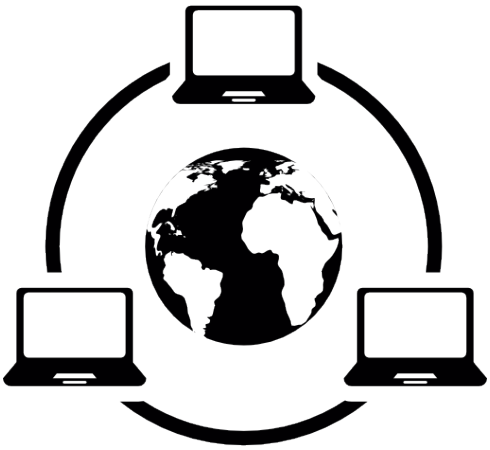
Universidad Zaragoza
1542

Diseño y Administración de Redes

Práctica 1.2 - Diseño y gestión de escenarios IPv4 NAT



Alvaro Orellana Serrano

Germán Toyos Doello

2021

Cuestión 1

Indica cómo debe configurarse el NAT en los router PCA3 y PCB3. El NAT que acabas de configurar, ¿es estático o dinámico?

Para configurar el NAT en el PCA3 deberemos de ejecutar el comando:

```
iptables -t nat -A POSTROUTING -o eth2 -j SNAT --to 192.168.7.1
```

Este comando permitirá que en el POSTROUTING del Router PCA3 todos los paquetes que entren por esa interfaz, tengan como dirección origen 192.168.7.1 (cambia la dirección fuente del paquete al valor 192.168.7.1, antes de enviarlo por la interfaz de salida eth2.)

Para configurar el NAT en el PCB3 deberemos de ejecutar el comando:

```
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 192.168.7.2
```

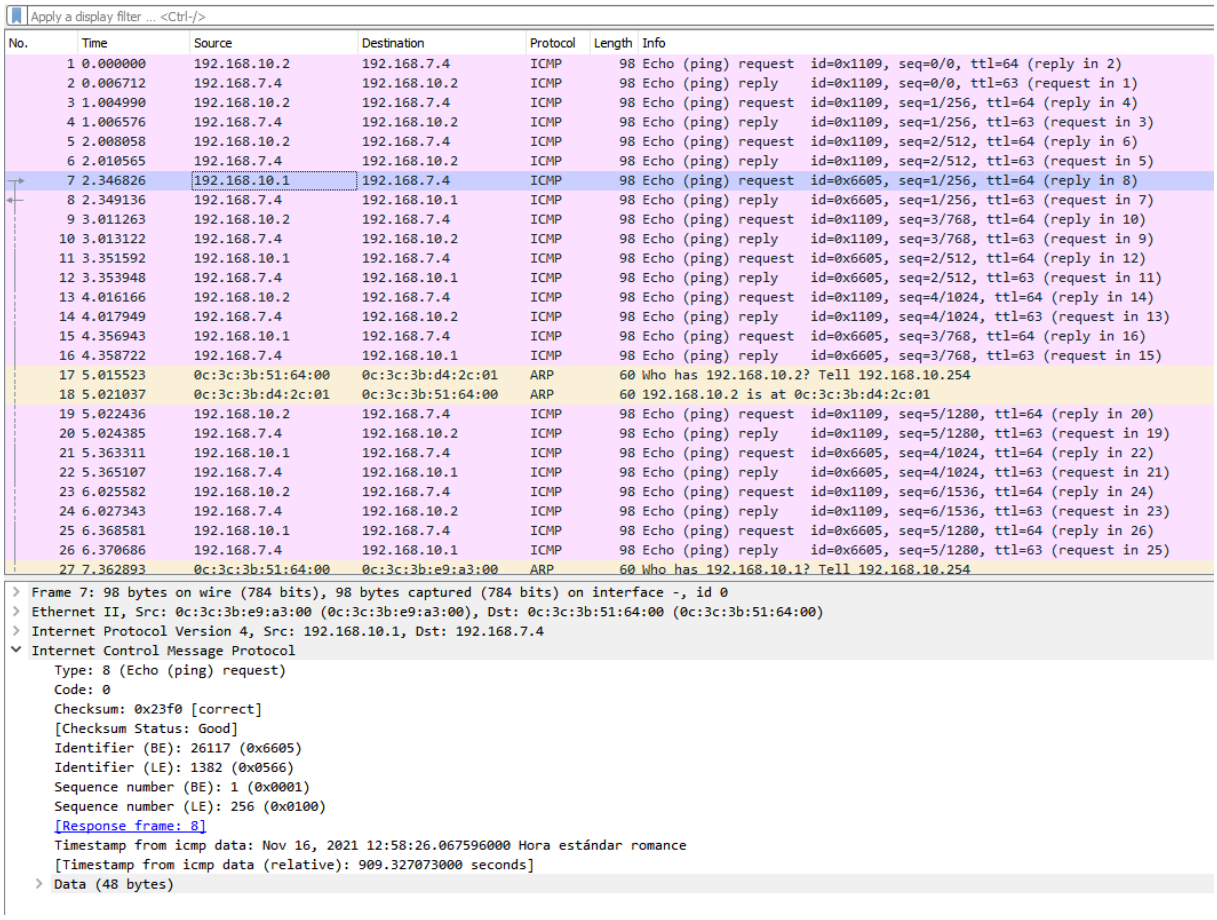
Este comando permitirá que en el POSTROUTING del Router PCB3 todos los paquetes que entren por esa interfaz, tengan como dirección origen 192.168.7.2

El NAT que acabamos de configurar es dinámico, ya que estamos configurando un NAT para que una red privada tenga acceso a la red pública, es decir, desde la red privada podemos acceder a todos los nodos de la red pública. Sin embargo, si quisiéramos entrar desde la red pública a algún nodo de la red privada, deberíamos de configurar uno a uno esos nodos, clasificandolos según el puerto por el que nos interese entrar. Este caso de configuración manual uno a uno sería un buen ejemplo de configuracion estática.

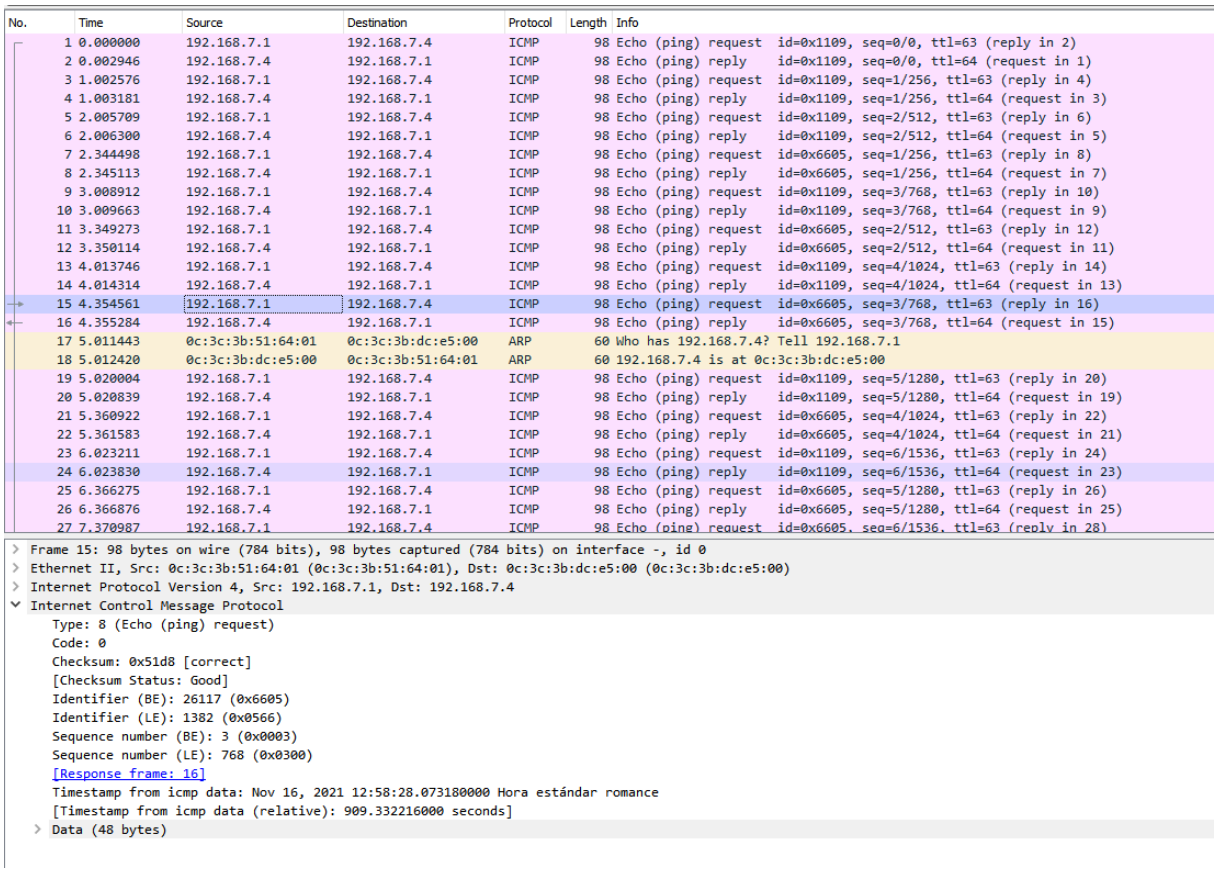
Cuestión 2

Una vez configurado el escenario, verifica su funcionamiento en los casos a) y b) descritos a continuación. Captura en las interfaces oportunas e identifica el uso de direccionamiento a la entrada y salida del router NAT y los datos de las cabeceras ICMP y TCP implicados en el NAT.

