Práctica 1.2. TCP y NAT

Objetivos

En esta práctica estudiaremos el funcionamiento del protocolo TCP. Además, veremos algunos parámetros que permiten ajustar el comportamiento de las aplicaciones TCP. Finalmente, se verá cómo configurar NAT con `iptables`.

Activar el **portapapeles bidireccional** (menú Dispositivos) en las máquinas.

Usar la opción de Virtualbox (menú Ver) para realizar **capturas de pantalla**.

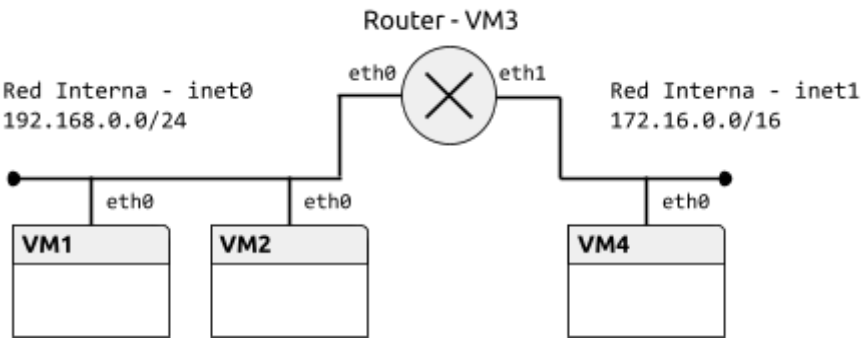
La contraseña del usuario `cursoredes` es `cursoredes`.

Contenidos

- Preparación del entorno para la práctica
- Estados de una conexión TCP
- Introducción a la seguridad en el protocolo TCP
- Opciones y parámetros TCP
- Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)

Preparación del entorno para la práctica

Configuraremos la topología de red que se muestra en la siguiente figura, igual a la empleada en la práctica anterior.



Antes de crear el entorno **eliminar las máquinas virtuales de ASOR de VirtualBox**, junto con todos sus archivos. Después **importar el servidor** usando `/mnt/DiscoVMs/ASOR/ASOR-FE.ova`. Finalmente **crear la topología con `vtopol`**.

El contenido del fichero de configuración de la topología debe ser el siguiente:

```
netprefix inet
machine 1 0 0
machine 2 0 0
machine 3 0 0 1 1
machine 4 0 1
```

Finalmente, configurar la red de todas las máquinas de la red según la siguiente tabla. Después de configurar todas las máquinas, comprobar la conectividad con la orden `ping`.

Máquina	**Dirección IPv4**	**Comentarios**
VM1	`192.168.0.1/24`	Añadir Router como encaminador por defecto
VM2	`192.168.0.2/24`	Añadir Router como encaminador por defecto
Router - VM3	`192.168.0.3/24 (eth0)` `172.16.0.3/16 (eth1)`	Activar el *forwarding* de paquetes
VM4	`172.16.0.4/16`	Añadir Router como encaminador por defecto

Estados de una conexión TCP

En esta parte usaremos la herramienta Netcat, que permite leer y escribir en conexiones de red. Netcat es muy útil para investigar y depurar el comportamiento de la red en la capa de transporte, ya que permite especificar un gran número de los parámetros de la conexión. Además para ver el estado de las conexiones de red usaremos el comando `ss` (similar a `netstat`, pero más moderno y completo).

Ejercicio 1. Consultar las páginas de manual de `nc` y `ss`. En particular, consultar las siguientes opciones de `ss`: `-a`, `-l`, `-n`, `-t` y `-o`. Probar algunas de las opciones para ambos programas para familiarizarse con su comportamiento.

Ejercicio 2. (LISTEN) Abrir un servidor TCP en el puerto 7777 en VM1 usando el comando `nc -l 7777`. Comprobar el estado de la conexión en el servidor con el comando `ss -tln`. Abrir otro servidor en el puerto 7776 en VM1 usando el comando `nc -l 192.168.0.1 7776`. Observar la diferencia entre ambos servidores usando `ss`. Comprobar que no es posible la conexión desde VM1 con `localhost` como dirección destino usando el comando `nc localhost 7776`.

