dinámico con RIP.

En el paquete 1, se puede observar una solicitud RIP de PCA3 a una dirección multicast (224.0.0.9) para informar que solicita información de las tablas de rutas de otros equipos. A continuación, en el paquete 3 responde a esta misma solicitud informando que conoce una ruta para la subred 192.168.10.0 con métrica 1 (es decir, está a un salto). PCB3 tienen un comportamiento similar: difunde una solicitud en el paquete 5 y responde que tiene una ruta de métrica 1 a la red 192.168.20.0. El objetivo es que todos los equipos con RIP habilitado reciban sus tablas de rutas.

En las LAN A y B se puede observar como registra el router saltos de una LAN a otra. En el paquete 6 de la LAN A, se puede observar que PCA3 devuelve que en su tabla puede alcanzar la red de LAN B en dos saltos.

Finalmente, se puede comprobar con un ping que hay conectividad entre las LAN A y B una vez establecidas las rutas de encaminamiento. Se puede observar el resultado en la Figura 1.

RIP está configurado para que se actualice periódicamente. Tras desactivar ripd en PCB3, se puede observar que primero contiene una ruta a la red LAN A. Tras tres minutos, la métrica a dicha ruta asciende a 16 (el máximo de saltos que permite RIP). Si pasan 5 minutos desde la pérdida de conexión con la red, se borra la entrada de la tabla.

```
User Access Verification
Password:
localhost:localdomain# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface
Network      Next Hop      Metric From      Tag Time
C(1) 192.168.7.0/24 0.0.0.0      1 self          0
R(n) 192.168.10.0/24 192.168.7.10 2 192.168.7.10  0 02:52
C(1) 192.168.20.0/24 0.0.0.0      1 self          0
localhost:localdomain# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface
Network      Next Hop      Metric From      Tag Time
C(1) 192.168.7.0/24 0.0.0.0      1 self          0
R(n) 192.168.10.0/24 192.168.7.10 2 192.168.7.10  0 02:42
C(1) 192.168.20.0/24 0.0.0.0      1 self          0
localhost:localdomain# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface
Network      Next Hop      Metric From      Tag Time
C(1) 192.168.7.0/24 0.0.0.0      1 self          0
R(n) 192.168.10.0/24 192.168.7.10 16 192.168.7.10  0 01:56
C(1) 192.168.20.0/24 0.0.0.0      1 self          0
localhost:localdomain# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface
Network      Next Hop      Metric From      Tag Time
C(1) 192.168.7.0/24 0.0.0.0      1 self          0
R(n) 192.168.10.0/24 192.168.7.10 16 192.168.7.10  0 00:18
C(1) 192.168.20.0/24 0.0.0.0      1 self          0
localhost:localdomain# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface
Network      Next Hop      Metric From      Tag Time
C(1) 192.168.7.0/24 0.0.0.0      1 self          0
R(n) 192.168.10.0/24 192.168.7.10 16 192.168.7.10  0 00:01
C(1) 192.168.20.0/24 0.0.0.0      1 self          0
localhost:localdomain# show ip rip
Codes: R - RIP, C - connected, S - Static, O - OSPF, B - BGP
Sub-codes:
  (n) - normal, (s) - static, (d) - default, (r) - redistribute,
  (i) - interface
Network      Next Hop      Metric From      Tag Time
C(1) 192.168.7.0/24 0.0.0.0      1 self          0
C(1) 192.168.20.0/24 0.0.0.0      1 self          0
localhost:localdomain#
```

Figura 2: Evolución de la tabla de encaminamiento del router PCB3 tras la detención del servicio ripd.

Cuestión 3

Quando el escenario esté correctamente configurado, verificarlo mediante una conexión a los equipos internos de las redes A y B, utilizando la herramienta ping -R para Linux (Anexo I.2.B) y explica los resultados. No hace falta captura de wireshark. Vamos a aprovechar para comprobar si hay paquetes ICMP redirect en la red y justifica su presencia. Ahora sí será necesaria la captura de wireshark.

