Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

dar-pr-1

Diseño y administración de redes

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

9 de octubre de 2023

Índice

01

Pregunta 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 Network Address Translation (NAT) en nuestras redes internas LAN_A y LAN_B, utilizaremos iptables, una herramienta de administración de reglas de firewall en sistemas Linux que a su vez permite la configuración del NAT. Esto se logrará mediante la configuración de una regla de postrouting SNAT en la tabla nat, que garantizará que todos los paquetes que salgan de la red tengan como dirección de origen la del router.

```
# En PCA3
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 192.168.7.10
```

```
# En PCB3
iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 192.168.7.20
```

Parámetro	**Descripción**
-t nat	Tabla de filtrado NAT para modificar paquetes de Network Address Translation (NAT).
-A POSTROUTING	Agrega una regla al final de la cadena POSTROUTING, consultada antes de que los paquetes salgan de la red.
-o eth1	Especifica la interfaz de red de salida (eth1) a través de la cual se aplicará la regla.
-j SNAT -to 192.168.7.10	Establece que si un paquete coincide con la regla, se aplicará la acción SNAT para modificar la dirección de origen del paquete a 192.168.7.10. Útil en enrutamiento y NAT.

Al usar como tarjet “SNAT” (Source NAT) para configurar NAT en nuestras redes internas LAN_A y LAN_B, hemos establecido un NAT ~~estático~~. Esto implica que nuestras direcciones de origen para todos los paquetes que salen de la red serán constantes y corresponderán a la direcciónes del router en la red C (192.168.7.10) en el caso de la LAN_A y (192.168.7.20) en el caso de la LAN_B. Si hubiéramos optado por configurar el NAT utilizando la regla MASQUERADE, habríamos implementado un NAT dinámico, lo que permitiría que la dirección de origen varíe dinámicamente en función de la disponibilidad y el enrutamiento de la red, lo cual puede ser útil en escenarios diferentes.

02

Pregunta 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. Nota: Para ver el status del NAT, se debe usar el comando: cat /proc/net/nf_conntrack

---

ICMP

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.10.1	192.168.7.2	ICMP	98	Echo (ping) request id=0x1306, seq=1/256, ttl=64 (reply in 2)
2	0.004997	192.168.7.2	192.168.10.1	ICMP	98	Echo (ping) reply id=0x1306, seq=1/256, ttl=63 (request in 1)
9	9.237554	192.168.10.2	192.168.7.2	ICMP	98	Echo (ping) request id=0x2706, seq=1/256, ttl=64 (reply in 12)
12	9.243537	192.168.7.2	192.168.10.2	ICMP	98	Echo (ping) reply id=0x2706, seq=1/256, ttl=63 (request in 9)

Figura 1: Petición y respuesta ping desde PCA1 y PCA2 a PCC2 capturado en la lan A.

No.	Time	Source	Destination	Protocol	Length	Info
29	25.875247	192.168.7.10	192.168.7.2	ICMP	98	Echo (ping) request id=0x1306, seq=1/256, ttl=63 (reply in 30)
30	25.877232	192.168.7.2	192.168.7.10	ICMP	98	Echo (ping) reply id=0x1306, seq=1/256, ttl=64 (request in 29)
49	35.111801	192.168.7.10	192.168.7.2	ICMP	98	Echo (ping) request id=0x2706, seq=1/256, ttl=63 (reply in 50)
50	35.112801	192.168.7.2	192.168.7.10	ICMP	98	Echo (ping) reply id=0x2706, seq=1/256, ttl=64 (request in 49)

Figura 2: Petición y respuesta ping desde PCA1 y PCA2 a PCC2 capturado en la lan C.

En las capturas 1 y 2 se pueden observar los paquetes de tipo “request” y “response” que fueron generados por las máquinas PCA1 y PCA2 y enviados hacia PCC2. Los paquetes mostrados en las figuras 1 y 2 tienen una correspondencia uno a uno, lo que significa que cada paquete en la figura 1 se corresponde con un paquete específico en la figura 2. Esta correspondencia se ha comprobado verificando el contenido del campo “identification” (identificación) dentro de la cabecera IP de cada paquete.

Como se puede observar en las figuras, en la red LAN_A, la dirección de origen de los paquetes es 192.168.10.1. Sin embargo, cuando estos paquetes salen de la red LAN_A para dirigirse a LAN_C, se ha aplicado la regla de NAT configurada en la sección 1 para cambiar la dirección de origen. En este caso, los paquetes han sido modificados para que la dirección de origen sea la del router, que es 192.168.7.10, antes de salir al exterior de la red. Esto permite preservar la privacidad de las direcciones IP internas y permitir que múltiples dispositivos compartan una única dirección IP pública cuando se comunican fuera de la red local.

SSH

```
# En PCA1
iptables -t nat -A POSTROUTING -o eth0 -p tcp --dport 22 -j SNAT --to 192.168.10.1:2016
```

```
# En PCA2
iptables -t nat -A POSTROUTING -o eth0 -p tcp --dport 22 -j SNAT --to 192.168.10.2:2016
```

Parámetro	Descripción
-t nat	Tabla de filtrado NAT para modificar paquetes de Network Address Translation (NAT).
-A POSTROUTING	Agrega una regla al final de la cadena POSTROUTING, consultada antes de que los paquetes salgan de la red.
-o eth0	Especifica la interfaz de red de salida (eth0) a través de la cual se aplicará la regla.
-p tcp	Protocolo del paquete a comprobar.
-dport 22	Puerto de destino.
-j SNAT –to 192.168.7.10	Establece que si un paquete coincide con la regla, se aplicará la acción SNAT (source nat) para modificar la dirección de origen del paquete a 192.168.7.10. Útil en enrutamiento y NAT.

