

Después, reiniciar la interfaz de red:

```
/etc/init.d/networking restart
```

Y después reiniciar la vm con un halt.

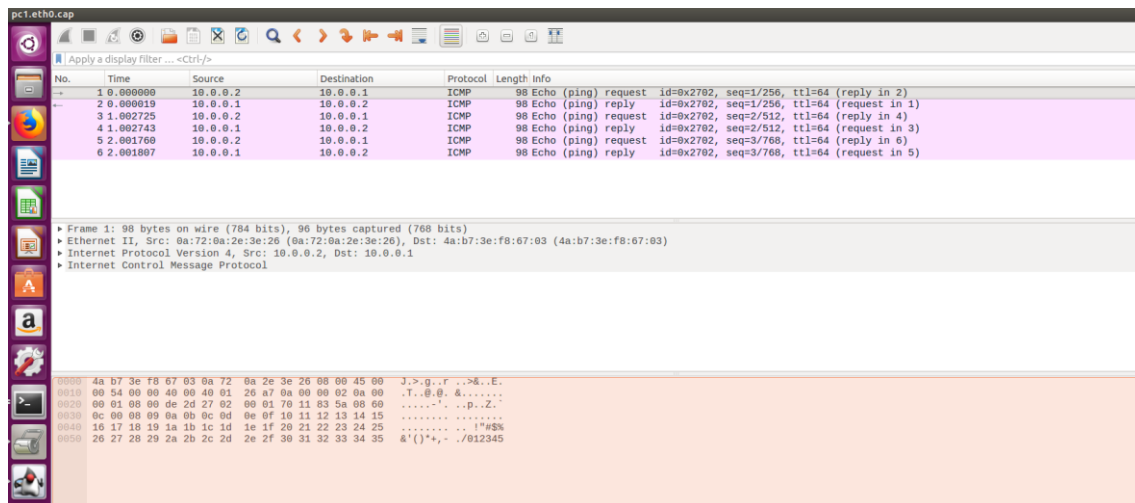
Para hacer el ping pondremos una máquina a escribir los datos que reciba usando:

```
tcpdump -l -w /hosthome/pc1.eth0.cap
```

y la otra enviará 3 paquetes.

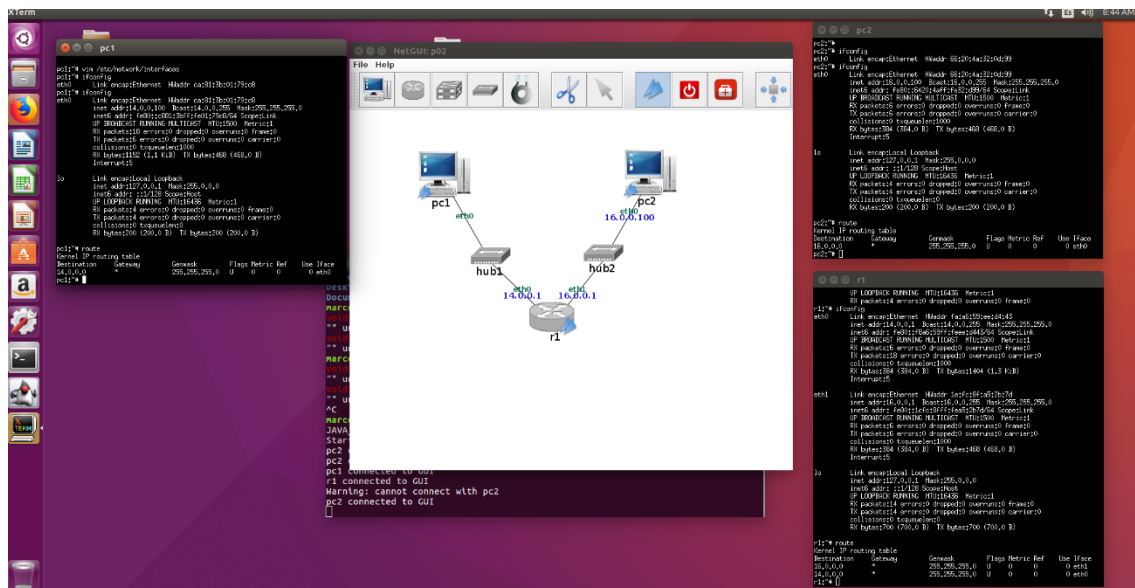
Nota: todo lo que se guarde en hosthome en la vm se guardará en el /home de la máquina real.

Al leer la captura con wireshark nos queda esto:



Ejercicio 2

Tras configurar adecuadamente las redes nos quedaría así (nótese el comando `ifconfig` y `route`) en cada una de las máquinas virtuales:



Para el apartado 10, añado la ruta desde pc1:

```
route add default gw 14.0.0.1
```

Contestando al apartado 10, no funciona ping porque el paquete puede ir pero no volver. Falta crear una ruta en pc2 para que pueda salir el tráfico a r1 y a pc1. Por lo tanto, en pc2:

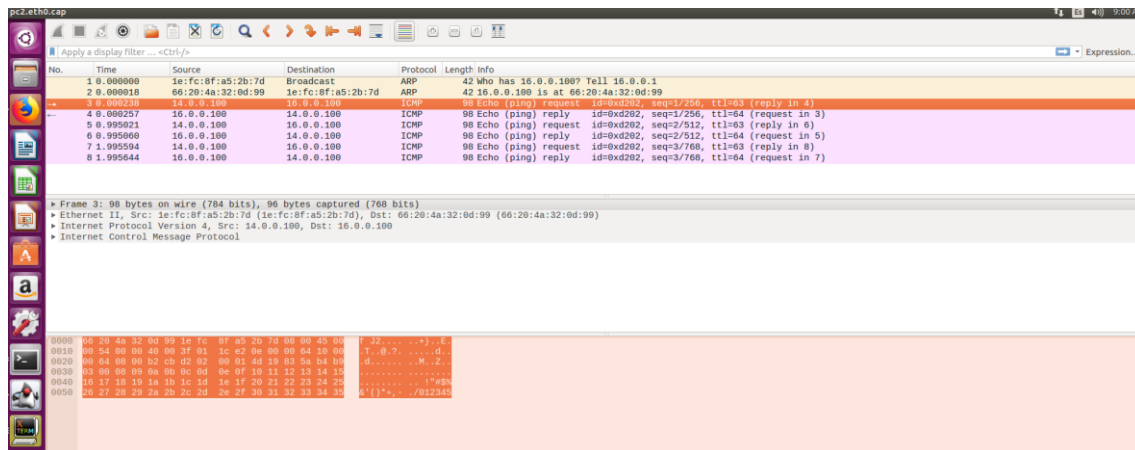
```
route add default gw 16.0.0.1
```

Ahora ya funciona el ping porque los paquetes pueden ir y volver.

Realizo el `tcpdump` en pc2 y envío pingazos desde el pc1.

```
-l -w /hosthome/pc2.eth0.cap
```

El dumpeo del pingazo en wireshark queda así:



¿Con qué TTL llegan los mensajes ICMP a pc2?

63 las peticiones (porque llegan a través de r1) y 64 las replys.

¿Qué valor tienen los campos Type y Code de los mensajes ICMP?

Type: tienen 8 cuando son request y 0 cuando son replys.

Code: siempre es 0. (éxito).

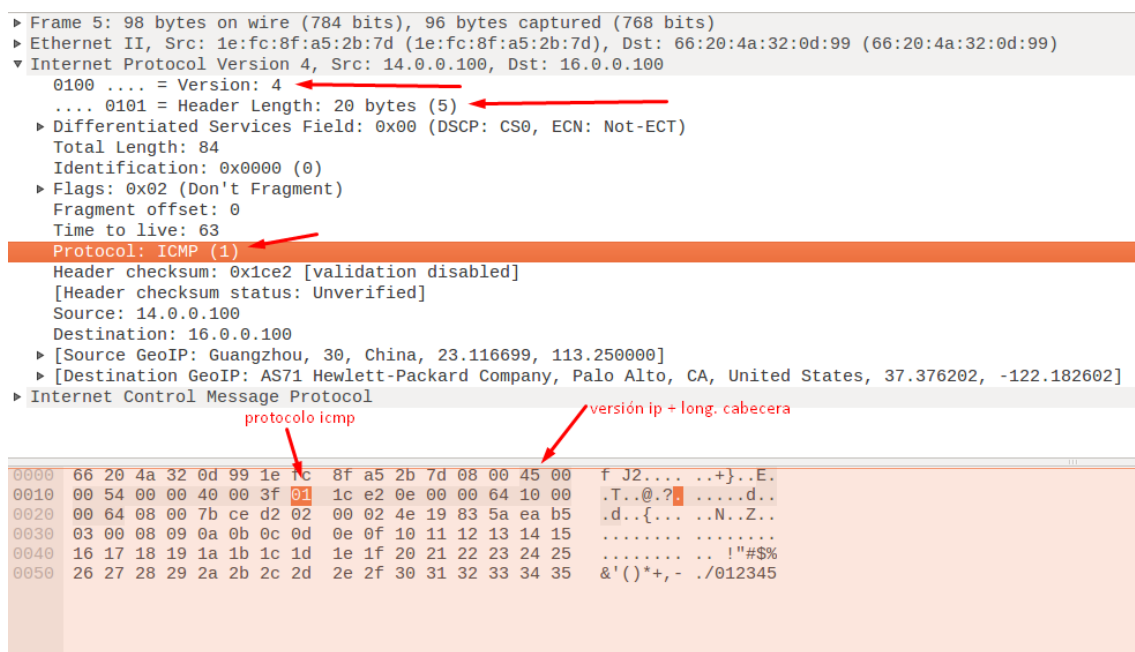
¿Qué valor numérico hexadecimal tiene el campo Protocolo de los datagramas IP en los que viajan los mensajes ICMP?

Valor hex: 45. (En binario es 0100 (Versión 4) + 0101 (longitud de la cabecera))

(Ver imagen en apartado inferior)

Qué valor numérico hexadecimal tiene el campo de tipo de protocolo de las tramas Ethernet en las que viajan los mensajes ICMP?