Primer servidor:

```
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      100    127.0.0.1:25                *:*
LISTEN   0      10       *:7777                    *:*
LISTEN   0      128      *:111                     *:*
LISTEN   0      128      *:22                      *:*
LISTEN   0      128    127.0.0.1:631               *:*
LISTEN   0      100     ::1:25                    :::*
LISTEN   0      10      :::7777                   :::*
LISTEN   0      128     :::111                    :::*
LISTEN   0      128     :::22                     :::*
LISTEN   0      128     ::1:631                   :::*
```

Segundo servidor:

```
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      100    127.0.0.1:25                *:*
LISTEN   0      10   192.168.0.1:7776                *:*
LISTEN   0      10       *:7777                    *:*
LISTEN   0      128      *:111                     *:*
LISTEN   0      128      *:22                      *:*
LISTEN   0      128    127.0.0.1:631               *:*
LISTEN   0      100     ::1:25                    :::*
LISTEN   0      10      :::7777                   :::*
LISTEN   0      128     :::111                    :::*
```

LISTEN 0 128 :::22 :::
LISTEN 0 128 ::1:631 :::*

La diferencia es que si no introducimos dirección ip, el servidor puede atarse a cualquier dirección ip. Sin embargo, al hacer nc -l <dir_ip> ya le estamos dando una dirección ip fija al servidor.

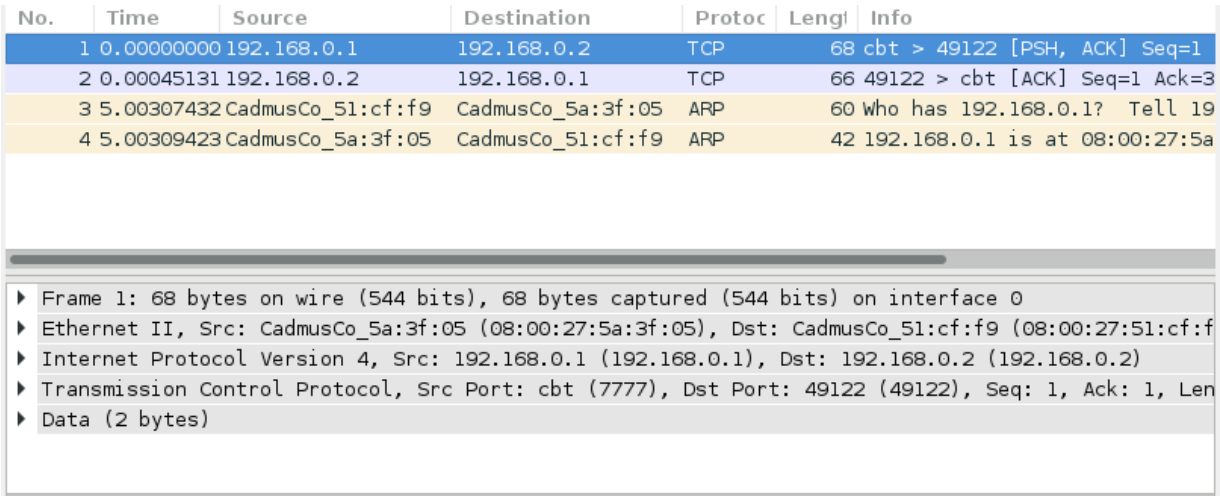
No es posible conectar con localhost:
[cursoredes@localhost ~]$ nc localhost 7776
Ncat: Connection refused.

Ejercicio 3. (ESTABLISHED) En VM2, iniciar una conexión cliente al primer servidor arrancado en el ejercicio anterior usando el comando `nc 192.168.0.1 7777`.