Al realizar un ping tras borrar la tabla de encaminamiento de PCC1, y establecer un *gateway* predeterminado al router PCA3, se puede observar que el comportamiento de los pings varía según la red a la que se envía. Al no conocer la ruta a PCB3, todas las solicitudes a la red LAN B se envían a través de PCA3.

```

PCCI

root@localhost:~# service network start
Starting network: [ OK ]
root@localhost:~# ip route
kernel IP routing table

```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Type
192.168.7.0	*	255.255.255.0	U	0	0	0	eth0
192.168.1	*	255.255.0.0	U	1002	0	0	eth1
link-local	*	255.255.0.0	U	1003	0	0	eth1

```

root@localhost:~# ip ping -R 192.168.10.1
PING 192.168.10.1 (192.168.10.1) 56(124) bytes of data.
64 bytes from 192.168.10.1: icmp_seq=1 ttl=63 time=4.08 ms
rtt: 192.168.7.1
192.168.7.10
192.168.20.3
192.168.20.1
192.168.20.1
192.168.7.20
192.168.7.1
64 bytes from 192.168.10.1: icmp_seq=2 ttl=63 time=12.9 ms (same route)
64 bytes from 192.168.10.1: icmp_seq=3 ttl=63 time=49.47 ms (same route)
64 bytes from 192.168.10.1: icmp_seq=4 ttl=63 time=4.4 ms (same route)

--- 192.168.10.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3247ms
rtt min/avg/max/mdev = 4.080/10.992/16.402/3.801 ms
root@localhost:~# ip ping -R 192.168.20.1
PING 192.168.20.1 (192.168.20.1) 56(124) bytes of data.
from 192.168.7.10: icmp_seq=1 Redirect Host(New nexthop: 192.168.7.20)
64 bytes from 192.168.20.1: icmp_seq=1 ttl=63 time=15.1 ms
rtt: 192.168.7.10
192.168.7.10
192.168.20.3
192.168.20.1
192.168.20.1
192.168.7.20
192.168.7.1
From 192.168.7.10: icmp_seq=2 Redirect Host(New nexthop: 192.168.7.20)
64 bytes from 192.168.20.1: icmp_seq=2 ttl=63 time=49.7 ms (same route)
From 192.168.7.10: icmp_seq=3 Redirect Host(New nexthop: 192.168.7.20)
64 bytes from 192.168.20.1: icmp_seq=3 ttl=63 time=0.97 ms (same route)
From 192.168.7.10: icmp_seq=4 Redirect Host(New nexthop: 192.168.7.20)
64 bytes from 192.168.20.1: icmp_seq=4 ttl=63 time=5.18 ms (same route)

--- 192.168.20.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3899ms
rtt min/avg/max/mdev = 5.185/10.895/16.750/1.174 ms
root@localhost:~# ip ping -R 192.168.20.1
PING 192.168.20.1 (192.168.20.1) 56(124) bytes of data.
from 192.168.7.10: icmp_seq=1 Redirect Host(New nexthop: 192.168.7.20)
64 bytes from 192.168.20.1: icmp_seq=1 ttl=63 time=4.57 ms
rtt: 192.168.7.10
192.168.7.10
192.168.20.3
192.168.20.1
192.168.20.1
192.168.7.20
192.168.7.1
From 192.168.7.10: icmp_seq=2 Redirect Host(New nexthop: 192.168.7.20)
64 bytes from 192.168.20.1: icmp_seq=2 ttl=63 time=1.5 ms (same route)
64 bytes from 192.168.20.1: icmp_seq=3 ttl=63 time=10.3 ms (same route)

```

solarwinds | Solar-PuTTY per host

© 2019 SolarWinds Worldwide, LLC. All rights reserved.

16:54

Figura 3: Ejemplo de ping a los equipos PCA1 y PCB1 desde PCC1. Se puede observar que se produce una redirección en el caso de PCB1.

La presencia del *redirect* en el paquete ICMP indica que el router PCA3 ha detectado que hay una forma más eficiente de llegar a la red de B, por lo que envía un mensaje ICMP a PCC1 recomendándole a este que envíe los paquetes a través de PCB3. Por cuestiones de seguridad, no se usa la información de los *redirect* para actualizar rutas, sino que es un mensaje informativo.