01 – ICMP

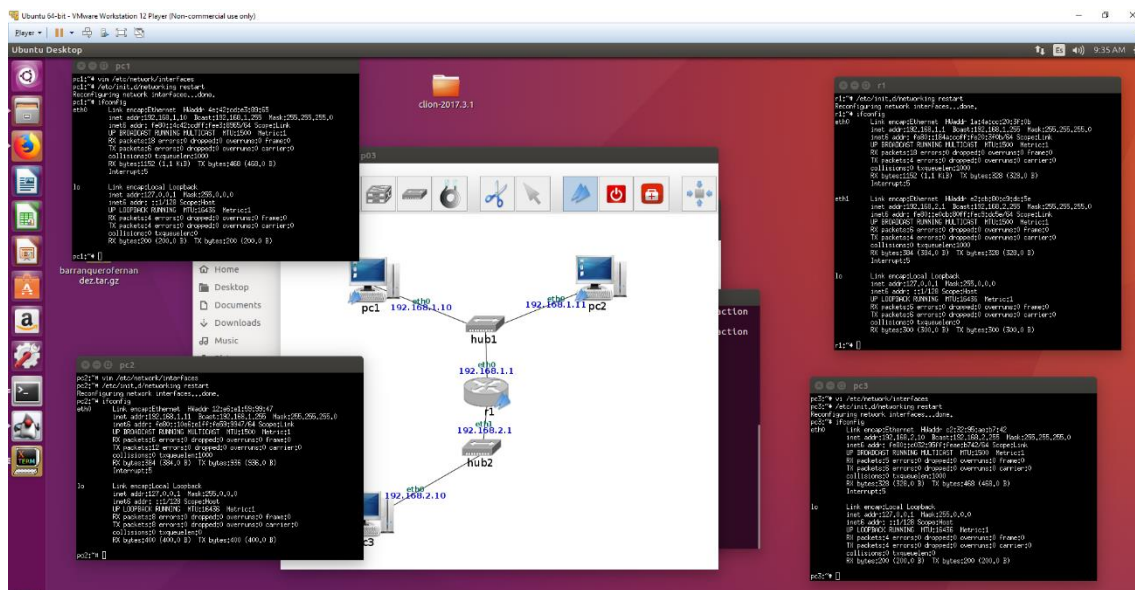


En qué subred/subredes has tenido que capturar el tráfico con tcpdump para responder a las preguntas de este apartado?

Tcpdump se ha ejecutado en pc2, y he enviado ping's desde pc1.

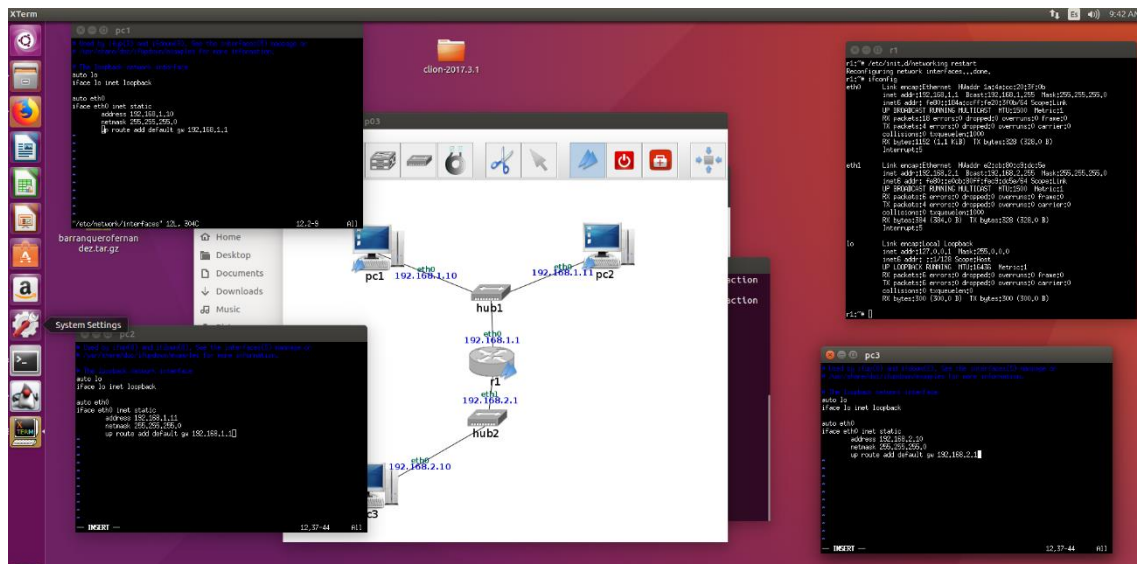
Ejercicio 3

Tras configurar todas las IP's debería de quedar así:

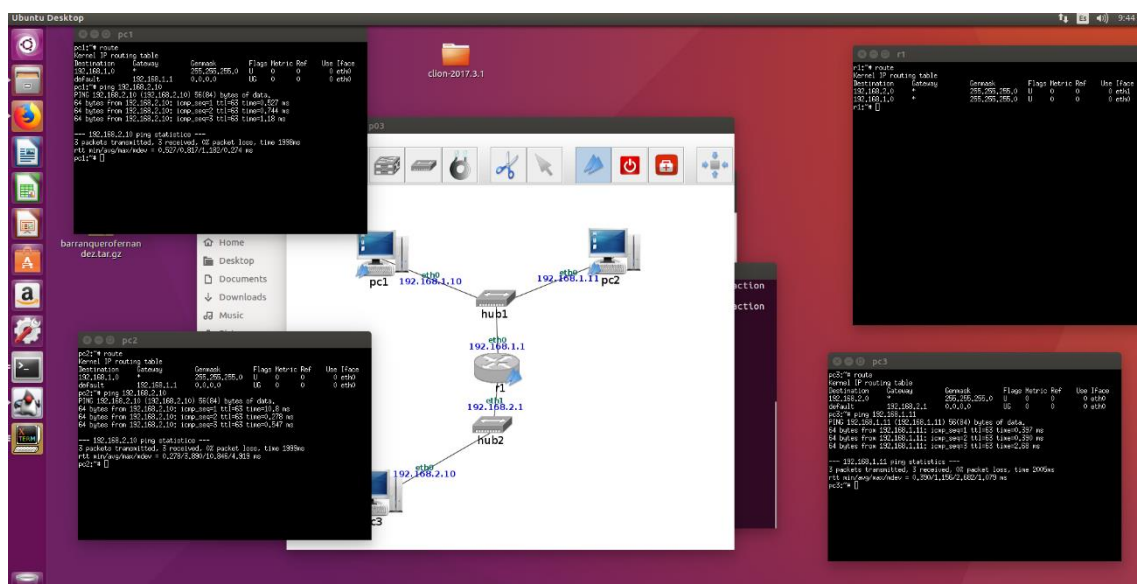


El apartado 5 y 6 funcionan porque se pinguean a sí mismos. En el apartado 7 no porque no está configurada la ruta de paquetes por defecto para direcciones que no sean propias.

Al añadir las rutas por defecto en cada pc el archivo /etc/network/interfaces debería de quedar así:

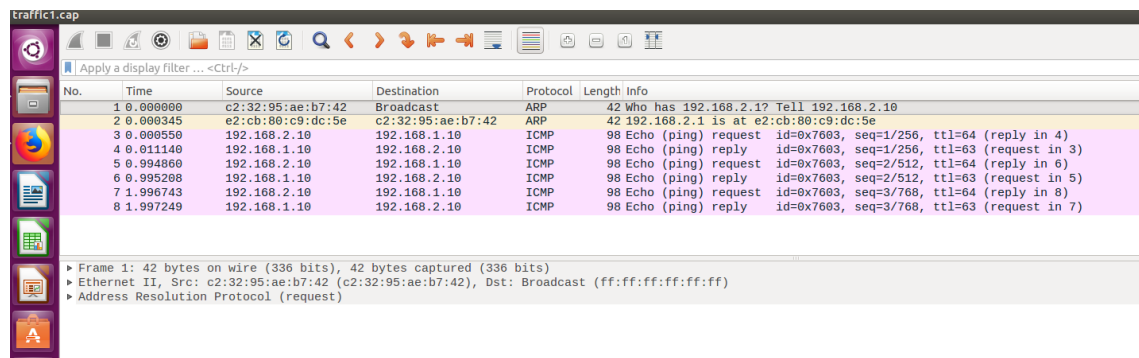


Hacemos el restart y ya funcionan los ping's. Captura de los ping's funcionando y la ruta:



Apartado 13: Traffic1

Son dos mensajes de protocolo ARP, que preguntan a r1 dónde está la IP a la que se hace ping, y se lo diga a pc3. Después, son ping's request y reply's:



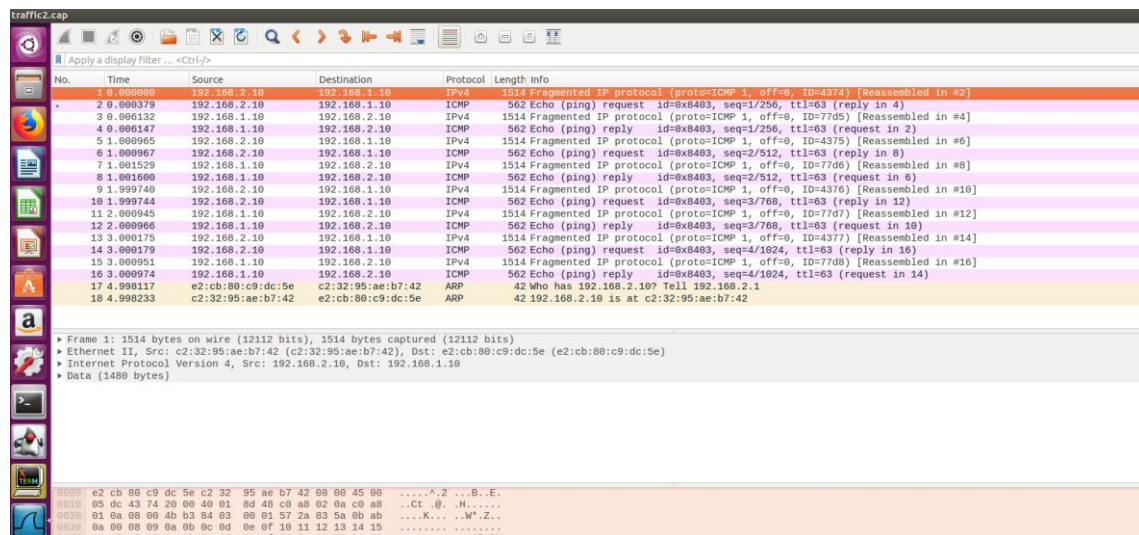
Wireshark capture of traffic1.cap. The capture shows an ARP request from 192.168.2.10 to Broadcast, followed by an ARP reply from 192.168.1.10 to 192.168.2.10. This is followed by a series of ICMP Echo (ping) requests and replies between 192.168.2.10 and 192.168.1.10.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	c2:32:95:ae:b7:42	Broadcast	ARP	42	Who has 192.168.2.1? Tell 192.168.2.10
2	0.000345	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	192.168.2.1 is at e2:cb:80:c9:dc:5e
3	0.000550	192.168.2.10	192.168.1.10	ICMP	98	Echo (ping) request id=0x7603, seq=1/256, ttl=64 (reply in 4)
4	0.011140	192.168.1.10	192.168.2.10	ICMP	98	Echo (ping) reply id=0x7603, seq=1/256, ttl=63 (request in 3)
5	0.994860	192.168.2.10	192.168.1.10	ICMP	98	Echo (ping) request id=0x7603, seq=2/512, ttl=64 (reply in 6)
6	0.995208	192.168.1.10	192.168.2.10	ICMP	98	Echo (ping) reply id=0x7603, seq=2/512, ttl=63 (request in 5)
7	1.996743	192.168.2.10	192.168.1.10	ICMP	98	Echo (ping) request id=0x7603, seq=3/768, ttl=64 (reply in 8)
8	1.997249	192.168.1.10	192.168.2.10	ICMP	98	Echo (ping) reply id=0x7603, seq=3/768, ttl=63 (request in 7)