Caso a) Paquetes ICMP



En esta captura de Wireshark de la LAN A se puede observar como el ping es lanzado tanto desde la dirección 192.168.10.1 como desde la dirección 192.168.10.2 . Tomaremos nota del campo id, este campo lo podemos observar en el apartado de Internet Control Message Protocol (ICMP) que se denomina Identifier (BE) y que está tanto en numero hexadecimal como en decimal, nos fijaremos en esa parte decimal.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=0/0, ttl=63 (reply in 2)
2	0.002946	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=0/0, ttl=64 (request in 1)
3	1.002576	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=1/256, ttl=63 (reply in 4)
4	1.003181	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=1/256, ttl=64 (request in 3)
5	2.005709	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=2/512, ttl=63 (reply in 6)
6	2.006300	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=2/512, ttl=64 (request in 5)
7	2.344498	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x6605, seq=1/256, ttl=63 (reply in 8)
8	2.345113	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x6605, seq=1/256, ttl=64 (request in 7)
9	3.008912	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=3/768, ttl=63 (reply in 10)
10	3.009663	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=3/768, ttl=64 (request in 9)
11	3.349273	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x6605, seq=2/512, ttl=63 (reply in 12)
12	3.350114	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x6605, seq=2/512, ttl=64 (request in 11)
13	4.013746	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=4/1024, ttl=63 (reply in 14)
14	4.014314	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=4/1024, ttl=64 (request in 13)
15	4.354561	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x6605, seq=3/768, ttl=63 (reply in 16)
16	4.355284	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x6605, seq=3/768, ttl=64 (request in 15)
17	5.011443	0c:3c:3b:51:64:01	0c:3c:3b:dc:e5:00	ARP	60	Who has 192.168.7.4? Tell 192.168.7.1
18	5.012420	0c:3c:3b:dc:e5:00	0c:3c:3b:51:64:01	ARP	60	192.168.7.4 is at 0c:3c:3b:dc:e5:00
19	5.020004	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=5/1280, ttl=63 (reply in 20)
20	5.020839	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=5/1280, ttl=64 (request in 19)
21	5.360922	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x6605, seq=4/1024, ttl=63 (reply in 22)
22	5.361583	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x6605, seq=4/1024, ttl=64 (request in 21)
23	6.023211	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x1109, seq=6/1536, ttl=63 (reply in 24)
24	6.023830	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x1109, seq=6/1536, ttl=64 (request in 23)
25	6.366275	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x6605, seq=5/1280, ttl=63 (reply in 26)
26	6.366876	192.168.7.4	192.168.7.1	ICMP	98	Echo (ping) reply id=0x6605, seq=5/1280, ttl=64 (request in 25)
27	7.370987	192.168.7.1	192.168.7.4	ICMP	98	Echo (ping) request id=0x6605, seq=6/1536, ttl=63 (reply in 28)

```
> Frame 15: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface -, id 0
> Ethernet II, Src: 0c:3c:3b:51:64:01 (0c:3c:3b:51:64:01), Dst: 0c:3c:3b:dc:e5:00 (0c:3c:3b:dc:e5:00)
> Internet Protocol Version 4, Src: 192.168.7.1, Dst: 192.168.7.4
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x51d8 [correct]
    [Checksum Status: Good]
    Identifier (BE): 26117 (0x6605)
    Identifier (LE): 1382 (0x0566)
    Sequence number (BE): 3 (0x0003)
    Sequence number (LE): 768 (0x0300)
    [Response frame: 16]
    Timestamp from icmp data: Nov 16, 2021 12:58:28.073180000 Hora estándar romance
    [Timestamp from icmp data (relative): 909.332216000 seconds]
  > Data (48 bytes)
```

En esta captura de Wireshark de la LAN C se puede observar como el ping es lanzado desde la dirección 192.168.7.1 hasta la dirección 192.168.7.4 . Tomaremos nota del campo id, este campo lo podemos observar en el apartado de Internet Control Message Protocol (ICMP) que se denomina Identifier (BE) y que está tanto en numero hexadecimal como en decimal, nos fijaremos en esa parte decimal.

Una vez hemos tomado nota de los campos id de los paquetes procedentes del PCA1 y del PCA2, comprobaremos el fichero ***mf_conntrack*** que verifica el status del NAT. En este fichero podremos observar cómo se ha clasificado a las diferentes comunicaciones según un id.

Cabe destacar que el mismo procedimiento de investigación que se ha seguido mostrando la información ICMP en wireshark con el PCA1, es el mismo con el PCA2, que comprobrueba que coincide el id de la captura wireshark, con el id que nos está mostrando el fichero ***mf_conntrack***.

```
[root@localhost ~]# cat /proc/net/nf_conntrack
ipv4 2 icmp 1 23 src=192.168.10.1 dst=192.168.7.4 type=8 code=0
id=26117 src=192.168.7.4 dst=192.168.7.1 type=0 code=0 id=26117 mark=0
secmark=0 use=2

ipv4 2 icmp 1 22 src=192.168.10.2 dst=192.168.7.4 type=8 code=0
id=4361 src=192.168.7.4 dst=192.168.7.1 type=0 code=0 id=4361 mark=0
secmark=0 use=2
```

Caso b) Paquetes TCP

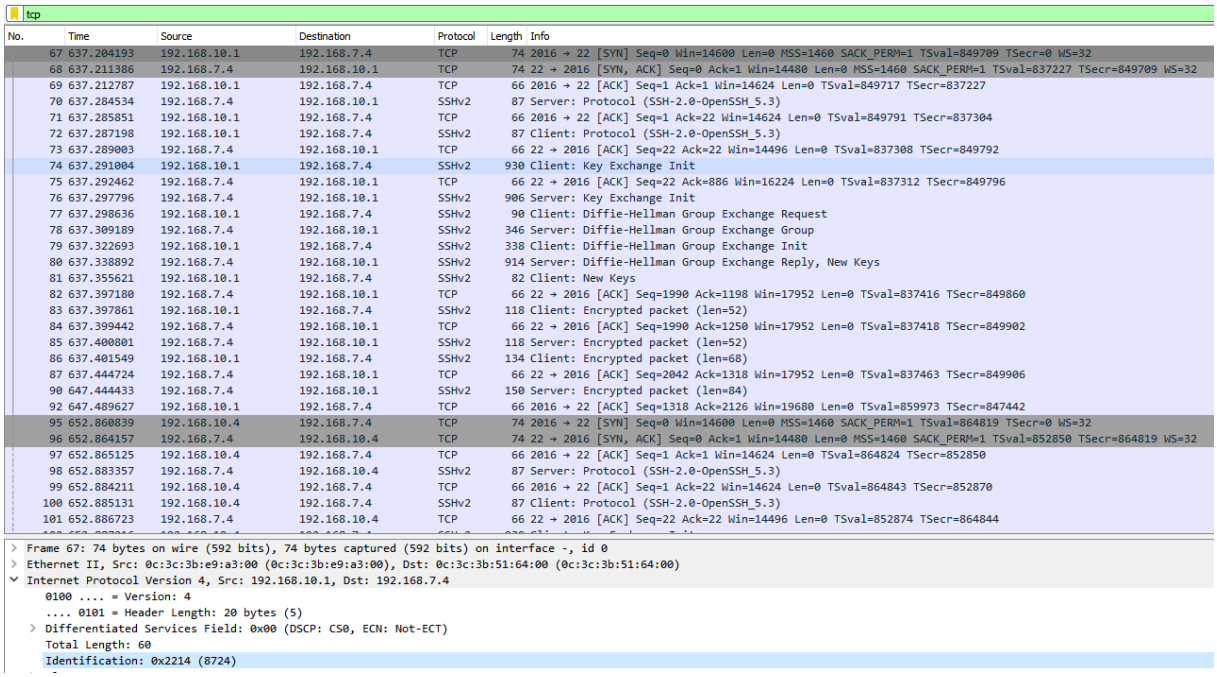
```
[root@localhost ~]#  cat /proc/net/nf_conntrack
ipv4 2 tcp 6 431997 ESTABLISHED src=192.168.10.4 dst=192.168.7.4
sport=2016 dport=22 src=192.168.7.4 dst=192.168.7.1 sport=22 dport=1024
[ASSURED] mark=0 secmark=0 use=2

ipv4 2 tcp 6 431977 ESTABLISHED src=192.168.10.1 dst=192.168.7.4
sport=2016 dport=22 src=192.168.7.4 dst=192.168.7.1 sport=22 dport=2016
[ASSURED] mark=0 secmark=0 use=2
```

En este caso en el que hemos puesto el mismo puerto de destino para las dos máquinas (PCA1 y PCA2) que realizan la conexión ssh a PCC2, si nos fijamos en el fichero **nf_conntrack**, en los paquetes que van hacia el PCC2, ambos puertos de origen son el puerto establecido 2016. Sin embargo, cuando los paquetes vuelven de PCC2, se han cambiado los puertos destino y se han puesto dos diferentes (2016 y 1024). Se ha realizado esta acción para garantizar que se pueda establecer la comunicación correctamente.

A continuación veremos unas capturas sobre ese tráfico TCP generado a causa de realizar el la conexión ssh. De nuevo, se puede observar cómo podemos identificar los paquetes en ambas interfaces de la red gracias a los identificadores de estos. Es curioso cómo en una conexión ssh los paquetes que se envian empiezan por un identificador y ese identificador se va aumentando de 1 en 1 a medida que se envían paquetes hacia el destino. De igual manera, se dedica un identificador base a los paquetes que se reciben y se incrementa de 1 en 1 por cada paquete que se recibe.