Al aplicar estas reglas en PCA1 y PCA2, modificamos la dirección de origen y el puerto de origen cuando se establece una conexión SSH saliente a través de la interfaz de red eth0. La dirección de origen se cambia a 192.168.10.1:2016 en el caso de PCA1 y a 192.168.10.2:2016 en el caso de PCA2. Esto se reflejará en las tramas de red generadas por estas conexiones salientes y como se puede ver en las capturas 3 y 4 podemos conectarnos desde la red interna a PCC1.

No.	Time	Source	Destination	Protocol	Length	Info
39	67.568656	192.168.10.1	192.168.7.1	SSHv2	214	Client: Encrypted packet (len=148)
40	67.572653	192.168.7.1	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=2126 Ack=1466 Win=19680 Len=0 TSval=3547796 TSecr=3543602
41	67.637626	192.168.7.1	192.168.10.1	SSHv2	102	Server: Encrypted packet (len=36)
42	67.639624	192.168.10.1	192.168.7.1	TCP	66	2016 → 22 [ACK] Seq=1466 Ack=2162 Win=19680 Len=0 TSval=3543672 TSecr=3547860
43	67.639624	192.168.10.1	192.168.7.1	SSHv2	202	Client: Encrypted packet (len=136)
44	67.645621	192.168.7.1	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=2162 Ack=1602 Win=21408 Len=0 TSval=3547869 TSecr=3543673
45	67.806530	192.168.7.1	192.168.10.1	SSHv2	118	Server: Encrypted packet (len=52)
46	67.807514	192.168.10.1	192.168.7.1	SSHv2	526	Client: Encrypted packet (len=460)
47	67.811516	192.168.7.1	192.168.10.1	TCP	66	22 → 2016 [ACK] Seq=2214 Ack=2062 Win=23136 Len=0 TSval=3548034 TSecr=3543841
48	67.821521	192.168.7.1	192.168.10.1	SSHv2	190	Server: Encrypted packet (len=124)
49	67.830517	192.168.7.1	192.168.10.1	SSHv2	166	Server: Encrypted packet (len=100)
50	67.832515	192.168.10.1	192.168.7.1	TCP	66	2016 → 22 [ACK] Seq=2062 Ack=2438 Win=19680 Len=0 TSval=3543864 TSecr=3548044
51	67.883534	192.168.7.1	192.168.10.1	SSHv2	134	Server: Encrypted packet (len=68)
52	67.924814	192.168.10.1	192.168.7.1	TCP	66	2016 → 22 [ACK] Seq=2062 Ack=2506 Win=19680 Len=0 TSval=3543957 TSecr=3548106

Figura 3: SSH desde PCA1 a PCC1 capturado en la lan A.

No.	Time	Source	Destination	Protocol	Length	Info
63	150.852553	192.168.7.10	192.168.7.1	SSHv2	214	Client: Encrypted packet (len=148)
64	150.854553	192.168.7.1	192.168.7.10	TCP	66	22 → 2016 [ACK] Seq=2126 Ack=1466 Win=19680 Len=0 TSval=3547796 TSecr=3543602
65	150.919525	192.168.7.1	192.168.7.10	SSHv2	102	Server: Encrypted packet (len=36)
66	150.924522	192.168.7.10	192.168.7.1	TCP	66	2016 → 22 [ACK] Seq=1466 Ack=2162 Win=19680 Len=0 TSval=3543672 TSecr=3547860
67	150.924522	192.168.7.10	192.168.7.1	SSHv2	202	Client: Encrypted packet (len=136)
68	150.927521	192.168.7.1	192.168.7.10	TCP	66	22 → 2016 [ACK] Seq=2162 Ack=1602 Win=21408 Len=0 TSval=3547869 TSecr=3543673
69	151.088428	192.168.7.1	192.168.7.10	SSHv2	118	Server: Encrypted packet (len=52)
70	151.091428	192.168.7.10	192.168.7.1	SSHv2	526	Client: Encrypted packet (len=460)
71	151.093430	192.168.7.1	192.168.7.10	TCP	66	22 → 2016 [ACK] Seq=2214 Ack=2062 Win=23136 Len=0 TSval=3548034 TSecr=3543841
72	151.103421	192.168.7.1	192.168.7.10	SSHv2	190	Server: Encrypted packet (len=124)
73	151.111416	192.168.7.1	192.168.7.10	SSHv2	166	Server: Encrypted packet (len=100)
74	151.117413	192.168.7.10	192.168.7.1	TCP	66	2016 → 22 [ACK] Seq=2062 Ack=2438 Win=19680 Len=0 TSval=3543864 TSecr=3548044
75	151.165428	192.168.7.1	192.168.7.10	SSHv2	134	Server: Encrypted packet (len=68)
76	151.208728	192.168.7.10	192.168.7.1	TCP	66	2016 → 22 [ACK] Seq=2062 Ack=2506 Win=19680 Len=0 TSval=3543957 TSecr=3548106

Figura 4: SSH desde PCA1 a PCC1 capturado en la lan C.

En la sesión de prácticas no se comprobó qué ocurría si se creaba una conexión desde el mismo puerto cuando había una ya establecida, pero podemos simularlo con un escenario similar creado con las máquinas virtuales de prácticas de la asignatura Seguridad Informática (figura 5).

La problemática se refleja en la tabla 2. En esta tabla, la máquina encargada de enrutar las conexiones no tiene información sobre la máquina interna de destino, a la que pertenece la conexión, dado que las reglas nat están configuradas en las máquinas cliente.