En la captura de Wireshark de la LAN C, se puede observar los paquetes ICMP *redirect* en diversas tramas, como por ejemplo en la 21, la cual si entramos para ver la información del protocolo ICMP veremos que no indica que es: Type 5 (Redirect). En este caso es una respuesta al paquete 20, que solicita un envío a LAN B cuando PCA3 sabe que existe una

ruta directa. PCA3 reenvía el mensaje (paquete 22), y al conocer PCB3 cómo alcanzar PCC1, reenvía la respuesta de PCB1 directamente (paquete 25).

Cuestión 4

Cuando el escenario esté correctamente configurado, verificarlo mediante una conexión a los equipos internos de las redes A y B, utilizando la herramienta ping -R para Linux (Anexo I.2.B) y explica los resultados. No hace falta captura de wireshark. Comprueba que no aparecen ahora ICMP redirect y justifica el porqué.

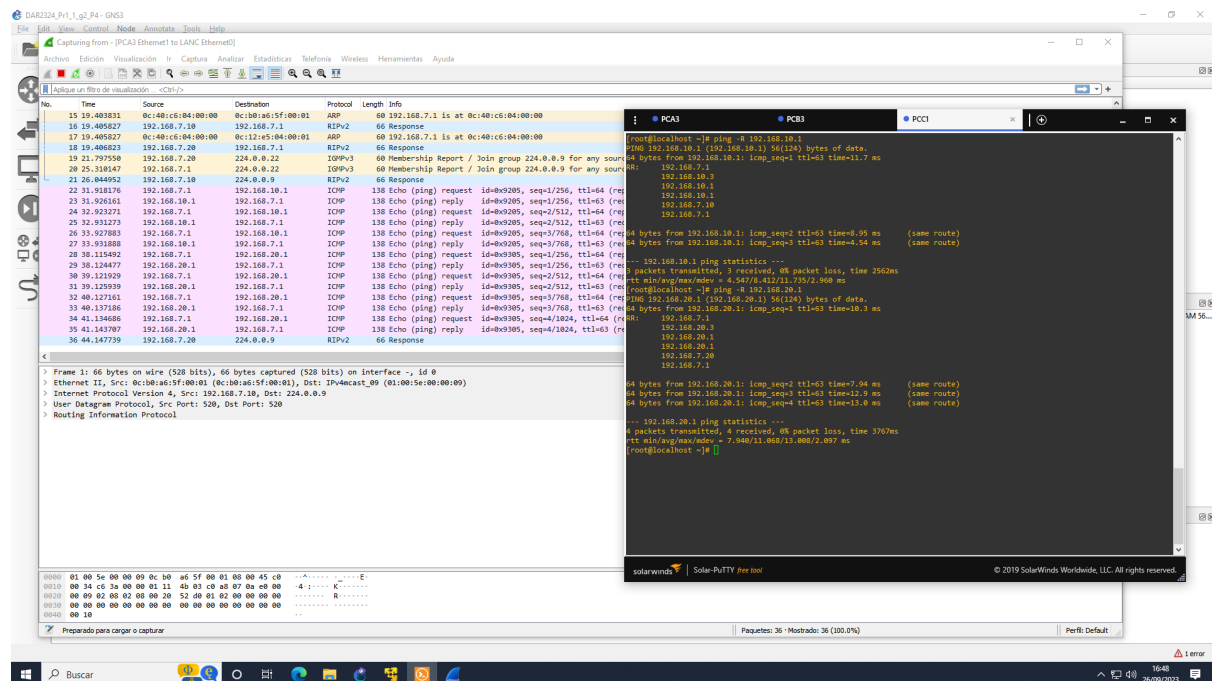


Figura 4: Ejemplo de ping a los equipos PCA1 y PCB1 desde PCC1. Esta vez se aprecia que no se producen redirecciones al conocer PCC1 la ruta más directa a la LAN B.

En este caso ya no se ve el paquete *redirect* ya que al construir dinámicamente las tablas de encaminamiento de PCC1 ya saben todos los caminos posibles para llegar a la red B, por lo que no es necesario que el router PCA3 le indique a PCC1 que hay una forma más eficiente de llegar a la red B al no tener que pasar por PCA3.