LAN A

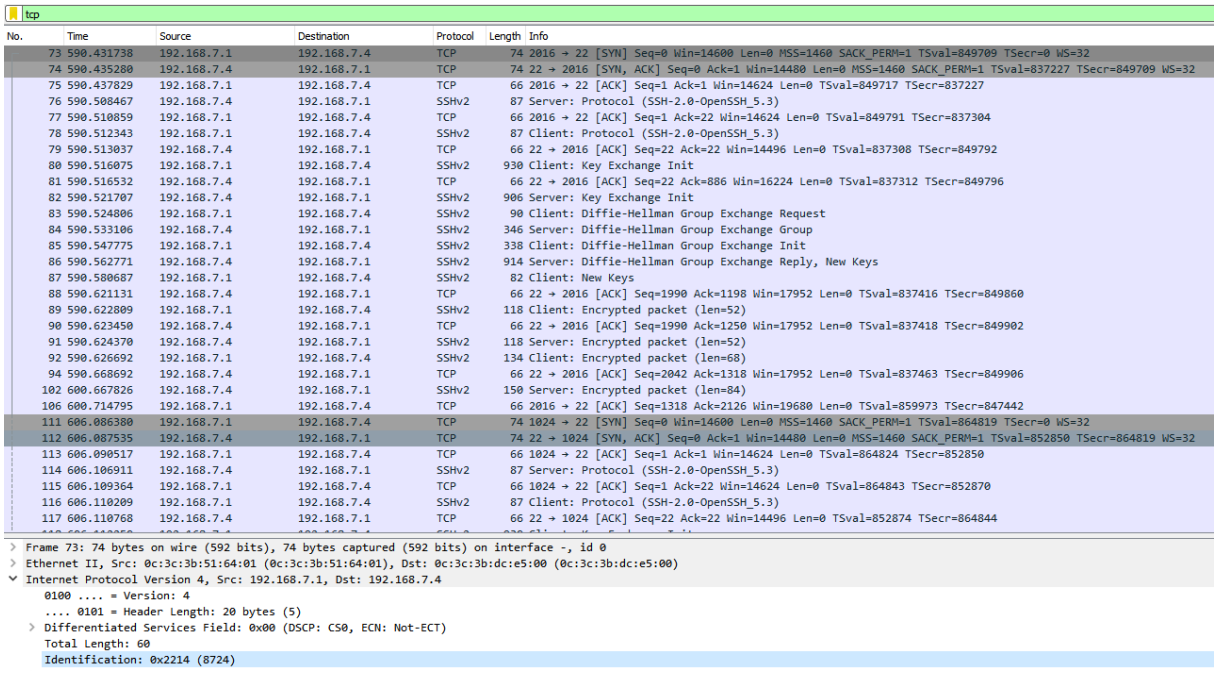


tcp

No.	Time	Source	Destination	Protocol	Length	Info
67	637.204193	192.168.10.1	192.168.7.4	TCP	74	2016 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=849709 TSecr=0 WS=32
68	637.211386	192.168.7.4	192.168.10.1	TCP	74	22 → 2016 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=837227 TSecr=849709 WS=32
69	637.212787	192.168.10.1	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=849717 TSecr=837227
70	637.284534	192.168.7.4	192.168.10.1	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
71	637.285851	192.168.10.1	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=849791 TSecr=837304
72	637.287198	192.168.10.1	192.168.7.4	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
73	637.289003	192.168.7.4	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=837308 TSecr=849792
74	637.291004	192.168.10.1	192.168.7.4	SSHv2	930	Client: Key Exchange Init
75	637.292462	192.168.7.4	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=22 Ack=886 Win=16224 Len=0 TSval=837312 TSecr=849796
76	637.297796	192.168.7.4	192.168.10.1	SSHv2	906	Server: Key Exchange Init
77	637.298636	192.168.10.1	192.168.7.4	SSHv2	90	Client: Diffie-Hellman Group Exchange Request
78	637.309189	192.168.7.4	192.168.10.1	SSHv2	346	Server: Diffie-Hellman Group Exchange Group
79	637.322693	192.168.10.1	192.168.7.4	SSHv2	338	Client: Diffie-Hellman Group Exchange Init
80	637.338892	192.168.7.4	192.168.10.1	SSHv2	914	Server: Diffie-Hellman Group Exchange Reply, New Keys
81	637.355621	192.168.10.1	192.168.7.4	SSHv2	82	Client: New Keys
82	637.397180	192.168.7.4	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=1990 Ack=1198 Win=17952 Len=0 TSval=837416 TSecr=849860
83	637.397861	192.168.10.1	192.168.7.4	SSHv2	118	Client: Encrypted packet (len=52)
84	637.399442	192.168.7.4	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=1990 Ack=1250 Win=17952 Len=0 TSval=837418 TSecr=849902
85	637.400801	192.168.7.4	192.168.10.1	SSHv2	118	Server: Encrypted packet (len=52)
86	637.401549	192.168.10.1	192.168.7.4	SSHv2	134	Client: Encrypted packet (len=68)
87	637.444724	192.168.7.4	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=2042 Ack=1318 Win=17952 Len=0 TSval=837463 TSecr=849906
90	647.444433	192.168.7.4	192.168.10.1	SSHv2	150	Server: Encrypted packet (len=84)
92	647.489627	192.168.10.1	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1318 Ack=2126 Win=19680 Len=0 TSval=859973 TSecr=847442
95	652.860839	192.168.10.4	192.168.7.4	TCP	74	2016 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=864819 TSecr=0 WS=32
96	652.864157	192.168.7.4	192.168.10.4	TCP	74	22 → 2016 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=852850 TSecr=864819 WS=32
97	652.865125	192.168.10.4	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=864824 TSecr=852850
98	652.883357	192.168.7.4	192.168.10.4	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
99	652.884211	192.168.10.4	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=864843 TSecr=852870
100	652.885131	192.168.10.4	192.168.7.4	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
101	652.886723	192.168.7.4	192.168.10.4	TCP	66	22 → 2016 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=852874 TSecr=864844

```
> Frame 67: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface -, id 0
> Ethernet II, Src: 0c:3c:3b:e9:a3:00 (0c:3c:3b:e9:a3:00), Dst: 0c:3c:3b:51:64:00 (0c:3c:3b:51:64:00)
v Internet Protocol Version 4, Src: 192.168.10.1, Dst: 192.168.7.4
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x2214 (8724)
```

LAN C



tcp

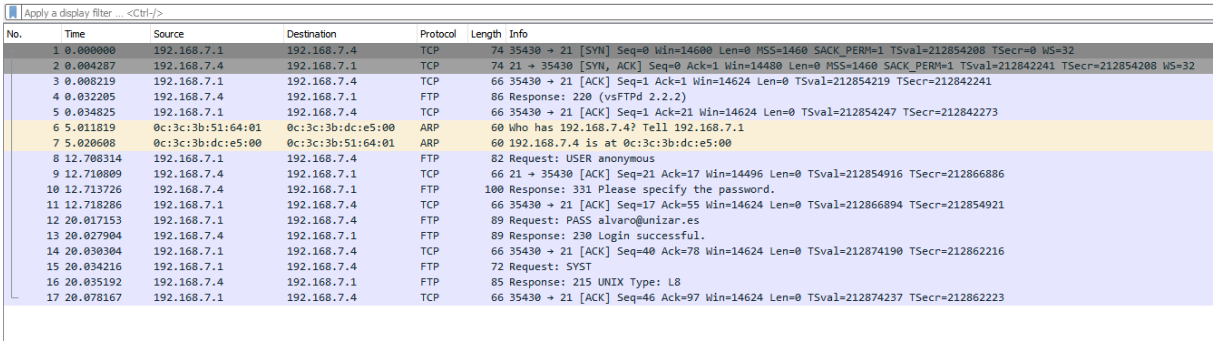
No.	Time	Source	Destination	Protocol	Length	Info
73	590.431738	192.168.7.1	192.168.7.4	TCP	74	2016 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=849709 TSecr=0 WS=32
74	590.435280	192.168.7.4	192.168.7.1	TCP	74	22 → 2016 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=837227 TSecr=849709 WS=32
75	590.437829	192.168.7.1	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=849717 TSecr=837227
76	590.508467	192.168.7.4	192.168.7.1	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
77	590.510859	192.168.7.1	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=849791 TSecr=837304
78	590.512343	192.168.7.1	192.168.7.4	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
79	590.513037	192.168.7.4	192.168.7.1	TCP	66	22 → 2016 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=837308 TSecr=849792
80	590.516075	192.168.7.1	192.168.7.4	SSHv2	930	Client: Key Exchange Init
81	590.516532	192.168.7.4	192.168.7.1	TCP	66	22 → 2016 [ACK] Seq=22 Ack=886 Win=16224 Len=0 TSval=837312 TSecr=849796
82	590.521707	192.168.7.4	192.168.7.1	SSHv2	906	Server: Key Exchange Init
83	590.524806	192.168.7.1	192.168.7.4	SSHv2	90	Client: Diffie-Hellman Group Exchange Request
84	590.533106	192.168.7.4	192.168.7.1	SSHv2	346	Server: Diffie-Hellman Group Exchange Group
85	590.547775	192.168.7.1	192.168.7.4	SSHv2	338	Client: Diffie-Hellman Group Exchange Init
86	590.562771	192.168.7.4	192.168.7.1	SSHv2	914	Server: Diffie-Hellman Group Exchange Reply, New Keys
87	590.580687	192.168.7.1	192.168.7.4	SSHv2	82	Client: New Keys
88	590.621131	192.168.7.4	192.168.7.1	TCP	66	22 → 2016 [ACK] Seq=1990 Ack=1198 Win=17952 Len=0 TSval=837416 TSecr=849860
89	590.622809	192.168.7.1	192.168.7.4	SSHv2	118	Client: Encrypted packet (len=52)
90	590.623450	192.168.7.4	192.168.7.1	TCP	66	22 → 2016 [ACK] Seq=1990 Ack=1250 Win=17952 Len=0 TSval=837418 TSecr=849902
91	590.624370	192.168.7.4	192.168.7.1	SSHv2	118	Server: Encrypted packet (len=52)
92	590.626692	192.168.7.1	192.168.7.4	SSHv2	134	Client: Encrypted packet (len=68)
94	590.668692	192.168.7.4	192.168.7.1	TCP	66	22 → 2016 [ACK] Seq=2042 Ack=1318 Win=17952 Len=0 TSval=837463 TSecr=849906
102	600.667826	192.168.7.4	192.168.7.1	SSHv2	150	Server: Encrypted packet (len=84)
106	600.714795	192.168.7.1	192.168.7.4	TCP	66	2016 → 22 [ACK] Seq=1318 Ack=2126 Win=19680 Len=0 TSval=859973 TSecr=847442
111	606.086380	192.168.7.1	192.168.7.4	TCP	74	1024 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=864819 TSecr=0 WS=32
112	606.087535	192.168.7.4	192.168.7.1	TCP	74	22 → 1024 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=852850 TSecr=864819 WS=32
113	606.090517	192.168.7.1	192.168.7.4	TCP	66	1024 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=864824 TSecr=852850
114	606.106911	192.168.7.4	192.168.7.1	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
115	606.109364	192.168.7.1	192.168.7.4	TCP	66	1024 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=864843 TSecr=852870
116	606.110209	192.168.7.1	192.168.7.4	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
117	606.110768	192.168.7.4	192.168.7.1	TCP	66	22 → 1024 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=852874 TSecr=864844

```
> Frame 73: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface -, id 0
> Ethernet II, Src: 0c:3c:3b:51:64:01 (0c:3c:3b:51:64:01), Dst: 0c:3c:3b:dc:e5:00 (0c:3c:3b:dc:e5:00)
v Internet Protocol Version 4, Src: 192.168.7.1, Dst: 192.168.7.4
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x2214 (8724)
```

Cuestión 3

Conéctate con el servidor ftp de PCC1: ftp 192.168.7.xx
name = anonymous
password = 'Una dirección de e-mail'

La dirección de correo puede ser inexistente. Trata de establecer una conexión ftp de datos mediante el comando ls dentro del ftp (ya que este comando usa una conexión de datos), tanto en modo activo como pasivo. Para los casos a) y b) descritos a continuación, verifica el correcto funcionamiento en base a la captura en las interfaces oportunas y analiza el resultado de dichas capturas.