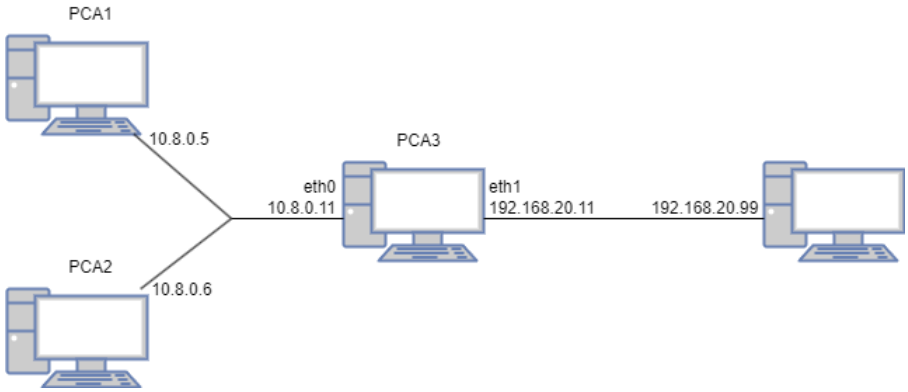

Figura 5: Escenario simulado para el experimento con SSH.

IP externa	Puerto externo	IP interna	Puerto interno
192.168.7.10	2016	192.168.10.1	2016
192.168.7.10	2016	192.168.10.2	2016

Los mensajes de respuesta de la máquina PCC1 los recibe el NAT, pero en ambas conexiones la dirección destino y el puerto coincidirían (192.168.7.10:2016). No obstante, podemos comprobar que esto no es lo que ocurre y que el NAT encuentra una solución al conflicto de puertos:

En el experimento se conecta primero por SSH desde la máquina PCA1. En las capturas de ambas redes podemos ver que la conexión se crea con puerto origen 2016 (observar paquetes 1-3 relativos al handshake TCP de ambas capturas). En segundo lugar se conecta desde PCA2 y la conexión se crea también con puerto origen 2016, pero sólo se aprecia en la red A (paquetes 47-49). En la red C se observa que el NAT ha intervenido, y que sustituye el puerto origen de la conexión por uno aleatorio (en este caso el 42196).

La tabla de traducción quedaría de esta forma:

IP externa	Puerto externo	IP interna	Puerto interno
192.168.7.10	2016	192.168.10.1	2016
192.168.7.10	42196	192.168.10.2	2016

Podemos concluir que el NAT no sólo enmascara las direcciones de la red privada, sino que también enmascara los puertos utilizados.

03

Pregunta 3

Conéctate con el servidor ftp de PCC1. 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.

a) Sin configuraciones adicionales en el NAT, comprobaremos la diferencia de funcionamiento de la conexión de datos ftp tanto en modo activo como pasivo (Para comprobar la diferencia lo haremos mediante el comando ls, que visualiza el contenido del directorio en el servidor remoto). ¿Qué modo de ftp funciona correctamente? Captura en las interfaces oportunas. Observa tanto el nivel IP como los mensajes enviados desde la aplicación ftp. Explica lo sucedido.

b) Incorpora el módulo ip_nat_ftp. Vuelve a realizar la conexión (tanto en modo activo como pasivo) y comprueba que funciona en ambos casos. Captura en las interfaces oportunas. Observa tanto el nivel IP como los mensajes enviados desde la aplicación ftp. Explica nuevamente lo sucedido

---

NAT sin configuraciones adicionales

Capturando en las redes lan A y C ejecutamos nos conectamos con el cliente ftp desde PCA1 a PCC1:

```
[root@localhost ~]# ftp 192.168.7.1
Connected to 192.168.7.1 (192.168.7.1).
220 (vsFTPd 2.2.2)
Name (192.168.7.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
```

En la captura de la red A se pueden identificar los paquetes FTP correspondientes a cada mensaje, ya que la herramienta muestra por pantalla el contenido de los datos enviados sin tratar (son cadenas de caracteres). Los paquetes 1 al 3 corresponden al handshake TCP, el 4 es el saludo del servidor, en los paquetes 8 y 12 se envían las credenciales (mensajes USER y PASS) y el mensaje del paquete 16 indica el sistema operativo del servidor. En la captura de la red C aparecen los mismos mensajes en los paquetes 9 a 28, con la dirección de PCA1 enmascarada por el nat.

Una vez establecida la conexión ejecutamos el comando ls:

```
ftp> ls
227 Entering Passive Mode (192,168,7,1,204,182).
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Mar 22  2017 pub
226 Directory send OK.
```

Este comando requiere un envío de datos, que se produce por medio de una conexión tcp exclusiva. Por defecto, el cliente actúa en modo pasivo, lo que significa que es él quien tiene que solicitar la conexión, no sin antes indicar el servidor qué puerto ha abierto. Esto se ve reflejado en el paquete 18 de la red A, en el que se envía el mensaje PASV, que provoca la respuesta del servidor, que envía la dirección IP y el puerto de la conexión de datos como una cadena de texto de 6 números de 1 byte (4 de dirección y 2 de puerto).

Los paquetes 21 al 23 son el handshake de esta conexión. Una vez establecida, el cliente envía el mensaje LIST en el paquete 24, que le indica al servidor ejecutar ls y devolver la salida. Una vez enviados los datos la conexión se cierra (paquetes 27 a 30).

Cambiamos a modo activo con el comando passive y volvemos a ejecutar ls:

```
ftp> passive
Passive mode off.
ftp> ls
500 Illegal PORT command.
```

Ahora es el servidor el que tendrá que establecer la conexión de datos, por lo que el cliente envía el mensaje PORT, indicando su ip y el puerto que ha abierto con el mismo formato de cadena de caracteres de antes (paquete 35 de la red A). En este caso el servidor indica con el código 500 que hay un problema en su lado: el cliente había indicado que se debía conectar a su dirección ip (192.168.10.1), la cual corresponde a una red privada a la que el servidor no sabe cómo acceder. En la captura de la red C podemos comprobar en el paquete 103 que al servidor le ha llegado exactamente la dirección enviada por PCA1.

En este primer caso, el NAT sólo permitía la conexión al exterior de PCA1, enmascarando su dirección en la cabecera IP. Sin embargo, sin la ayuda de un módulo externo no puede sustituir la dirección si esta se encuentra dentro de los datos.