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
Ethernet II, Src: c2:32:95:ae:b7:42 (c2:32:95:ae:b7:42), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)

Apartado 14: Traffic2

Por cada datagrama request, de long. 562, se capturan 1 datagrama de longitud 1514:



Wireshark capture of traffic2.cap. The capture shows a series of fragmented IP packets (protocol=1, offset=0) from 192.168.2.10 to 192.168.1.10. These are followed by ICMP Echo (ping) requests and replies between 192.168.2.10 and 192.168.1.10. The final frame shows an ARP request from 192.168.2.10 to Broadcast, followed by an ARP reply from 192.168.1.10 to 192.168.2.10.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.10	192.168.1.10	IPv4	1514	Fragmented IP protocol (protocol=1, offset=0) [Reassembled in #2]
2	0.000379	192.168.2.10	192.168.1.10	ICMP	562	Echo (ping) request id=0x8403, seq=1/256, ttl=63 (reply in 4)
3	0.000132	192.168.1.10	192.168.2.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=77d5) [Reassembled in #4]
4	0.000147	192.168.1.10	192.168.2.10	ICMP	562	Echo (ping) reply id=0x8403, seq=1/256, ttl=63 (request in 2)
5	1.000965	192.168.2.10	192.168.1.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=4375) [Reassembled in #6]
6	1.000967	192.168.2.10	192.168.1.10	ICMP	562	Echo (ping) request id=0x8403, seq=2/512, ttl=63 (reply in 8)
7	1.001529	192.168.1.10	192.168.2.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=77d6) [Reassembled in #8]
8	1.001600	192.168.1.10	192.168.2.10	ICMP	562	Echo (ping) reply id=0x8403, seq=2/512, ttl=63 (request in 6)
9	1.999740	192.168.2.10	192.168.1.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=4376) [Reassembled in #10]
10	1.999744	192.168.2.10	192.168.1.10	ICMP	562	Echo (ping) request id=0x8403, seq=3/768, ttl=63 (reply in 12)
11	2.000945	192.168.1.10	192.168.2.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=77d7) [Reassembled in #12]
12	2.000966	192.168.1.10	192.168.2.10	ICMP	562	Echo (ping) reply id=0x8403, seq=3/768, ttl=63 (request in 10)
13	3.000175	192.168.2.10	192.168.1.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=4377) [Reassembled in #14]
14	3.000179	192.168.2.10	192.168.1.10	ICMP	562	Echo (ping) request id=0x8403, seq=4/1024, ttl=63 (reply in 16)
15	3.000951	192.168.1.10	192.168.2.10	IPv4	1514	Fragmented IP protocol (proto=ICMP 1, offset=0, ID=77d8) [Reassembled in #16]
16	3.000974	192.168.1.10	192.168.2.10	ICMP	562	Echo (ping) reply id=0x8403, seq=4/1024, ttl=63 (request in 14)
17	4.998117	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	Who has 192.168.2.10? Tell 192.168.2.1
18	4.998233	c2:32:95:ae:b7:42	e2:cb:80:c9:dc:5e	ARP	42	192.168.2.10 is at c2:32:95:ae:b7:42

Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
Ethernet II, Src: c2:32:95:ae:b7:42 (c2:32:95:ae:b7:42), Dst: e2:cb:80:c9:dc:5e (e2:cb:80:c9:dc:5e)
Internet Protocol Version 4, Src: 192.168.2.10, Dst: 192.168.1.10
Data (1480 bytes)

Apartado 15: Traffic5

Primer datagrama: 1514. (46 bytes de datos).

Segundo datagrama: 562.