Cuestión 5

Indicar los distintos casos de entrega directa e indirecta observados especificando su relación con el tráfico ARP capturado:

Pon un ejemplo de un paquete IP encaminado mediante entrega directa especificando la IP destino y la MAC destino. Pon un ejemplo de un paquete IP encaminado mediante entrega indirecta especificando la IP destino y la MAC destino.

Una entrega directa se produce entre dos máquinas que se encuentran dentro de la misma red. En estos casos, solo se requiere el uso de ARP para obtener la asociación dirección-interfaz.

Esta resolución de dirección se puede apreciar en las capturas de Wireshark (nos vamos a centrar en la de la LAN A). El ejemplo mostrado es parte de un envío de *pings* de PCA1 a PCB1, pero el funcionamiento sería similar en el caso de un *ping* a otro equipo dentro de la LAN A.

En la trama 4 se ve como la maquina 192.168.10.3 trata de averiguar la dirección a nivel de enlace de la máquina 192.168.10.1, para ello envía un paquete ARP con las siguientes características:

- IP origen: 192.168.10.3
- MAC origen: 0c:b0:a6:5f:00:00
- IP destino: 192.168.10.1 (cuya MAC es desconocida)
- MAC destino: ff:ff:ff:ff:ff:ff (difusión)

A continuación se encuentra la trama 5 la respuesta, en la cual 192.168.10.1 responde a la 192.168.10.3 indicándole su MAC:

- IP origen: 192.168.10.1
- MAC origen: 0c:c5:46:fa:00:00
- IP destino: 192.168.10.1
- MAC destino: 0c:b0:a6:5f:00:00

En el caso de una entrega indirecta la máquina destino se encuentra fuera de nuestra red, por lo tanto se hace uso en este caso de la tabla de encaminamiento del router. Esto lo podemos apreciar en las capturas de Wireshark (nos centramos en la LAN C):

Continuando con el ejemplo anterior, al no encontrarse PCB1 dentro de la red no es posible obtener una asociación entre su dirección e interfaz, así que no se trata de informar de su MAC. Al llegar un ICMP *request* de PCA1 a la LAN C, PCA3 necesita interrogar a la red dónde se ubica PCB3 para poder continuar. Se puede observar la solicitud en la trama 5:

- IP origen: 192.168.7.10
- MAC origen: 0c:b0:a6:5f:00:01

- IP destino: 192.168.7.20 (no conoce su MAC)
- MAC destino: ff:ff:ff:ff:ff:ff (difusión)

La respuesta se observa en la trama 6:

- IP origen: 192.168.7.20
- MAC origen: 0c:12:e5:04:00:01
- IP destino: 192.168.7.10
- MAC destino: 0c:b0:a6:5f:00:01

PCB3 reexpide el paquete a LAN B, que debe pasar por un proceso similar.

Cuestión 6

¿A qué se deben las preguntas ARP durante la transmisión del primer ICMP Echo Request?

Se deben a que se ha eliminado anteriormente las asociaciones de direcciones IP-interfaz de las máquinas, por lo que ninguna máquina en la red conoce cómo llegar al resto a nivel de enlace. Para ello, es necesario interrogar a los equipos de su red para obtener su MAC.

Cuestión 7

¿Por qué no hay preguntas ARP durante la transmisión del primer ICMP Echo Reply?

No hay preguntas ARP en el camino de regreso porque al realizar una transacción ARP el destinatario de la consulta graba la MAC de la IP origen, y el emisor inicial la del destino al recibir su respuesta.

Cuestión 8

¿Qué ocurre con los siguientes paquetes ICMP?

Los siguientes *echos* generalmente no producen tráfico ARP al conocerse ya las asociaciones necesarias para los envíos. Ocasionalmente se pueden producir envíos ARP para refrescar la tabla (p.e. en la LAN C se envía otra interrogación por parte de PCB3 en las tramas 19 y 20 para conocer la MAC de PCA3).

Cuestión 9

Sin dejar de capturar, parar el anterior ping e inmediatamente después iniciar un ping desde un host de LAN C > PCA1. ¿Aparecen mensajes ARP en LAN interna y en LAN externa? ¿Por qué?