NAT con módulo para conexiones FTP

En este segundo caso hemos incluído el módulo ip_nat_ftp en la máquina PCA3 y hemos vuelto a ejecutar el experimento:

```
[root@localhost ~]# ftp 192.168.7.1
Connected to 192.168.7.1 (192.168.7.1).
220 (vsFTPd 2.2.2)
Name (192.168.7.1:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,7,1,50,217).
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Mar 22  2017 pub
226 Directory send OK.
```

Los paquetes 51 a 65 de la red A son los correspondientes a la conexión, el envío de credenciales e identificación del sistema operativo. Al igual que en el caso anterior, el cliente actúa en modo pasivo por defecto (paquete 66), y no hay problema en establecer la conexión de datos porque es el cliente el que se conecta a una dirección visible (el paquete 69 es el inicio de la conexión, el paquete 79 es el último ACK de cierre).

Volvemos a ejecutar ls en modo activo:

```
ftp> passive
Passive mode off.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0        0            4096 Mar 22  2017 pub
226 Directory send OK.
```

La salida de consola indica que esta vez ha funcionado. Si comprobamos el paquete 82 (el que envía el mensaje PORT en la red A), vemos que la dirección indicada para que se conecte el servidor sigue siendo la de PCA1 (192.168.10.1). Sin embargo, al servidor ftp no le llega el mismo mensaje enviado. El paquete correspondiente al mensaje PORT en la red C es el número 192, y no indica conectarse a la dirección privada de PCA1, sino a la dirección del NAT (PCA3) en la red C (192.168.7.10), lo que significa que el NAT ha sido capaz de enmascarar la dirección dentro de los datos de la capa de aplicación. Como esa dirección es visible para PCC1, el servidor puede crear la conexión de datos conectándose al NAT: el paquete 195 de la red C envía un TCP SYN a PCA3 por el puerto que indicaba el mensaje PORT ($152 * 2^8 + 156 = 39068$), y es redirigido por la red A (paquete 84) hacia PCA3, por lo que la conexión se establece con el cliente. No obstante, no había ninguna conexión previa

que indicase al NAT que ese puerto debía ser redirigido a PCA1, así que se deduce que el módulo ip_nat_ftp no solo enmascara los datos del mensaje PORT, sino que también crea esa asociación en la tabla NAT.

04

Pregunta 4

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

Con el propósito de habilitar conexiones entrantes a la subred interna B mediante el protocolo SSH, se ha configurado un NAT estático en la máquina PCB3. Esta configuración se ha conseguido al aplicar la siguiente regla mediante la utilidad iptables:

```
# En PCB3
iptables -t nat -A PREROUTING -i eth0 -p tcp -dport 22 -j DNAT --to 192.168.20.1
```

Esta regla se encarga de redirigir el tráfico TCP con destino al puerto 22 en la maquina donde se aplica (PCB3) hacia la dirección IP 192.168.20.1 a través de la interfaz de red eth0, permitiendo así el acceso a la subred interna B a través del protocolo SSH.

Parámetro	**Descripción**
-t nat	Tabla de filtrado NAT para modificar paquetes de Network Address Translation (NAT).
-A PREROUTING	Indica que la regla se debe aplicar en la cadena ”PREROUTING”de la tabla ”nat”. La cadena PREROUTING se utiliza para modificar paquetes antes de que se tome cualquier decisión de enrutamiento.
-i eth0	Especifica la interfaz de red de entrada (eth0) a través de la cual se aplicará la regla.
-p tcp	Protocolo del paquete a comprobar.
-dport 22	Puerto de destino.
-j DNAT –to 192.168.20.1	Establece que si un paquete coincide con la regla, se aplicará la acción DNAT (destination nat), la cual realizará una traducción de la dirección de destino. En este caso, los paquetes que cumplan con las condiciones anteriores se redirigirán a la dirección IP 192.168.20.1.

05

Pregunta 5

Tomando como base las capturas, verifica y demuestra el funcionamiento de la conexión al servicio ssh.

Para verificar el servicio ssh se ha realizado una conexión desde PCA1 hacia PCB1 mientras capturamos en las redes LAN A Y LAN C. Los paquetes 7 a 9 de la red A confirman que la regla del nat aplicada en el ejercicio anterior funciona, ya que el handshake de la conexión tcp con destino 192.168.7.20:20 es satisfactorio. El resto del tráfico (figura 6) confirma que se trata del servidor ssh, ya que hace uso del protocolo SSHv2.