La diferencia es que ahora la longitud del request es mayor que la del archivo fragmentado, y antes era al revés.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.10	192.168.1.10	ICMP	1514	Echo (ping) request id=0x9c03, seq=1/256, ttl=64 (reply in 3)
2	0.000059	192.168.2.10	192.168.1.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=4378)
3	0.000532	192.168.1.10	192.168.2.10	ICMP	1514	Echo (ping) reply id=0x9c03, seq=1/256, ttl=63 (request in 1)
4	0.000541	192.168.1.10	192.168.2.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=77d9)
5	0.999096	192.168.2.10	192.168.1.10	ICMP	1514	Echo (ping) request id=0x9c03, seq=2/512, ttl=64 (reply in 7)
6	0.999097	192.168.2.10	192.168.1.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=4379)
7	0.999358	192.168.1.10	192.168.2.10	ICMP	1514	Echo (ping) reply id=0x9c03, seq=2/512, ttl=63 (request in 5)
8	0.999362	192.168.1.10	192.168.2.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=77da)
9	1.998721	192.168.2.10	192.168.1.10	ICMP	1514	Echo (ping) request id=0x9c03, seq=3/768, ttl=64 (reply in 11)
10	1.998725	192.168.2.10	192.168.1.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=437a)
11	1.999355	192.168.1.10	192.168.2.10	ICMP	1514	Echo (ping) reply id=0x9c03, seq=3/768, ttl=63 (request in 9)
12	1.999352	192.168.1.10	192.168.2.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=77db)
13	2.997103	192.168.2.10	192.168.1.10	ICMP	1514	Echo (ping) request id=0x9c03, seq=4/1024, ttl=64 (reply in 15)
14	2.997104	192.168.2.10	192.168.1.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=437b)
15	2.997467	192.168.1.10	192.168.2.10	ICMP	1514	Echo (ping) reply id=0x9c03, seq=4/1024, ttl=63 (request in 13)
16	2.997472	192.168.1.10	192.168.2.10	IPv4	562	Fragmented IP protocol (proto=ICMP 1, off=1480, ID=77dc)
17	4.990352	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	Who has 192.168.2.10? Tell 192.168.2.1
18	4.991357	c2:32:95:ae:b7:42	e2:cb:80:c9:dc:5e	ARP	42	192.168.2.10 is at c2:32:95:ae:b7:42

Identifier (LE):	924 (0x039c)
Sequence number (BE):	1 (0x0001)
Sequence number (LE):	256 (0x0100)
[Response frame: 3]	
Timestamp from icmp data: Feb 13, 2018 10:15:35.905442000 PST	
[Timestamp from icmp data (relative): 0.000338000 seconds]	
▼ Data (46 bytes)	
Data: 08090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f...	
[Length: 46]	

WireShark	08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15	16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25	26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35	36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
0000	e2 cb 80 c9 dc 5e c2 32 95 ae b7 42 08 00 45 00A.Z...B..E.		
0010	05 dc 43 78 20 00 00 01 8d 44 c0 a8 02 0a c0 a8	..Cx..D.....		
0020	08 00 69 8c 9c 03 00 01 47 2b 63 0a e2 006e.Z...		
0030	08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15		
0040	16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25		
0050	26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35		

Apartado 16: Traffic6

Nota:

IP PC3: 192.168.2.10

IP PC1: 192.168.1.10

MAC PC3: c2:32:95:ae:b7:42

MAC PC1: 4e:42:cd:e3:89:65

MAC R1 eth1: e2:cb:80:c9:dc:5e

MAC R1 eth0: 1a:4a:cc:20:3f:0b

Primero, el cliente pregunta al hub dónde está el servidor (ip 192.168.1.10), y éste le responde que se llega a través de eth1. (Paquetes 0 y 1).

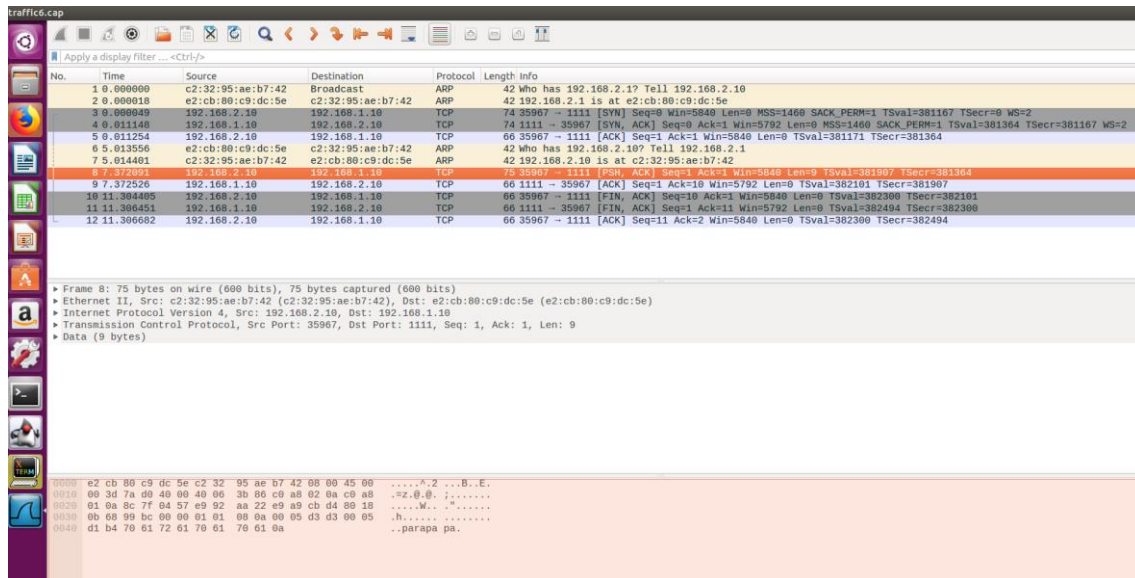
Después, se ha establecido la conexión de la capa de transporte (TCP), paquetes de SYN y ACK 3 y 4.

Tras esto, se ha enviado el ACK del servidor al cliente (paquete 5), y para que llegue, el servidor pregunta al r1 cómo llega a él, y este le responde que a través de eth1. (Paquetes 6 y 7).