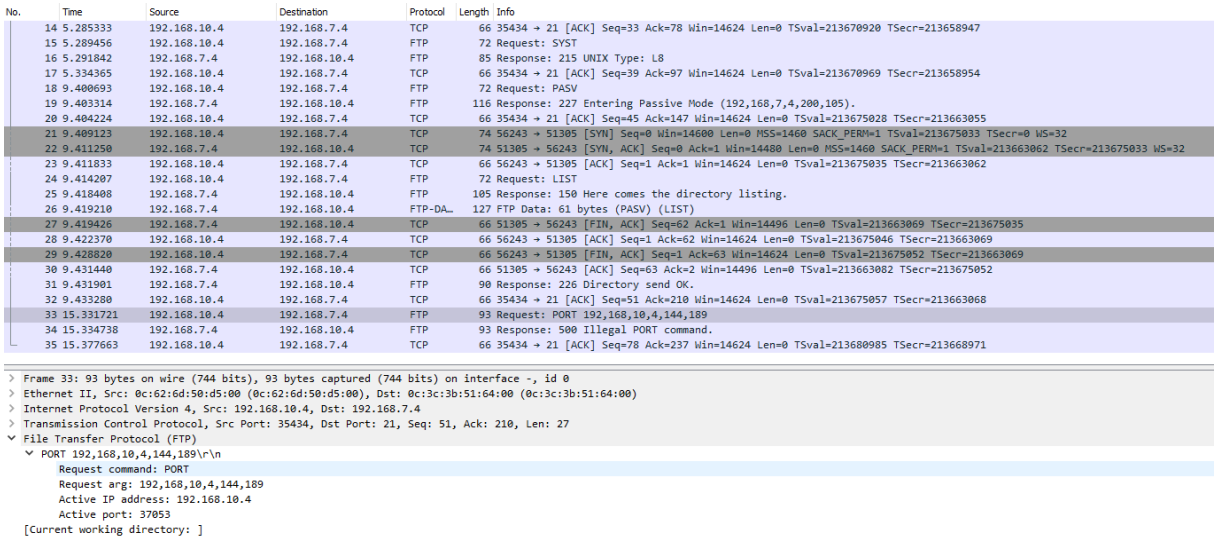
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.7.1	192.168.7.4	TCP	74	35430 → 21 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=212854208 TSecr=0 WS=32
2	0.004287	192.168.7.4	192.168.7.1	TCP	74	21 → 35430 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=212842241 TSecr=212854208 WS=32
3	0.008219	192.168.7.1	192.168.7.4	TCP	66	35430 → 21 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=212854219 TSecr=212842241
4	0.032205	192.168.7.4	192.168.7.1	FTP	86	Response: 220 (vsFTPd 2.2.2)
5	0.034825	192.168.7.1	192.168.7.4	TCP	66	35430 → 21 [ACK] Seq=1 Ack=21 Win=14624 Len=0 TSval=212854247 TSecr=212842273
6	5.011819	0c:3c:3b:51:64:01	0c:3c:3b:dc:e5:00	ARP	60	Who has 192.168.7.4? Tell 192.168.7.1
7	5.020608	0c:3c:3b:dc:e5:00	0c:3c:3b:51:64:01	ARP	60	192.168.7.4 is at 0c:3c:3b:dc:e5:00
8	12.708314	192.168.7.1	192.168.7.4	FTP	82	Request: USER anonymous
9	12.710809	192.168.7.4	192.168.7.1	TCP	66	21 → 35430 [ACK] Seq=21 Ack=17 Win=14496 Len=0 TSval=212854916 TSecr=212866886
10	12.713726	192.168.7.4	192.168.7.1	FTP	100	Response: 331 Please specify the password.
11	12.718286	192.168.7.1	192.168.7.4	TCP	66	35430 → 21 [ACK] Seq=17 Ack=55 Win=14624 Len=0 TSval=212866894 TSecr=212854921
12	20.017153	192.168.7.1	192.168.7.4	FTP	89	Request: PASS alvaro@unizar.es
13	20.027904	192.168.7.4	192.168.7.1	FTP	89	Response: 230 Login successful.
14	20.030304	192.168.7.1	192.168.7.4	TCP	66	35430 → 21 [ACK] Seq=40 Ack=78 Win=14624 Len=0 TSval=212874190 TSecr=212862216
15	20.034216	192.168.7.1	192.168.7.4	FTP	72	Request: SYST
16	20.035192	192.168.7.4	192.168.7.1	FTP	85	Response: 215 UNIX Type: L8
17	20.078167	192.168.7.1	192.168.7.4	TCP	66	35430 → 21 [ACK] Seq=46 Ack=97 Win=14624 Len=0 TSval=212874237 TSecr=212862223

En los paquetes 8 y 12 podemos observar como están al descubierto tanto el usuario introducido como la contraseña. Esto es porque el protocolo FTP, al igual que ocurre con HTTP, no cifra los datos. Es decir, los datos viajan por la red en texto plano, fácilmente observable para todo el mundo. En el caso del protocolo FTP, existe una alternativa más segura llamada SFTP.

Ahora vamos a realizar una prueba de cómo funciona el módulo **ip_nat_ftp**, este módulo se encarga de que el NAT trabaje a nivel de aplicación y no solo a nivel de red. Lo comprobaremos a continuación con una prueba en la que permitiremos a un cliente pasar a modo activo.

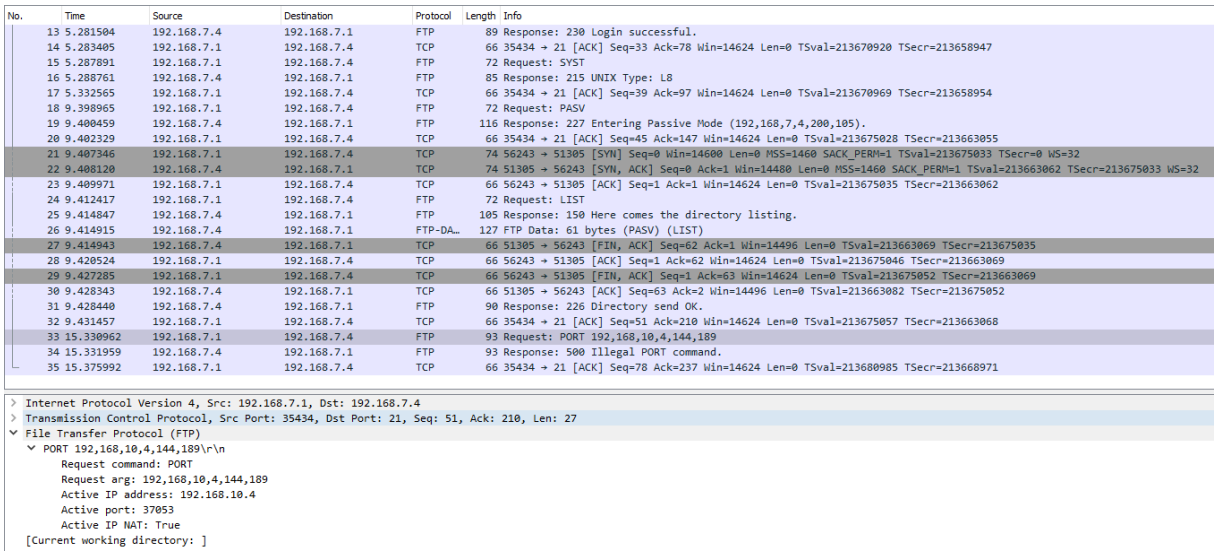
LAN A: Sin módulo **ip_nat_ftp**:



No.	Time	Source	Destination	Protocol	Length	Info
14	5.285333	192.168.10.4	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=33 Ack=78 Win=14624 Len=0 TSval=213670920 TSecr=213658947
15	5.289456	192.168.10.4	192.168.7.4	FTP	72	Request: SYST
16	5.291842	192.168.7.4	192.168.10.4	FTP	85	Response: 215 UNIX Type: L8
17	5.334365	192.168.10.4	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=39 Ack=97 Win=14624 Len=0 TSval=213670969 TSecr=213658954
18	9.400693	192.168.10.4	192.168.7.4	FTP	72	Request: PASV
19	9.403314	192.168.7.4	192.168.10.4	FTP	116	Response: 227 Entering Passive Mode (192,168,7,4,200,105).
20	9.404224	192.168.10.4	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=45 Ack=147 Win=14624 Len=0 TSval=213675028 TSecr=213663055
21	9.409123	192.168.10.4	192.168.7.4	TCP	74	56243 → 51305 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=213675033 TSecr=0 WS=32
22	9.411250	192.168.7.4	192.168.10.4	TCP	74	51305 → 56243 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=213663062 TSecr=213675033 WS=32
23	9.411833	192.168.10.4	192.168.7.4	TCP	66	56243 → 51305 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=213675035 TSecr=213663062
24	9.414207	192.168.10.4	192.168.7.4	FTP	72	Request: LIST
25	9.418408	192.168.7.4	192.168.10.4	FTP	105	Response: 150 Here comes the directory listing.
26	9.419210	192.168.7.4	192.168.10.4	FTP-DA…	127	FTP Data: 61 bytes (PASV) (LIST)
27	9.419426	192.168.7.4	192.168.10.4	TCP	66	51305 → 56243 [FIN, ACK] Seq=62 Ack=1 Win=14496 Len=0 TSval=213663069 TSecr=213675035
28	9.422370	192.168.10.4	192.168.7.4	TCP	66	56243 → 51305 [ACK] Seq=1 Ack=62 Win=14624 Len=0 TSval=213675046 TSecr=213663069
29	9.428820	192.168.10.4	192.168.7.4	TCP	66	56243 → 51305 [FIN, ACK] Seq=1 Ack=63 Win=14624 Len=0 TSval=213675052 TSecr=213663069
30	9.431440	192.168.7.4	192.168.10.4	TCP	66	51305 → 56243 [ACK] Seq=63 Ack=2 Win=14496 Len=0 TSval=213663082 TSecr=213675052
31	9.431901	192.168.7.4	192.168.10.4	FTP	90	Response: 226 Directory send OK.
32	9.433280	192.168.10.4	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=51 Ack=210 Win=14624 Len=0 TSval=213675057 TSecr=213663068
33	15.331721	192.168.10.4	192.168.7.4	FTP	93	Request: PORT 192,168,10,4,144,189
34	15.334738	192.168.7.4	192.168.10.4	FTP	93	Response: 500 Illegal PORT command.
35	15.377663	192.168.10.4	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=78 Ack=237 Win=14624 Len=0 TSval=213680985 TSecr=213668971

```
> Frame 33: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface -, id 0
> Ethernet II, Src: 0c:62:6d:50:d5:00 (0c:62:6d:50:d5:00), Dst: 0c:3c:3b:51:64:00 (0c:3c:3b:51:64:00)
> Internet Protocol Version 4, Src: 192.168.10.4, Dst: 192.168.7.4
> Transmission Control Protocol, Src Port: 35434, Dst Port: 21, Seq: 51, Ack: 210, Len: 27
v File Transfer Protocol (FTP)
  v PORT 192,168,10,4,144,189\r\n
      Request command: PORT
      Request arg: 192,168,10,4,144,189
      Active IP address: 192.168.10.4
      Active port: 37053
  [Current working directory: ]
```