Los mensajes ARP no se lanzan hasta que no se produzca una comunicación entre dos equipos. En la LAN C, se puede apreciar que se produce un intercambio (en las tramas 30 y 31) entre PCA3 y PCC1 al ser la primera vez que interactúan. Por otro lado, tampoco se produce tráfico ARP en la LAN A porque PCA1 y PCA3 ya se conocen, con un refresco en las tramas 34 y 35 en medio del *ping*.

Cuestión 10

Indicar el número y tamaño de los paquetes capturados en las redes LAN A, LAN C y LAN B. Para ello, rellena la tabla adjunta y justifica teóricamente los tamaños indicados.

	Nº Paquete	Tamaño	Campos de cabecera IP				
			ID	Flags		Offset	
				DF	MF	Original	Wireshark
LAN A	4	1428	0x7b05	0	0	0	0
	5	796	0x7b05	0	1	0	0
	6	524	0x7b05	0	1	97	776
	7	148	0x7b05	0	0	160	1280
LAN B	6	700	0x7b05	0	1	0	0
	7	420	0x7b05	0	1	85	680
	8	348	0x7b05	0	0	135	1080
	9	1428	0x7b05	0	0	0	0
LAN C	5	1100	0x7b05	0	1	0	0
	6	348	0x7b05	0	0	135	1080
	7	1300	0x7b05	0	1	0	0
	8	148	0x7b05	0	0	160	1280

- Al salir el ping de PCA1 (LAN A - 4), llega a PCA3 íntegro, al no tener PCA1 un MTU.
- Al salir de PCA3, se aplica un MTU de 1100. El paquete se subdivide en dos (LAN C - 5 y 6), con el segundo paquete teniendo un offset total de 1080 bytes (los paquetes

se fragmentan en múltiplos de 8). El offset es 1080 en vez de 1100 porque necesita espacio para la cabecera IP (20 bytes), la cual cuenta para el MTU.

- Al salir a PCB3 (LAN B - paquetes 6, 7 y 8), se vuelve a aplicar un MTU, esta vez de 700. El paquete de 1100 (LAN C - 5) tiene que volver a ser dividido, esta vez en dos paquetes de 700 y 420 bytes respectivamente. La segunda mitad cabe y no se altera.
- Al enviar PCB1 una respuesta, llega íntegra (LAN B - 9).
- Al salir de PCB3 se aplica un MTU de 1300. La respuesta (LAN C - 7 y 8) se fragmenta en dos paquetes de 1300 y 148.
- Al salir de PCA3 (LAN A - 5, 6 y 7) se fragmenta el paquete de 1300 al tener PCA3 un MTU de 800 en la interfaz. Se divide en dos paquetes de 796 y 524 (800B datos - 20B IP = 780B, no es múltiplo de 8, así que se tiene que dividir el contenido del ICMP a partir del múltiplo más cercano, que es 776). La segunda parte no es alterada.

Cuestión 11

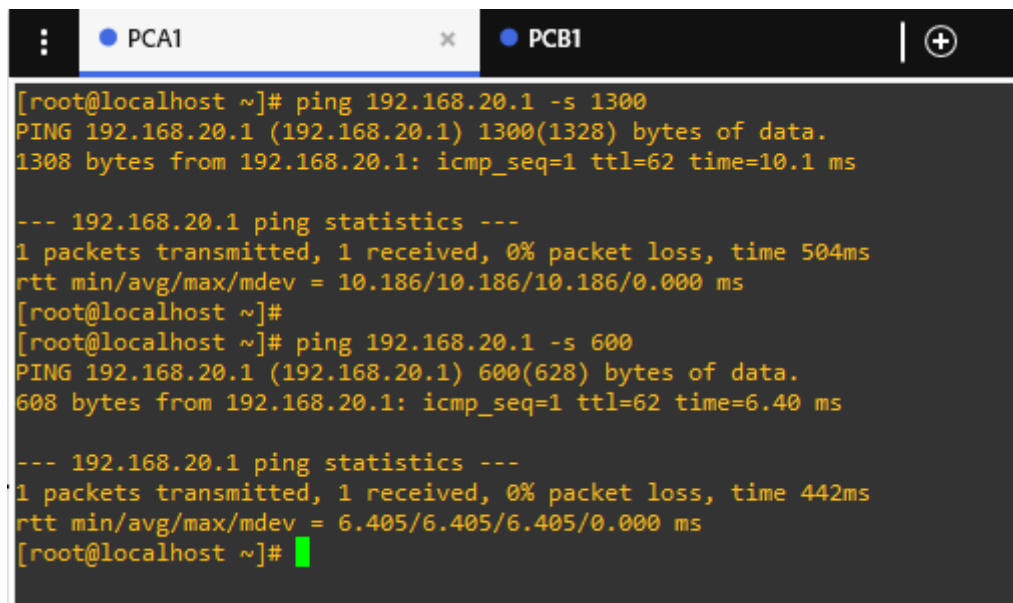
¿Cuántas cabeceras ICMP Echo Request aparecen en LAN A, B y C?