Después, el cliente envía el mensaje (parapapa), y el servidor le envía el ack. (Paquetes 8 y 9).

Antes de terminar, se cierra la conexión por protocolo TCP. (paquetes 10 y 11).

Finalmente, el cliente envía el ack al servidor de que ha entendido que cierra la conexión. (paquete 12).



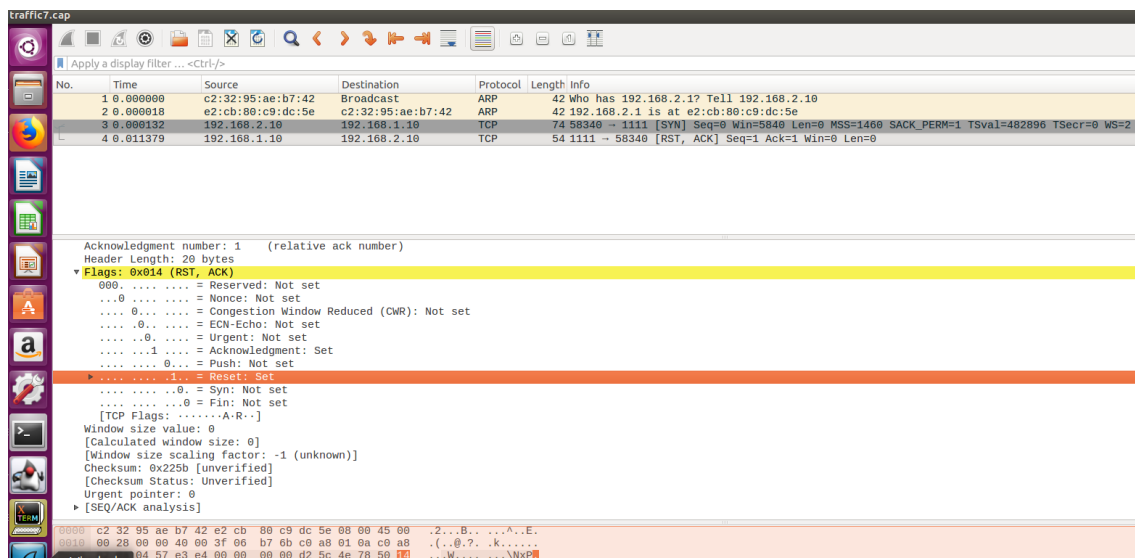
Wireshark capture of traffic6.cap. The packet list shows a successful TCP connection from 192.168.2.10 to 192.168.2.1. The packet details pane shows the selected packet (No. 12) as a TCP Reset (RST) with sequence number 35967 and acknowledgment number 1111. The packet bytes pane shows the raw data of the reset packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	c2:32:95:ae:b7:42	Broadcast	ARP	42	Who has 192.168.2.1? Tell 192.168.2.10
2	0.000018	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	192.168.2.1 is at e2:cb:80:c9:dc:5e
3	0.000049	192.168.2.10	192.168.2.10	TCP	74	35967 → 1111 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=381167 TSecr=0 WS=2
4	0.011140	192.168.2.10	192.168.2.10	TCP	74	1111 → 35967 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=381364 TSecr=381167 WS=2
5	0.011254	192.168.2.10	192.168.2.10	TCP	66	35967 → 1111 [ACK] Seq=1 Ack=1 Win=5840 Len=0 TSval=381171 TSecr=381364
6	5.013556	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	Who has 192.168.2.10? Tell 192.168.2.1
7	5.014401	c2:32:95:ae:b7:42	e2:cb:80:c9:dc:5e	ARP	42	192.168.2.10 is at c2:32:95:ae:b7:42
8	7.372072	192.168.2.10	192.168.2.10	TCP	66	35967 → 1111 [RST, ACK] Seq=0 Ack=1 Win=5840 Len=0 TSval=381197 TSecr=381364
9	7.372526	192.168.2.10	192.168.2.10	TCP	66	1111 → 35967 [ACK] Seq=1 Ack=10 Win=5792 Len=0 TSval=382101 TSecr=381907
10	11.384405	192.168.2.10	192.168.2.10	TCP	66	35967 → 1111 [FIN, ACK] Seq=10 Ack=1 Win=5840 Len=0 TSval=382300 TSecr=382101
11	11.386451	192.168.2.10	192.168.2.10	TCP	66	1111 → 35967 [FIN, ACK] Seq=1 Ack=11 Win=5792 Len=0 TSval=382494 TSecr=382300
12	11.386682	192.168.2.10	192.168.2.10	TCP	66	35967 → 1111 [ACK] Seq=11 Ack=2 Win=5840 Len=0 TSval=382300 TSecr=382494

Apartado 17: Traffic7

Primero, se pregunta al r1 dónde está el servidor. El r1 responde que a través de eth1.

Se intenta establecer conexión por TCP de pc3 a pc1, pero pc1 envía un tcp con flag de reset. Al llegar eso a r1, no se reenvía el mensaje.



Wireshark capture of traffic7.cap. The packet list shows a TCP Reset (RST) packet from 192.168.2.10 to 192.168.2.1. The packet details pane shows the selected packet (No. 4) as a TCP Reset (RST) with sequence number 58340 and acknowledgment number 1111. The packet bytes pane shows the raw data of the reset packet.