7	5.739544	192.168.10.1	192.168.7.20	TCP	74 2016 → 22 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM TSval=4163204 TSecr=0 WS=32
8	5.755117	192.168.7.20	192.168.10.1	TCP	74 22 → 2016 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM TSval=4160942 TSecr=4163204 WS=32
9	5.757115	192.168.10.1	192.168.7.20	TCP	66 2016 → 22 [ACK] Seq=1 Ack=1 Win=14624 Len=0 TSval=4163221 TSecr=4160942
10	6.363739	192.168.7.20	192.168.10.1	SSHv2	87 Server: Protocol (SSH-2.0-OpenSSH_5.3)
11	6.366737	192.168.10.1	192.168.7.20	TCP	66 2016 → 22 [ACK] Seq=1 Ack=22 Win=14624 Len=0 TSval=4163829 TSecr=4161554
12	6.366737	192.168.10.1	192.168.7.20	SSHv2	87 Client: Protocol (SSH-2.0-OpenSSH_5.3)
13	6.372804	192.168.7.20	192.168.10.1	TCP	66 22 → 2016 [ACK] Seq=22 Ack=22 Win=14496 Len=0 TSval=4161561 TSecr=4163830
14	6.373799	192.168.10.1	192.168.7.20	SSHv2	930 Client: Key Exchange Init
15	6.382780	192.168.7.20	192.168.10.1	TCP	66 22 → 2016 [ACK] Seq=22 Ack=886 Win=16224 Len=0 TSval=4161572 TSecr=4163837
16	6.435749	192.168.7.20	192.168.10.1	SSHv2	906 Server: Key Exchange Init
17	6.437748	192.168.10.1	192.168.7.20	SSHv2	90 Client: Diffie-Hellman Group Exchange Request
18	6.441759	192.168.7.20	192.168.10.1	TCP	66 22 → 2016 [ACK] Seq=862 Ack=910 Win=16224 Len=0 TSval=4161631 TSecr=4163900
19	6.465734	192.168.7.20	192.168.10.1	SSHv2	346 Server: Diffie-Hellman Group Exchange Group
20	6.477725	192.168.10.1	192.168.7.20	SSHv2	338 Client: Diffie-Hellman Group Exchange Init
21	6.509717	192.168.7.20	192.168.10.1	SSHv2	914 Server: Diffie-Hellman Group Exchange Reply, New Keys
22	6.551306	192.168.10.1	192.168.7.20	TCP	66 2016 → 22 [ACK] Seq=1182 Ack=1990 Win=19680 Len=0 TSval=4164014 TSecr=4161699
23	8.255388	192.168.10.1	192.168.7.20	SSHv2	82 Client: New Keys
24	8.298382	192.168.7.20	192.168.10.1	TCP	66 22 → 2016 [ACK] Seq=1990 Ack=1198 Win=17952 Len=0 TSval=4163483 TSecr=4165713
25	8.299622	192.168.10.1	192.168.7.20	SSHv2	118 Client: Encrypted packet (len=52)
26	8.304569	192.168.7.20	192.168.10.1	TCP	66 22 → 2016 [ACK] Seq=1990 Ack=1250 Win=17952 Len=0 TSval=4163488 TSecr=4165758

Figura 6: Tráfico en la LAN A

En este experimento se puede ver el trabajo de dos nat:

- Los paquetes originarios de PCA1 aparecen en la red A con su dirección privada (192.168.10.1), mientras que en las redes C y B son enmascarados con la dirección pública de PCA3 (192.168.7.10). Esto se puede ver en el paquete 7 de la red A, que en la red C corresponde al paquete 33, que tiene modificada la dirección origen. Lo mismo ocurre con los paquetes que tienen como destino PCA1, véase el paquete 8 de la red A, que en la red C corresponde al 34, cuyas direcciones destino difieren.
- Los paquetes con destino PCB1 aparecen en la red B con su dirección privada (192.168.20.1), mientras que en las redes C y A son enmascarados con la dirección pública de PCB3 (192.168.7.20). En este caso faltaría una captura en la red B que lo corroborase. Lo que debería verse es que la dirección destino del paquete 7 en la red A fuese la de PCB3, y se viera modificada por la de PCB1 al entrar en la red B. En el paquete 33 de la red C no ocurre porque no ha pasado por PCB3.

06

Pregunta 6

En base a las capturas del tráfico, calcula el throughput (en pps) y el ancho de banda (en bps) total, útil a nivel IP y útil a nivel de aplicación, que se consigue en el escenario propuesto para cada uno los casos de generación de tráfico y compáralos con el dato de ancho de banda dado por la herramienta iperf. Analiza si la limitación aparece en el valor del throughput o el de ancho de banda y si es a causa del generador de tráfico o del router.

Mientras ejecutamos cada uno de los experimentos también capturamos simultáneamente en las redes A y C. Los cálculos que aparecen a continuación han sido realizados en base a las estadísticas de la captura en la red A. La red C sólo ha sido necesaria para comprobar si el NAT actúa como limitante.

Tráfico TCP

Para medir el throughput del envío tcp, aplicamos primero el siguiente filtro con el fin de que wireshark muestre únicamente los paquetes de la conexión abierta por iperf:

```
(ip.src == 192.168.10.1 && ip.dst == 192.168.7.1) ||
(ip.src == 192.168.7.1  && ip.dst == 192.168.10.1)
```

Ignoramos los dos primeros paquetes porque no corresponden al intervalo de medida y accedemos a las estadísticas de la captura: *Estadísticas* > *Propiedades de archivo de captura*, sección *Estadísticas*, columna *Mostrado*:

Paquetes:	35414
Espacio de tiempo:	30.995 s
Promedio tamaño de paquete:	842 B
Throughput (pps):	1142.6 pps
Ancho de banda (bps):	7.695 Mbps

Las fórmulas para calcular tanto el throughput como el ancho de banda son las siguientes:

$$throughput = paquetes/tiempo = 35414/30{,}995 = 1142{,}57\ pps$$

$$ancho\ de\ banda = throughput \cdot tamano\ de\ paquete\ (bytes) \cdot 8 = 1142{,}57 \cdot 842 \cdot 8 = 7696360\ bps = 7{,}693\ Mbps$$

Estos valores corresponden con el throughput y ancho de banda total. Wireshark facilita una herramienta de gráficos en la que se puede ver que el ancho de banda (figura 8) y el throughput (figura 7) se aproximan durante toda la conexión a los valores calculados.

Para calcular el el throughput y ancho de banda útiles hay que considerar únicamente los paquetes TCP que son enviados por el cliente. Además, hay que ignorar los paquetes duplicados y de control, por lo que aplicamos el siguiente filtro:

```
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && !tcp.analysis.flags && ip.len == 1500
```

Con TCP como protocolo de transporte el envío es siempre del máximo tamaño de datagrama IP. En este caso es 1500 por ser el máximo MTU de cualquiera de las máquinas.