LAN C: Sin módulo **ip_nat_ftp**:



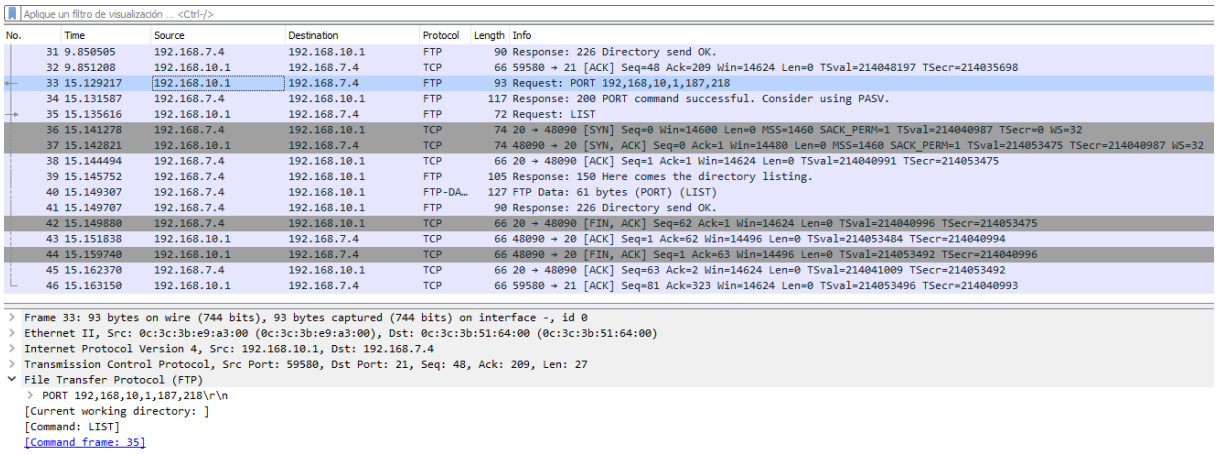
No.	Time	Source	Destination	Protocol	Length	Info
13	5.281504	192.168.7.4	192.168.7.1	FTP	89	Response: 230 Login successful.
14	5.283405	192.168.7.1	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=33 Ack=78 Win=14624 Len=0 TSval=213670920 TSecr=213658947
15	5.287891	192.168.7.1	192.168.7.4	FTP	72	Request: SYST
16	5.288761	192.168.7.4	192.168.7.1	FTP	85	Response: 215 UNIX Type: L8
17	5.332565	192.168.7.1	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=39 Ack=97 Win=14624 Len=0 TSval=213670969 TSecr=213658954
18	9.398965	192.168.7.1	192.168.7.4	FTP	72	Request: PASV
19	9.400459	192.168.7.4	192.168.7.1	FTP	116	Response: 227 Entering Passive Mode (192,168,7,4,200,105).
20	9.402329	192.168.7.1	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=45 Ack=147 Win=14624 Len=0 TSval=213675028 TSecr=213663055
21	9.407346	192.168.7.1	192.168.7.4	TCP	74	56243 → 51305 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=213675033 TSecr=0 WS=32
22	9.408120	192.168.7.4	192.168.7.1	TCP	74	51305 → 56243 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=213663062 TSecr=213675033 WS=32
23	9.409971	192.168.7.1	192.168.7.4	TCP	66	56243 → 51305 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=213675035 TSecr=213663062
24	9.412417	192.168.7.1	192.168.7.4	FTP	72	Request: LIST
25	9.414847	192.168.7.4	192.168.7.1	FTP	105	Response: 150 Here comes the directory listing.
26	9.414915	192.168.7.4	192.168.7.1	FTP-DA…	127	FTP Data: 61 bytes (PASV) (LIST)
27	9.414943	192.168.7.4	192.168.7.1	TCP	66	51305 → 56243 [FIN, ACK] Seq=62 Ack=1 Win=14496 Len=0 TSval=213663069 TSecr=213675035
28	9.420524	192.168.7.1	192.168.7.4	TCP	66	56243 → 51305 [ACK] Seq=1 Ack=62 Win=14624 Len=0 TSval=213675046 TSecr=213663069
29	9.427285	192.168.7.1	192.168.7.4	TCP	66	56243 → 51305 [FIN, ACK] Seq=1 Ack=63 Win=14624 Len=0 TSval=213675052 TSecr=213663069
30	9.428343	192.168.7.4	192.168.7.1	TCP	66	51305 → 56243 [ACK] Seq=63 Ack=2 Win=14496 Len=0 TSval=213663082 TSecr=213675052
31	9.428440	192.168.7.4	192.168.7.1	FTP	90	Response: 226 Directory send OK.
32	9.431457	192.168.7.1	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=51 Ack=210 Win=14624 Len=0 TSval=213675057 TSecr=213663068
33	15.330962	192.168.7.1	192.168.7.4	FTP	93	Request: PORT 192,168,10,4,144,189
34	15.331959	192.168.7.4	192.168.7.1	FTP	93	Response: 500 Illegal PORT command.
35	15.375992	192.168.7.1	192.168.7.4	TCP	66	35434 → 21 [ACK] Seq=78 Ack=237 Win=14624 Len=0 TSval=213680985 TSecr=213668971

```
> Internet Protocol Version 4, Src: 192.168.7.1, Dst: 192.168.7.4
> Transmission Control Protocol, Src Port: 35434, Dst Port: 21, Seq: 51, Ack: 210, Len: 27
v File Transfer Protocol (FTP)
  v PORT 192,168,10,4,144,189\r\n
      Request command: PORT
      Request arg: 192,168,10,4,144,189
      Active IP address: 192.168.10.4
      Active port: 37053
      Active IP NAT: True
  [Current working directory: ]
```

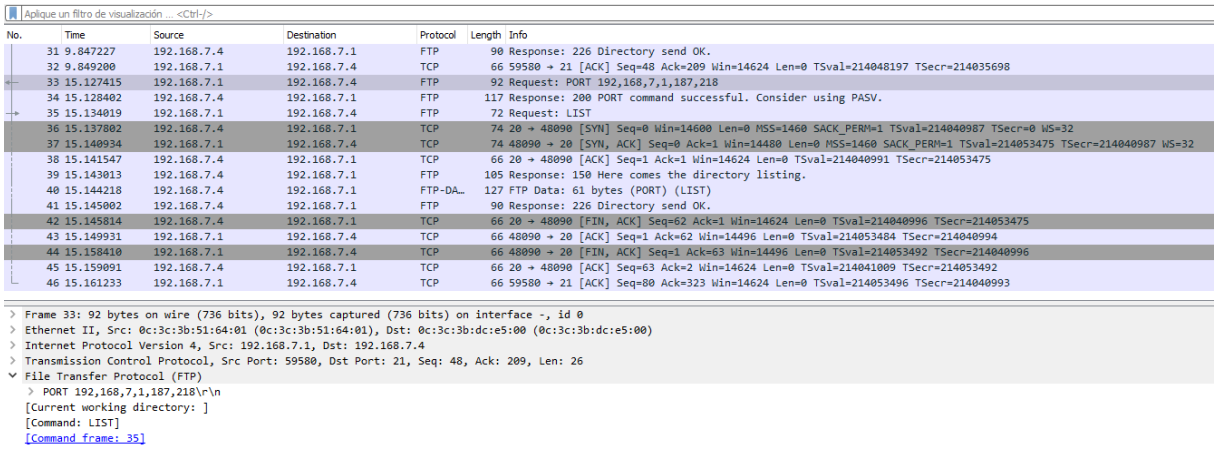
Cuando solicitamos el 'Active Mode', es decir, el modo activo, el cliente se convierte el servidor, como tal, debe proporcionar una dirección ip y un puerto. Esto lo realiza en el paquete 33, mediante un request en el que especifica la dirección ip y el puerto proporcionado. Para calcular el puerto deberemos de realizar la multiplicación de (256*penúltimo elemento) + ultimo elemento). Podemos ovservar en la captura cómo esta operación, fácilmente comprobable con la calculadora nos da el resultado de **37.053**

Como podíamos observar en las anteriores capturas, el NAT no actúa a nivel de apliación, ya que en la captura de la Lan C, se puede comprobar que el request del paquete 33 proporciona una dirección IP privada a la que, una vez le llegue al servidor destino, no va a poder acceder.

LAN A: Con módulo **ip_nat_ftp**:



LAN C: Con módulo **ip_nat_ftp**:



Tras activar el módulo **ipnat_ftp** se observa que, a diferencia de las capturas en las que este módulo no estaba activado, el NAT actúa a nivel de aplicación y en el paquete 33 de la Lan C, las direcciones IP ya no son las del PCA1 (192.168.10.1), sino que el NAT actúa y pone la dirección origen del Router origen (192.168.7.1). Esto permite que una vez esta dirección le llegue a el Servidor destino, pueda volver el paquete de manera correcta.

Se comprueba cómo el NAT ha actuado de forma correcta ya que se establece una conexión con éxito con el servidor.

Cuestión 4

Indica cómo has configurado el NAT y qué comandos has utilizado. Tal y como has configurado el NAT, ¿Es estático o dinámico?

Para configurar el NAT en el PCA3, tenemos que introducir los siguientes comandos que permitiran traducir las direcciones de red para que el paquete pueda llegar desde una red privada a otra red privada.