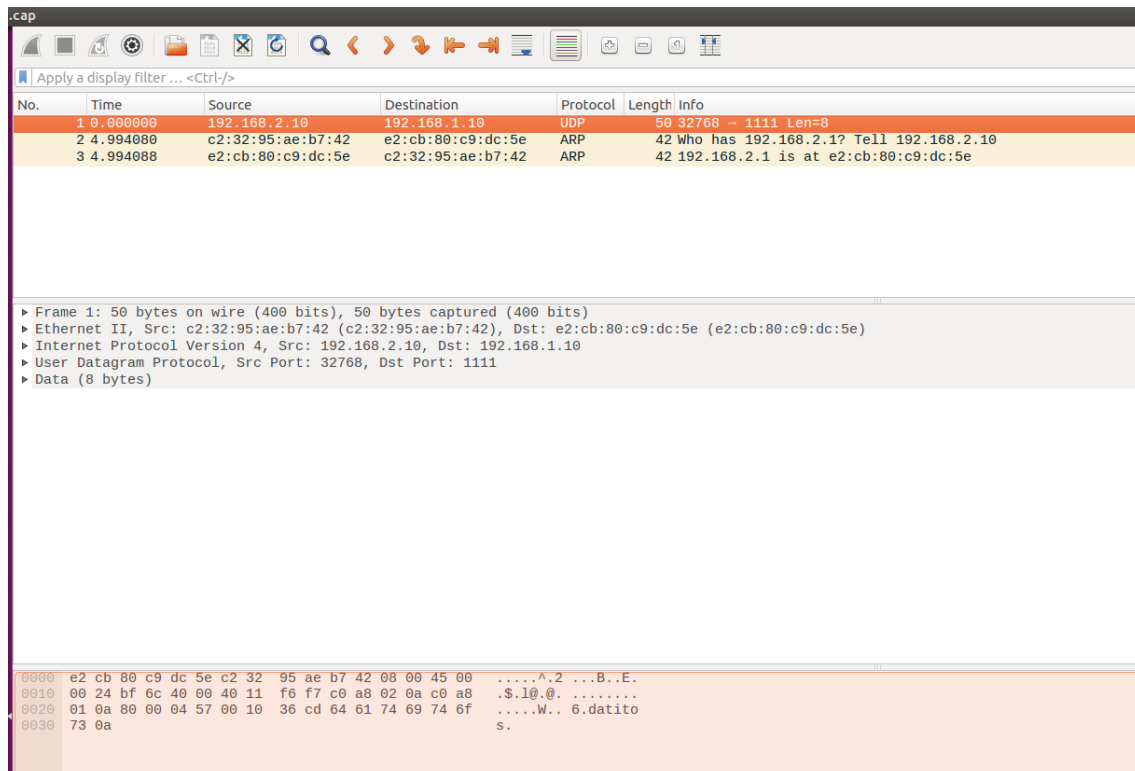
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	c2:32:95:ae:b7:42	Broadcast	ARP	42	Who has 192.168.2.1? Tell 192.168.2.10
2	0.000018	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	192.168.2.1 is at e2:cb:80:c9:dc:5e
4	0.011379	192.168.2.10	192.168.2.10	TCP	54	1111 → 58340 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Apartado 18: traffic8

Nota: NO se ha reseteado la tabla ARP porque NO lo pide el ejercicio. Si se reseteara, creo que harían falta 2 paquetes más de pregunta a r1 sobre cómo hacer llegar el paquete.

Se envían solo 3 paquetes. El primero es directamente el paquete con los datos. Se pregunta cómo se llega (paquete 2) y se contesta (paquete 3).

La diferencia entre TCP y UDP, que se puede observar en esta captura de tráfico, es que TCP se molesta en establecer y cerrar una conexión segura, mientras que UDP envía los datos y no se asegura de que lleguen bien o de establecer previamente un acuerdo de conexión. Es por ello que en la captura de TCP hay varios datagramas estableciendo y cerrando conexión, y en este sólo hay un datagrama UDP con datos.



cap

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.2.10	192.168.1.10	UDP	50	32768 → 1111 Len=8
2	4.994080	c2:32:95:ae:b7:42	e2:cb:80:c9:dc:5e	ARP	42	Who has 192.168.2.1? Tell 192.168.2.10
3	4.994088	e2:cb:80:c9:dc:5e	c2:32:95:ae:b7:42	ARP	42	192.168.2.1 is at e2:cb:80:c9:dc:5e

▶ Frame 1: 50 bytes on wire (400 bits), 50 bytes captured (400 bits)
▶ Ethernet II, Src: c2:32:95:ae:b7:42 (c2:32:95:ae:b7:42), Dst: e2:cb:80:c9:dc:5e (e2:cb:80:c9:dc:5e)
▶ Internet Protocol Version 4, Src: 192.168.2.10, Dst: 192.168.1.10
▶ User Datagram Protocol, Src Port: 32768, Dst Port: 1111
▶ Data (8 bytes)

```
0000 e2 cb 80 c9 dc 5e c2 32 95 ae b7 42 08 00 45 00  ....^..2 ...B..E.
0010 00 24 bf 6c 40 00 40 11 f6 f7 c0 a8 02 0a c0 a8  ..$.l@.@. ....
0020 01 0a 80 00 04 57 00 10 36 cd 64 61 74 69 74 6f  ....W...6.datito
0030 73 0a                                     s.
```