De la misma forma que antes obtenemos las estadísticas:

Paquetes:	18862
Espacio de tiempo:	30.995 s
Tamaño de paquete útil nivel ip:	1500 - 20 = 1480 B
Tamaño de paquete útil nivel aplicación:	1480 - 32 = 1448 B
Throughput (pps):	608.55 pps
Ancho de banda útil nivel ip (bps):	7.302 Mbps
Ancho de banda útil nivel aplicación (bps):	7.205 Mbps

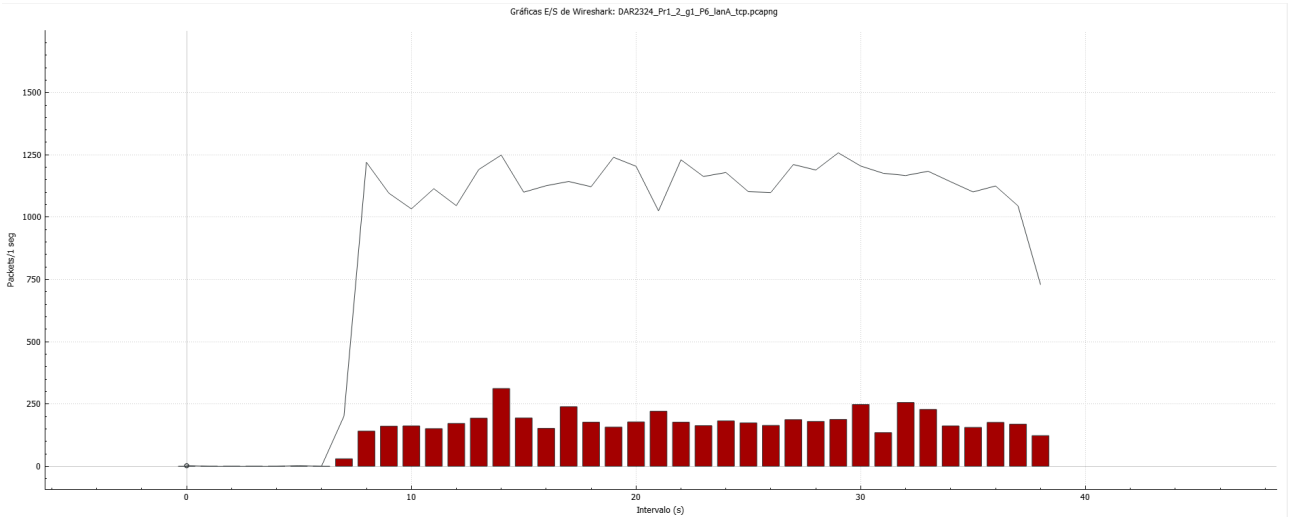

Figura 7: Throughput en pps producido por el envío con tcp

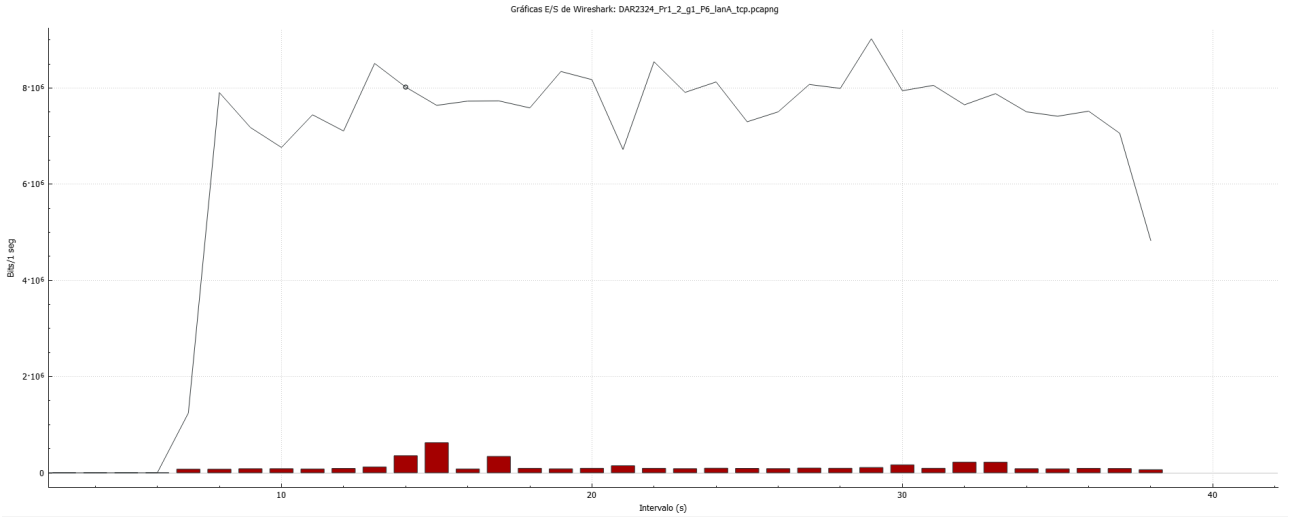

Figura 8: Throughput en pps producido por el envío con tcp

Para el espacio de tiempo hemos considerado el valor anterior, ya que los mensajes de control de tcp sí que influyen en el tiempo de envío. El tamaño de paquete útil a nivel ip es 20 bytes menos que el tamaño del paquete ip. Para el tamaño útil a nivel de aplicación se restan 32 bytes porque es el tamaño de cabecera tcp (con wireshark podemos ver que iperf utiliza el campo opcional de tcp para los timestamps, que añade 12 bytes a la cabecera).

Tráfico UDP

Para los envíos con protocolo udp hemos necesitado de un filtro para cada tamaño de datagrama:

```
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && udp.length == 8 + 100
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && udp.length == 8 + 200
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && udp.length == 8 + 300
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && udp.length == 8 + 600
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && udp.length == 8 + 1200
ip.src == 192.168.10.1 && ip.dst == 192.168.7.1 && udp.length == 8 + 1472
```

Para calcular el throughput y el ancho de banda hemos seguido el mismo método que con tcp, considerando lo siguiente:

- El tamaño útil a nivel de aplicación coincide con el valor introducido con el parámetro -l.
- El tamaño útil a nivel de ip es ese mismo valor sumados los 8 bytes de la cabecera udp.
- El ancho de banda total se ha calculado sumando los 20 bytes de cabecera ip.