PCB3

1. `iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 192.168.7.2`
2. `iptables -t nat -A PREROUTING -i eth2 -j DNAT --to 192.168.7.1`
3. `iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 192.168.20.3`

En el primer comando, hemos hecho un POSTROUTING, es decir, estamos modificando la dirección fuente (PCA3) del paquete, justo antes de hacer finalmente el envío. Con este comando cambiamos la dirección fuente del paquete al valor 192.168.7.1 antes de enviarlo por la interfaz de salida eht1.

En el segundo comando, hemos hecho un PREROUTING, es decir, hemos cambiado la dirección destino (PCA3) del paquete, justo antes de que el paquete entre en el resto de funcionalidades de procesado del paquetes. Hemos cambiado la dirección destino al valor 192.168.7.1 al momento de recibir el paquete por la interfaz de entrada eth2

En el tercer comando, hemos hecho un PREROUTING, es decir, hemos cambiado la dirección destino del paquete, justo antes de que el paquete entre en el resto de funcionalidades de procesado del paquetes. Hemos cambiado la dirección destino (PCB1) al valor 192.168.20.3 al momento de recibir el paquete por la interfaz de entrada eth1

PCA3

1. `iptables -t nat -A POSTROUTING -o eth2 -j SNAT --to 192.168.7.1`
2. `iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 192.168.7.2`
3. `iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 192.168.10.1`

En el PCA3 hemos seguido los mismos pasos que en el PCB3, simplemente hemos jugado con las interfaces del PCA3 de la misma forma que con el PCB3.

Hemos mapeado una dirección IP privada con una dirección IP pública de forma estática. De esta manera, cada equipo en la red privada debe tener su correspondiente IP pública asignada para poder acceder a Internet. Por tanto, las configuraciones que hemos realizado en este caso son estáticas.

Cuestión 5

Tomando como base las capturas, verifica y demuestra el funcionamiento del servicio ssh.

SSH LAN A

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.1	192.168.20.3	TCP	74	59898 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=7754210 TSecr=0 WS=32
2	0.008048	192.168.20.3	192.168.10.1	TCP	74	22 → 59898 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=7731887 TSecr=7754210 WS=32
3	0.008670	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=7754218 TSecr=7731887
4	0.048315	192.168.20.3	192.168.10.1	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
5	0.049127	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=7754259 TSecr=7731929
6	0.050628	192.168.10.1	192.168.20.3	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
7	0.054576	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=7731936 TSecr=7754260
8	0.055192	192.168.10.1	192.168.20.3	SSHv2	930	Client: Key Exchange Init
9	0.059039	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=22 Ack=886 Win=16224 Len=0 TSval=7731940 TSecr=7754265
10	0.064533	192.168.20.3	192.168.10.1	SSHv2	906	Server: Key Exchange Init
11	0.065407	192.168.10.1	192.168.20.3	SSHv2	90	Client: Diffie-Hellman Group Exchange Request
12	0.074307	192.168.20.3	192.168.10.1	SSHv2	346	Server: Diffie-Hellman Group Exchange Group
13	0.081270	192.168.10.1	192.168.20.3	SSHv2	338	Client: Diffie-Hellman Group Exchange Init
14	0.118006	192.168.20.3	192.168.10.1	SSHv2	914	Server: Diffie-Hellman Group Exchange Reply, New Keys
15	0.131156	192.168.10.1	192.168.20.3	SSHv2	82	Client: New Keys
16	0.175787	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=1990 Ack=1198 Win=17952 Len=0 TSval=7732056 TSecr=7754341
17	0.176312	192.168.10.1	192.168.20.3	SSHv2	118	Client: Encrypted packet (len=52)
18	0.180837	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=1990 Ack=1250 Win=17952 Len=0 TSval=7732061 TSecr=7754386
19	0.182056	192.168.20.3	192.168.10.1	SSHv2	118	Server: Encrypted packet (len=52)
20	0.182833	192.168.10.1	192.168.20.3	SSHv2	134	Client: Encrypted packet (len=68)
21	0.195808	192.168.20.3	192.168.10.1	SSHv2	150	Server: Encrypted packet (len=84)
34	0.236260	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1318 Ack=2126 Win=19680 Len=0 TSval=7754446 TSecr=7732077
35	3.724370	192.168.10.1	192.168.20.3	SSHv2	214	Client: Encrypted packet (len=148)
36	3.767757	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=2126 Ack=1466 Win=19680 Len=0 TSval=7735642 TSecr=7757927
37	3.791696	192.168.20.3	192.168.10.1	SSHv2	102	Server: Encrypted packet (len=36)
38	3.792322	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1466 Ack=2162 Win=19680 Len=0 TSval=7757995 TSecr=7735665
39	3.796408	192.168.10.1	192.168.20.3	SSHv2	202	Client: Encrypted packet (len=136)
40	3.800282	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=2162 Ack=1602 Win=21408 Len=0 TSval=7735674 TSecr=7757999
41	3.934141	192.168.20.3	192.168.10.1	SSHv2	118	Server: Encrypted packet (len=52)
42	3.935432	192.168.10.1	192.168.20.3	SSHv2	526	Client: Encrypted packet (len=460)
43	3.938447	192.168.20.3	192.168.10.1	TCP	66	22 → 59898 [ACK] Seq=2214 Ack=2062 Win=23136 Len=0 TSval=7735812 TSecr=7758138
44	3.943795	192.168.20.3	192.168.10.1	SSHv2	190	Server: Encrypted packet (len=124)
45	3.948081	192.168.20.3	192.168.10.1	SSHv2	166	Server: Encrypted packet (len=100)
46	3.951179	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=2062 Ack=2438 Win=19680 Len=0 TSval=7758151 TSecr=7735818
47	4.023859	192.168.20.3	192.168.10.1	SSHv2	134	Server: Encrypted packet (len=68)
48	4.065323	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=2062 Ack=2506 Win=19680 Len=0 TSval=7758267 TSecr=7735897
51	8.400314	192.168.10.1	192.168.20.3	SSHv2	118	Client: Encrypted packet (len=52)
52	8.404807	192.168.20.3	192.168.10.1	SSHv2	118	Server: Encrypted packet (len=52)
53	8.405242	192.168.10.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=2114 Ack=2558 Win=19680 Len=0 TSval=7762600 TSecr=7740271

En la LAN A, podemos observar las direcciones origen, aún sin modificar por los NAT del PCA3 y del PCB3. En esta captura se observa la petición que realiza el cliente SSH (Puerto 59898) al servidor (Puerto Ssh 22), pues tras realizar el intercambio de llaves SSH, comienza la conexión (cifrada como se puede ver). Lo más importante a recalcar para darnos cuenta como actúa el NAT es que podemos ver cómo el reply, a pesar de haber pasado por dos redes, nos muestra como dirección origen la dirección del PCB3, cuando realmente la máquina que está traduciendo las direcciones es el PCA3 (192.168.10.254 en la LAN A).

En la LAN C (siguiente captura), podemos observar también las direcciones origen y destino como han sido traducidas, tenemos como dirección origen la interfaz de la LAN C del PCA3 y como destino la interfaz de la LAN C del PCB3.

SSH LAN C

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.7.1	192.168.7.2	TCP	74	59898 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=7754210 TSecr=0 WS=32
2	0.004900	192.168.7.2	192.168.7.1	TCP	74	22 → 59898 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=7731887 TSecr=7754210 WS=32
3	0.007084	192.168.7.1	192.168.7.2	TCP	66	59898 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=7754218 TSecr=7731887
4	0.045482	192.168.7.2	192.168.7.1	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
5	0.047443	192.168.7.1	192.168.7.2	TCP	66	59898 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=7754259 TSecr=7731929
6	0.049262	192.168.7.1	192.168.7.2	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
7	0.051931	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=7731936 TSecr=7754260
8	0.053677	192.168.7.1	192.168.7.2	SSHv2	930	Client: Key Exchange Init
9	0.056249	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=22 Ack=886 Win=16224 Len=0 TSval=7731940 TSecr=7754265
10	0.061745	192.168.7.2	192.168.7.1	SSHv2	906	Server: Key Exchange Init
11	0.064650	192.168.7.1	192.168.7.2	SSHv2	90	Client: Diffie-Hellman Group Exchange Request
12	0.071601	192.168.7.2	192.168.7.1	SSHv2	346	Server: Diffie-Hellman Group Exchange Group
13	0.079951	192.168.7.1	192.168.7.2	SSHv2	338	Client: Diffie-Hellman Group Exchange Init
14	0.114855	192.168.7.2	192.168.7.1	SSHv2	914	Server: Diffie-Hellman Group Exchange Reply, New Keys
15	0.130008	192.168.7.1	192.168.7.2	SSHv2	82	Client: New Keys
16	0.172732	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=1990 Ack=1198 Win=17952 Len=0 TSval=7732056 TSecr=7754341
17	0.174921	192.168.7.1	192.168.7.2	SSHv2	118	Client: Encrypted packet (len=52)
18	0.178065	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=1990 Ack=1250 Win=17952 Len=0 TSval=7732061 TSecr=7754386
19	0.179354	192.168.7.2	192.168.7.1	SSHv2	118	Server: Encrypted packet (len=52)
20	0.181492	192.168.7.1	192.168.7.2	SSHv2	134	Client: Encrypted packet (len=68)
25	0.193142	192.168.7.2	192.168.7.1	SSHv2	150	Server: Encrypted packet (len=84)
38	0.234662	192.168.7.1	192.168.7.2	TCP	66	59898 → 22 [ACK] Seq=1318 Ack=2126 Win=19680 Len=0 TSval=7754446 TSecr=7732077
39	3.723141	192.168.7.1	192.168.7.2	SSHv2	214	Client: Encrypted packet (len=148)
40	3.764660	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=2126 Ack=1466 Win=19680 Len=0 TSval=7735642 TSecr=7757927
41	3.788508	192.168.7.2	192.168.7.1	SSHv2	102	Server: Encrypted packet (len=36)
42	3.790973	192.168.7.1	192.168.7.2	TCP	66	59898 → 22 [ACK] Seq=1466 Ack=2162 Win=19680 Len=0 TSval=7757995 TSecr=7735665
43	3.795163	192.168.7.1	192.168.7.2	SSHv2	202	Client: Encrypted packet (len=136)
44	3.797227	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=2162 Ack=1602 Win=21408 Len=0 TSval=7735674 TSecr=7757999
45	3.931076	192.168.7.2	192.168.7.1	SSHv2	118	Server: Encrypted packet (len=52)
46	3.934106	192.168.7.1	192.168.7.2	SSHv2	526	Client: Encrypted packet (len=460)
47	3.935476	192.168.7.2	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=2214 Ack=2062 Win=23136 Len=0 TSval=7735812 TSecr=7758138
48	3.941091	192.168.7.2	192.168.7.1	SSHv2	190	Server: Encrypted packet (len=124)
49	3.945371	192.168.7.2	192.168.7.1	SSHv2	166	Server: Encrypted packet (len=100)
50	3.950357	192.168.7.1	192.168.7.2	TCP	66	59898 → 22 [ACK] Seq=2062 Ack=2438 Win=19680 Len=0 TSval=7758151 TSecr=7735818
51	4.020757	192.168.7.2	192.168.7.1	SSHv2	134	Server: Encrypted packet (len=68)
52	4.064070	192.168.7.1	192.168.7.2	TCP	66	59898 → 22 [ACK] Seq=2062 Ack=2506 Win=19680 Len=0 TSval=7758267 TSecr=7735897
55	8.398804	192.168.7.1	192.168.7.2	SSHv2	118	Client: Encrypted packet (len=52)