Solo una. La cabecera ICMP no se repite al fragmentar un paquete IP que contiene un ICMP. Lo que si se añade es una cabecera IPv4 a cada fragmento, que cuenta para el MTU. Por tanto, para realizar el cálculo de cómo dividir el contenido de un paquete IPv4 es necesario tener en cuenta el espacio que ocupa la cabecera.

Cuestión 12

Mide el retardo existente entre PCA1 y PCB1, utilizando para ello las dos instrucciones que se muestran a continuación:

```
ping <IP_PC> -s 600  
ping <IP_PC> -s 1300
```



The screenshot shows a terminal window with two tabs: 'PCA1' and 'PCB1'. The terminal output shows the results of two ping commands from PCA1 to 192.168.20.1. The first command is 'ping 192.168.20.1 -s 1300', which returns a time of 10.1 ms. The second command is 'ping 192.168.20.1 -s 600', which returns a time of 6.40 ms. Both commands show 0% packet loss and 1 packet received.

```
[root@localhost ~]# ping 192.168.20.1 -s 1300  
PING 192.168.20.1 (192.168.20.1) 1300(1328) bytes of data.  
1308 bytes from 192.168.20.1: icmp_seq=1 ttl=62 time=10.1 ms  
  
--- 192.168.20.1 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 504ms  
rtt min/avg/max/mdev = 10.186/10.186/10.186/0.000 ms  
[root@localhost ~]#  
[root@localhost ~]# ping 192.168.20.1 -s 600  
PING 192.168.20.1 (192.168.20.1) 600(628) bytes of data.  
608 bytes from 192.168.20.1: icmp_seq=1 ttl=62 time=6.40 ms  
  
--- 192.168.20.1 ping statistics ---  
1 packets transmitted, 1 received, 0% packet loss, time 442ms  
rtt min/avg/max/mdev = 6.405/6.405/6.405/0.000 ms  
[root@localhost ~]#
```

Figura 5: Envíos de pings de diferentes tamaños de PCA1 a PCB1.

El tiempo de ida y vuelta (RTT) de hacer un ping de tamaño 600 a la máquina 192.168.20.1 desde la 192.168.10.1 es de 6.4 ms. Hay que tener en cuenta que se trata de un escenario simulado, por lo que los valores de tiempo son únicamente orientativos.

Usando las capturas de Wireshark se puede observar los retardos de cada paso de la comunicación, usando los timestamps de ICMP (pings del fondo de cada captura):

- PCA1 -> PCA3: 7.537493s
- PCA3 -> PCB3: 7.538493s (+1ms)
- PCB3 -> PCB1: 7.539492 (+1ms)
- PCB1 -> PCB3: 7.540492 (+1ms)
- PCB3 -> PCA3: 7.541491 (+1ms)
- PCA3 -> PCA1: 7.54291 (+1.5ms)

El RTT del ping de 1300 es de 10.1ms, al tener que fragmentar y reenviar más paquetes.

- PCA1 -> PCA3: 7.481825s
- PCA3 -> PCB3: 7.482825s (+1ms)
- PCB3 -> PCB1: 7.484823s (+2ms)

- PCB1 -> PCB3: 7.485843s (+1ms)
- PCB3 -> PCA3: 7.487822s (+2ms)
- PCA3 -> PCA1: 7.489824s (+2ms)