Los valores obtenidos se muestran en esta tabla (tiempo en segundos, tamaños en bytes, throughput en pps y anchos de banda en bps):

Paquetes	**Tiempo** (s)	**Tamaño útil**	**Thoughput**	**AB total**	**AB útil ip**	**AB útil app**
6606	10.027	100	658.821	0.674	0.527	0.569
6047	10.017	200	603.673	1.101	0.965	1.004
6410	10.012	300	640.231	1.679	1.536	1.577
5914	10.02	600	590.219	2.965	2.833	2.871
5872	10.03	1200	585.443	5.751	5.620	5.657
5456	10.035	1200	543.697	5.341	5.219	5.254

Lo que muestran los datos es que el throughput disminuye ligeramente a medida que se envían datagramas más largos, pero el ancho de banda aumenta con la misma proporción que el tamaño útil. Es decir, la comunicación no está limitada por la tasa de bits de la red, sino por el tiempo que requiere preparar y enviar cada paquete individual desde el cliente. Esta idea se hace más visible cuando observamos las gráficas de wireshark (figuras 9 y 10).

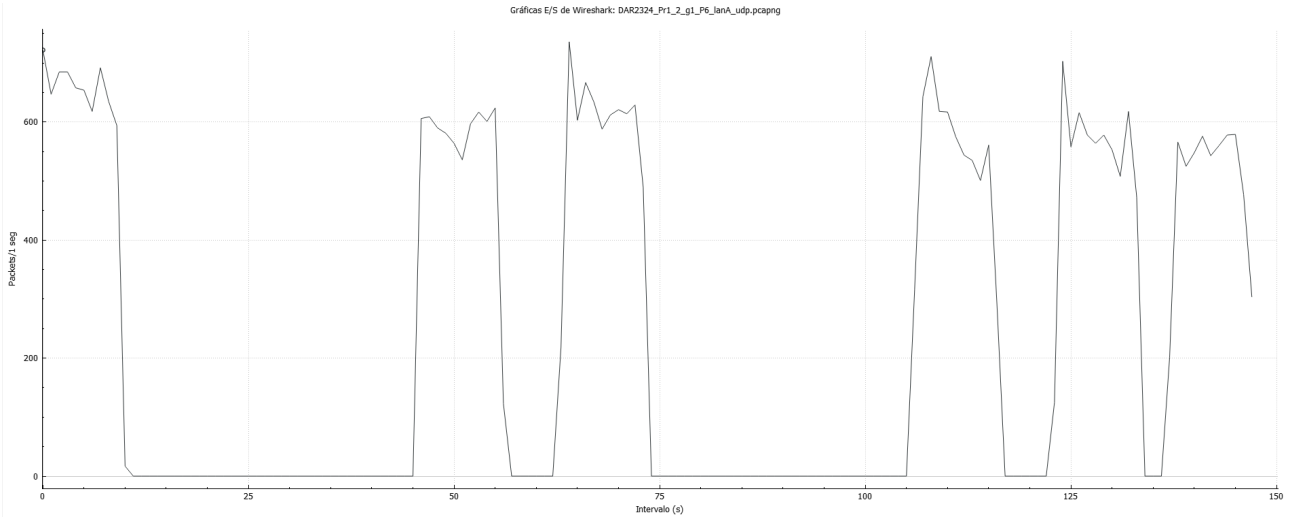

Figura 9: Throughput en pps producido por los envíos con udp. De menor a mayor payload.

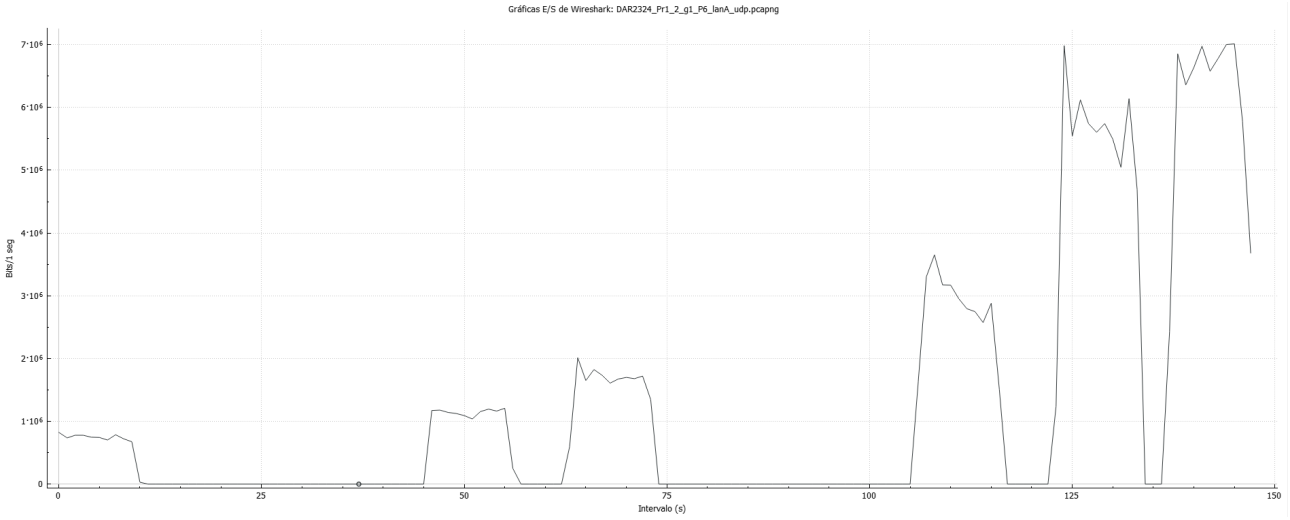

Figura 10: Throughput en pps producido por los envíos con udp. De menor a mayor payload.