SSH LAN B

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.7.1	192.168.20.3	TCP	74	59898 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=7754210 TSecr=0 WS=32
2	0.002414	192.168.20.3	192.168.7.1	TCP	74	22 → 59898 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=7731887 TSecr=7754210 WS=32
3	0.005563	192.168.7.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=7754218 TSecr=7731887
4	0.043036	192.168.20.3	192.168.7.1	SSHv2	87	Server: Protocol (SSH-2.0-OpenSSH_5.3)
5	0.046162	192.168.7.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=7754259 TSecr=7731929
6	0.049026	192.168.7.1	192.168.20.3	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_5.3)
7	0.049699	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=7731936 TSecr=7754260
8	0.053188	192.168.7.1	192.168.20.3	SSHv2	930	Client: Key Exchange Init
9	0.053585	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=22 Ack=886 Win=16224 Len=0 TSval=7731940 TSecr=7754265
10	0.059262	192.168.20.3	192.168.7.1	SSHv2	906	Server: Key Exchange Init
11	0.063122	192.168.7.1	192.168.20.3	SSHv2	90	Client: Diffie-Hellman Group Exchange Request
12	0.069117	192.168.20.3	192.168.7.1	SSHv2	346	Server: Diffie-Hellman Group Exchange Group
13	0.078560	192.168.7.1	192.168.20.3	SSHv2	338	Client: Diffie-Hellman Group Exchange Init
14	0.111997	192.168.20.3	192.168.7.1	SSHv2	914	Server: Diffie-Hellman Group Exchange Reply, New Keys
15	0.128668	192.168.7.1	192.168.20.3	SSHv2	82	Client: New Keys
16	0.170200	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=1990 Ack=1198 Win=17952 Len=0 TSval=7732056 TSecr=7754341
17	0.174621	192.168.7.1	192.168.20.3	SSHv2	118	Client: Encrypted packet (len=52)
18	0.175081	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=1990 Ack=1250 Win=17952 Len=0 TSval=7732061 TSecr=7754386
19	0.175871	192.168.20.3	192.168.7.1	SSHv2	118	Server: Encrypted packet (len=52)
20	0.180051	192.168.7.1	192.168.20.3	SSHv2	134	Client: Encrypted packet (len=68)
25	0.190626	192.168.20.3	192.168.7.1	SSHv2	150	Server: Encrypted packet (len=84)
38	0.233287	192.168.7.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1318 Ack=2126 Win=19680 Len=0 TSval=7754446 TSecr=7732077
39	3.721825	192.168.7.1	192.168.20.3	SSHv2	214	Client: Encrypted packet (len=148)
40	3.762173	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=2126 Ack=1466 Win=19680 Len=0 TSval=7735642 TSecr=7757927
41	3.785889	192.168.20.3	192.168.7.1	SSHv2	102	Server: Encrypted packet (len=36)
42	3.793692	192.168.7.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=1466 Ack=2162 Win=19680 Len=0 TSval=7757995 TSecr=7735665
43	3.794291	192.168.7.1	192.168.20.3	SSHv2	202	Client: Encrypted packet (len=136)
44	3.794631	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=2162 Ack=1602 Win=21408 Len=0 TSval=7735674 TSecr=7757999
45	3.928053	192.168.20.3	192.168.7.1	SSHv2	118	Server: Encrypted packet (len=52)
46	3.932658	192.168.7.1	192.168.20.3	SSHv2	526	Client: Encrypted packet (len=460)
47	3.933127	192.168.20.3	192.168.7.1	TCP	66	22 → 59898 [ACK] Seq=2214 Ack=2062 Win=23136 Len=0 TSval=7735812 TSecr=7758138
48	3.938517	192.168.20.3	192.168.7.1	SSHv2	190	Server: Encrypted packet (len=124)
49	3.942884	192.168.20.3	192.168.7.1	SSHv2	166	Server: Encrypted packet (len=100)
50	3.949309	192.168.7.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=2062 Ack=2438 Win=19680 Len=0 TSval=7758151 TSecr=7735818
51	4.017953	192.168.20.3	192.168.7.1	SSHv2	134	Server: Encrypted packet (len=68)
52	4.062667	192.168.7.1	192.168.20.3	TCP	66	59898 → 22 [ACK] Seq=2062 Ack=2506 Win=19680 Len=0 TSval=7758267 TSecr=7735897
55	8.397367	192.168.7.1	192.168.20.3	SSHv2	118	Client: Encrypted packet (len=52)

En la LAN B, podemos observar que la dirección destino ha sido traducida, tenemos como dirección destino la propia del servidor SSH al que nos estamos conectando. En todas estas capturas se puede comprobar el correcto funcionamiento de la configuración del NAT realizada y cómo tanto en los request como en los reply, se aprecian las intrucciones proporcionadas de traducción de direcciones, es decir, el correcto funcionamiento del NAT.

Cuestión 6

En base a las capturas del tráfico, calcula el trhougput (en pps) y el ancho de banda (en bps) total, el útil a nivel IP y el útil a nivel de aplicación, que se consigue en el escenario propuesto para todos los casos de generación de tráfico.

En este ejercicio vamos a generar paquetes entre el PCA1 y el PCC2 de diferentes tamaños. El objetivo es estimar el máximo ancho de banda que podemos conseguir para una transferencia fiable entre PCA1 y PCC2.

TCP

Comenzaremos generando paquetes TCP que son capaces de controlar la congestión de la red y adaptarse a los límites de la misma. Para ello estableceremos PCC2 como servidor TCP con el comando: **iperf -s**.

A continuación lanzamos el comando:

iperf -c 192.168.7.4 -t 10

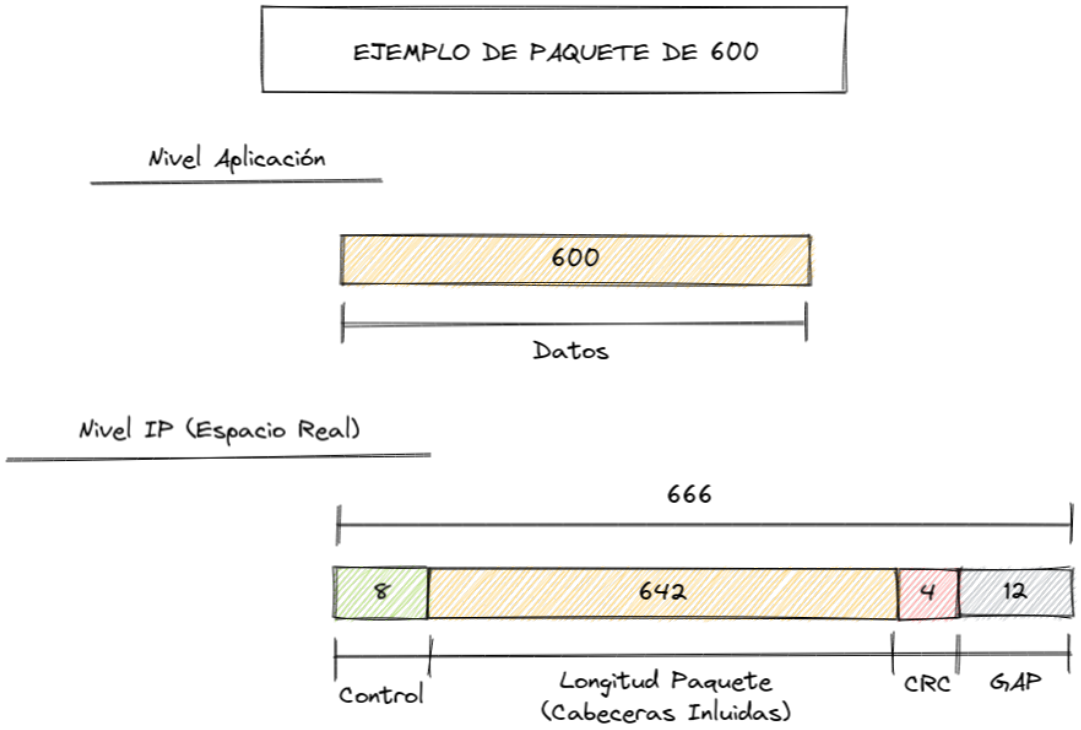
Este comando iperf nos permitirá medir el ancho de banda de nuestra red en una prueba durante 10 segundos:

No.	Time	Source	Destination	Protocol	Length	Info
194	21.009816	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=140481 Ack=1 Win=14624 Len=1448 TSval=130832 TSecr=244622
195	21.009888	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=141929 Ack=1 Win=14624 Len=1448 TSval=130832 TSecr=244622
196	21.009957	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#46] 45256 → 5001 [ACK] Seq=143377 Ack=1 Win=14624 Len=0 TSval=130832 TSecr=244622
197	21.010036	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=143377 Ack=1 Win=14624 Len=1448 TSval=130832 TSecr=244623
198	21.010229	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=144825 Ack=1 Win=14624 Len=1448 TSval=130832 TSecr=244623
199	21.010324	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#47] 45256 → 5001 [ACK] Seq=146273 Ack=1 Win=14624 Len=0 TSval=130832 TSecr=244623
200	21.013127	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=146273 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244626
201	21.013352	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=147721 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244626
202	21.013647	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=89801 Win=110048 Len=0 TSval=244630 TSecr=130805
203	21.013821	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#48] 45256 → 5001 [ACK] Seq=149169 Ack=1 Win=14624 Len=0 TSval=130840 TSecr=244626
204	21.013944	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=91249 Win=112960 Len=0 TSval=244630 TSecr=130805
205	21.014249	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=92697 Win=115840 Len=0 TSval=244631 TSecr=130805
206	21.014418	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=94145 Win=118752 Len=0 TSval=244631 TSecr=130805
207	21.014948	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=95593 Win=121632 Len=0 TSval=244631 TSecr=130805
208	21.015591	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=98489 Win=121632 Len=0 TSval=244632 TSecr=130805
209	21.016370	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=101385 Win=121632 Len=0 TSval=244633 TSecr=130805
210	21.016832	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=104281 Win=121632 Len=0 TSval=244633 TSecr=130819
211	21.017022	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=107177 Win=121632 Len=0 TSval=244633 TSecr=130819
212	21.017478	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=149169 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244630
213	21.017695	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [PSH, ACK] Seq=150617 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244630
214	21.017922	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#49] 45256 → 5001 [PSH, ACK] Seq=152065 Ack=1 Win=14624 Len=0 TSval=130840 TSecr=2…
215	21.018184	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=152065 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244631
216	21.018407	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=153513 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244631
217	21.018814	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#50] 45256 → 5001 [ACK] Seq=154961 Ack=1 Win=14624 Len=0 TSval=130840 TSecr=244631
218	21.019534	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=110073 Win=121632 Len=0 TSval=244636 TSecr=130819
219	21.019979	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=112969 Win=121632 Len=0 TSval=244636 TSecr=130819
220	21.020134	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=154961 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244631
221	21.020245	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=115865 Win=121632 Len=0 TSval=244637 TSecr=130824
222	21.020551	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=156409 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244631
223	21.020731	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#51] 45256 → 5001 [ACK] Seq=157857 Ack=1 Win=14624 Len=0 TSval=130840 TSecr=244631
224	21.021078	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=118761 Win=121632 Len=0 TSval=244637 TSecr=130824
225	21.022318	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=157857 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244632
226	21.022471	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=159305 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244632
227	21.023524	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=121657 Win=121632 Len=0 TSval=244639 TSecr=130824
228	21.023623	192.168.10.1	192.168.7.4	TCP	66	[TCP Dup ACK 4#52] 45256 → 5001 [ACK] Seq=160753 Ack=1 Win=14624 Len=0 TSval=130840 TSecr=244632
229	21.023799	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=124553 Win=121632 Len=0 TSval=244640 TSecr=130824
230	21.025316	192.168.7.4	192.168.10.1	TCP	66	5001 → 45256 [ACK] Seq=1 Ack=127449 Win=121632 Len=0 TSval=244641 TSecr=130824
231	21.025457	192.168.10.1	192.168.7.4	TCP	1514	45256 → 5001 [ACK] Seq=160753 Ack=1 Win=14624 Len=1448 TSval=130840 TSecr=244633

Hemos utilizado iperf con TCP para poder saber el máximo ancho de banda posible en la red sin que se congestione. Esto nos permitirá suponer un máximo ancho de banda teórico disponible de nuestro enlace para poder calcular el throughput.

Utilizaremos el comando **iperf -c (dirección_dest) -t (tiempo) -u -b (bandwith) -l (buffer)**, el parámetro bandwith lo dejamos fijo a 20Mbps. Sin embargo, variaremos la longitud del buffer para leer o escribir. La dirección será la de nuestra máquina PCC2.

Para calcular los diferentes anchos de banda según consideremos tomar la longitud de los paquetes a nivel de aplicación o considerar lo que verdaderamente ocupa un paquete en la red, se ilustra a continuación un esquema de cómo se interpreta los paquetes según a nivel de aplicación o a nivel IP.



En este sencillo esquema podemos ver cómo existe una diferencia entre lo que se considera un paquete a nivel de aplicación y lo que verdaderamente es a través de la red. Para ejemplos en los que los paquetes son de un tamaño considerable, añadirle 66 bytes más al paquete no supondrá mucha diferencia entre el ancho de banda que la aplicación cree que ocupa y lo que verdaderamente ocupa. Sin embargo, cuando los paquetes son de 100 bytes de datos, como es el caso de nuestro primer experimento, habrá una gran diferencia entre considerar el paquete de 100 bytes o de 166 bytes.

Iperf con buffer de 100 Bytes

```
[root@localhost ~]# iperf -c 192.168.7.4 -t 3 -u -b 20000000 -l 100
------------------------------------------------------------
Client connecting to 192.168.7.4, UDP port 5001
Sending 100 byte datagrams
UDP buffer size:  110 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 54819 connected with 192.168.7.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 3.0 sec   513 KBytes  1.40 Mbits/sec
[  3] Sent 5254 datagrams
[  3] Server Report:
[  3]  0.0- 3.0 sec   484 KBytes  1.32 Mbits/sec   0.254 ms  298/ 5253
   (5.7%)
[  3]  0.0- 3.0 sec  1 datagrams received out-of-order
```

Cuando tenemos un buffer de 100 bytes, obtenemos 1746.9 pps. A nivel de aplicación, estamos usando 1,40 Mbits/s de la red, aunque en realidad, a nivel IP, teniendo en cuenta el esquema anterior, estamos usando ~~2,21~~ Mbit/s de la red. Como podemos observar, cambia considerablemente el uso de la red.

Iperf con buffer de 200 Bytes

```
[root@localhost ~]# iperf -c 192.168.7.4 -t 3 -u -b 20000000 -l 200
------------------------------------------------------------
Client connecting to 192.168.7.4, UDP port 5001
Sending 200 byte datagrams
UDP buffer size:  110 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 40431 connected with 192.168.7.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 3.0 sec   974 KBytes  2.66 Mbits/sec
[  3] Sent 4986 datagrams
[  3] Server Report:
[  3]  0.0- 3.0 sec   974 KBytes  2.66 Mbits/sec   0.357 ms    0/ 4985
   (0%)
[  3]  0.0- 3.0 sec  1 datagrams received out-of-order
```

Cuando tenemos un buffer de 200 bytes, obtenemos 1657.4 pps. A nivel de aplicación, estamos usando 2,66 Mbits/s de la red, aunque en realidad, a nivel IP, estamos usando ~~3,36~~ Mbit/s de la red. Como podemos observar, cambia considerablemente el uso de la red.

Iperf con buffer de 300 bytes

```
[root@localhost ~]# iperf -c 192.168.7.4 -t 3 -u -b 20000000 -l 300
------------------------------------------------------------
Client connecting to 192.168.7.4, UDP port 5001
Sending 300 byte datagrams
UDP buffer size:  110 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 43364 connected with 192.168.7.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 3.0 sec  1.54 MBytes  4.30 Mbits/sec
[  3] Sent 5382 datagrams
[  3] Server Report:
[  3]  0.0- 3.0 sec  1.52 MBytes  4.26 Mbits/sec   0.595 ms   63/ 5381
   (1.2%)
[  3]  0.0- 3.0 sec  1 datagrams received out-of-order
```

Cuando tenemos un buffer de 300 bytes, obtenemos 1778.4 pps. A nivel de aplicación, estamos usando 4,30 Mbits/s de la red, aunque en realidad, a nivel IP, estamos usando ~~4.99~~ Mbit/s de la red. Como podemos observar, ya no cambia considerablemente el uso de la red.

Iperf con buffer de 600 bytes

```
[root@localhost ~]# iperf -c 192.168.7.4 -t 3 -u -b 20000000 -l 600
------------------------------------------------------------
Client connecting to 192.168.7.4, UDP port 5001
Sending 600 byte datagrams
UDP buffer size:  110 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 44749 connected with 192.168.7.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 3.0 sec  2.69 MBytes  7.53 Mbits/sec
[  3] Sent 4708 datagrams
[  3] Server Report:
[  3]  0.0- 3.0 sec  2.68 MBytes  7.51 Mbits/sec   0.393 ms   15/ 4707
   (0.32%)
[  3]  0.0- 3.0 sec  1 datagrams received out-of-order
```

Cuando tenemos un buffer de 600 bytes, obtenemos 1552.4 pps. A nivel de aplicación, estamos usando 7,53 Mbits/s de la red, aunque en realidad, a nivel IP, estamos usando ~~7,95~~ Mbit/s de la red. Como podemos observar, disminuye la diferencia en el uso de la red entre ambas consideraciones.

Iperf con buffer de 1200 bytes

```
[root@localhost ~]# iperf -c 192.168.7.4 -t 3 -u -b 20000000 -l 1200
------------------------------------------------------------
Client connecting to 192.168.7.4, UDP port 5001
Sending 1200 byte datagrams
UDP buffer size:  110 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 33426 connected with 192.168.7.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 3.0 sec  4.39 MBytes  12.3 Mbits/sec
[  3] Sent 3835 datagrams
[  3] Server Report:
[  3]  0.0- 3.0 sec  4.25 MBytes  11.9 Mbits/sec   0.594 ms  121/ 3834
   (3.2%)
[  3]  0.0- 3.0 sec  1 datagrams received out-of-order
```

Cuando tenemos un buffer de 1200 bytes, obtenemos 1275.1 pps. A nivel de aplicación, estamos usando 12,3 Mbits/s de la red, aunque en realidad, a nivel IP, estamos usando 12,31 Mbit/s de la red. Como podemos observar, la diferencia en el uso de la red entre ambas consideraciones empieza a no ser influyente.

Iperf con buffer de 1478 bytes

```
root@localhost ~]# iperf -c 192.168.7.4 -t 3 -u -b 20000000 -l 1472
------------------------------------------------------------
Client connecting to 192.168.7.4, UDP port 5001
Sending 1472 byte datagrams
UDP buffer size:  110 KByte (default)
------------------------------------------------------------
[  3] local 192.168.10.1 port 59505 connected with 192.168.7.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 3.0 sec  5.44 MBytes  15.2 Mbits/sec
[  3] Sent 3876 datagrams
[  3] Server Report:
[  3]  0.0- 3.0 sec  5.43 MBytes  15.2 Mbits/sec   0.805 ms    0/ 3875
   (0%)
[  3]  0.0- 3.0 sec  1 datagrams received out-of-order
```

Cuando tenemos un buffer de 1472 bytes, obtenemos 1275.1 pps. A nivel de aplicación, estamos usando 15,2 Mbits/s de la red, aunque en realidad, a nivel IP, estamos usando ~~15,2~~ Mbit/s de la red. Como podemos observar, la diferencia en el uso de la red entre ambas consideraciones no es influyente si los paquetes son grandes.

Cuestión 7, 8 y 9 en el Entregable Individual