Para comprobar que el router no limita el throughput lo que hemos hecho es aplicar cada uno de estos filtros en la captura de la red C y observar el espacio de tiempo de las estadísticas de wireshark.

```
ip.src == 192.168.7.10 && ip.dst == 192.168.7.1 && udp.length == 8 + 100
ip.src == 192.168.7.10 && ip.dst == 192.168.7.1 && udp.length == 8 + 200
ip.src == 192.168.7.10 && ip.dst == 192.168.7.1 && udp.length == 8 + 300
ip.src == 192.168.7.10 && ip.dst == 192.168.7.1 && udp.length == 8 + 600
ip.src == 192.168.7.10 && ip.dst == 192.168.7.1 && udp.length == 8 + 1200
ip.src == 192.168.7.10 && ip.dst == 192.168.7.1 && udp.length == 8 + 1472
```

Como el tiempo total es el mismo y el número de paquetes reenviados por el NAT tiene que ser idéntico a los enviados por PCA1, concluímos que PCA3 no está limitando la comunicación. También se puede ver gráficamente que las medidas son casi idénticas (figuras 11 y 12)

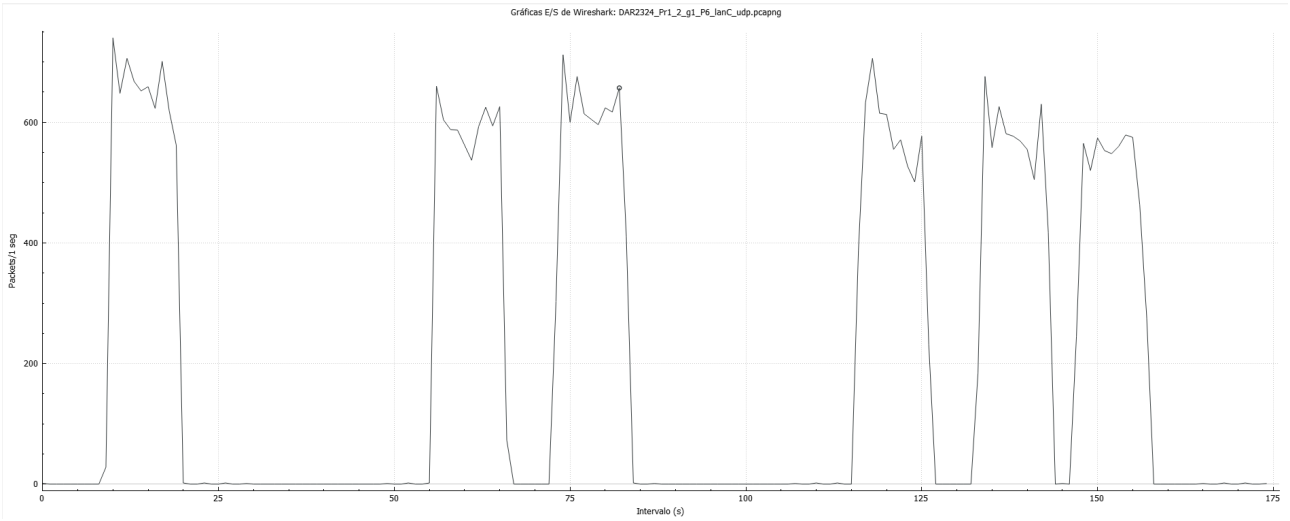

Figura 11: Throughput en pps producido por los envíos con udp en la red C. De menor a mayor payload.

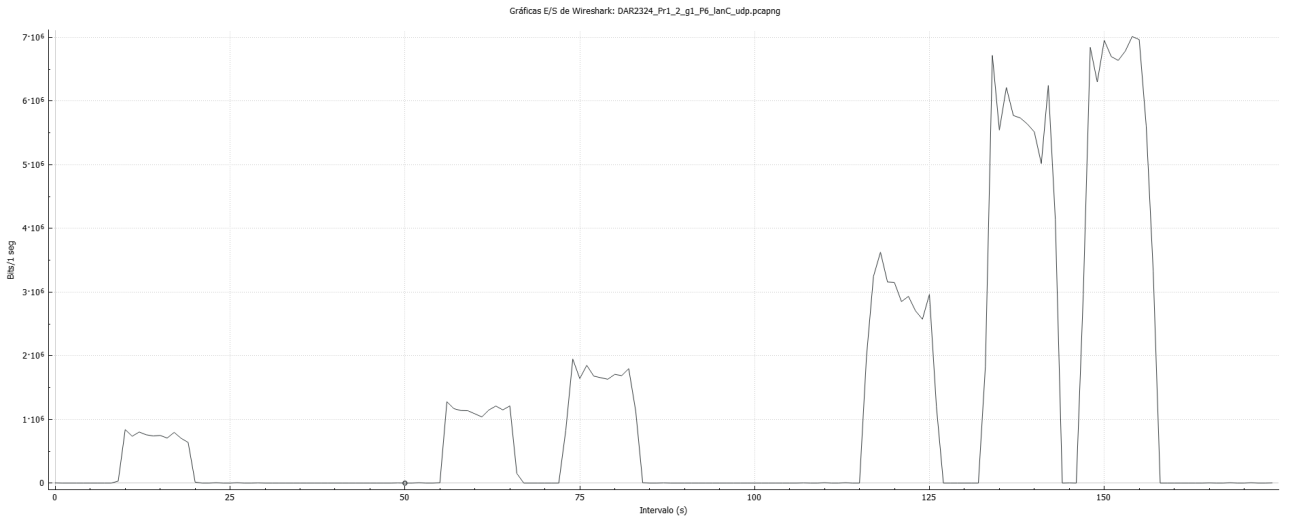

Figura 12: Throughput en pps producido por los envíos con udp en la red C. De menor a mayor payload.