- Comprobar el estado de la conexión e identificar los parámetros (dirección IP y puerto) con el comando `ss -tn`.
- Iniciar una captura con Wireshark. Intercambiar un único carácter con el cliente y observar los mensajes intercambiados (especialmente los números de secuencia, confirmación y flags TCP) y determinar cuántos bytes (y número de mensajes) han sido necesarios.

[cursoredes@localhost ~]$ ss -tn
State Recv-Q Send-Q Local Address:Port Peer Address:Port
ESTAB 0 0 192.168.0.2:49122 192.168.0.1:7777

La conexión se ha establecido correctamente, con puerto del cliente: 49122



Se transmiten 2 bytes (carácter + salto de línea). En el primer TCP del servidor (VM1) al cliente (VM2) se activan los flags PSH (el carácter se consume inmediatamente, sin pasar por el buffer) y ACK. El cliente confirma con ACK = 3 la llegada del mensaje del TCP (de tamaño 2 bytes).

Ejercicio 4. (TIME-WAIT) Cerrar la conexión en el cliente (con `Ctrl+C`) y comprobar el estado de la conexión usando `ss -tan`. Usar la opción `-o` de `ss` para observar el valor del temporizador TIME-WAIT.

En servidor 2 tras cerrar la conexión:

```
[cursoredes@localhost ~]$ ss -tano
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      100    127.0.0.1:25                      *:*
LISTEN     0      128        *:111                         *:*
LISTEN     0      128        *:22                          *:*
LISTEN     0      128    127.0.0.1:631                     *:*
TIME-WAIT  0      0      192.168.0.2:36860                 192.168.0.1:7777      timer:(timewait,54sec,0)
LISTEN     0      100      ::1:25                          :::*
LISTEN     0      128       :::111                         :::*
LISTEN     0      128       :::22                          :::*
LISTEN     0      128      ::1:631                         :::*
```

Ejercicio 5. (SYN-SENT y SYN-RECV) El comando `iptables` permite filtrar paquetes según los flags TCP del segmento con la opción `--tcp-flags` (consultar la página de manual `iptables-extensions`). Usando esta opción:

- Fijar una regla en el servidor (VM1) que bloquee un mensaje del acuerdo TCP de forma que el cliente (VM2) se quede en el estado SYN-SENT. Comprobar el resultado con `ss -tan` en el cliente.
- Borrar la regla anterior y fijar otra en el cliente (VM2) que bloquee un mensaje del acuerdo TCP de forma que el servidor se quede en el estado SYN-RECV. Comprobar el resultado con `ss -tan` en el servidor. Además, esta regla debe dejar al servidor también en el estado LAST-ACK después de cerrar la conexión en el cliente. Usar la opción `-o` de `ss` para determinar cuántas retransmisiones se realizan y con qué frecuencia. Borrar la regla al terminar.

En VM1:

sudo iptables -A OUTPUT -p tcp ! --syn -j REJECT -> (añade regla a cadena OUTPUT tal que todo mensaje que no sea de inicio de conexión no se envía, como por ejemplo los ACK).

En VM2 tras intentar establecer conexión con VM1:

```
[cursoredes@localhost ~]$ ss -tan
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      100    127.0.0.1:25                      *:*
LISTEN     0      128        *:111                         *:*
LISTEN     0      128        *:22                          *:*
LISTEN     0      128    127.0.0.1:631                     *:*
SYN-SENT   0      1      192.168.0.2:36870                 192.168.0.1:7777
LISTEN     0      100      ::1:25                          :::*
LISTEN     0      128       :::111                         :::*
LISTEN     0      128       :::22                          :::*
LISTEN     0      128      ::1:631                         :::*
```

Para borrar la regla:

sudo iptables -D OUTPUT 1

Para la segunda parte:

En VM2:
sudo iptables -A OUTPUT -p tcp --tcp-flags ALL ACK -j DROP -> (prohíbe envíos de ACK)

Así conseguimos que el servidor se quede en dicho estado y cuando cerremos la conexión permanezca en LAST-ACK.

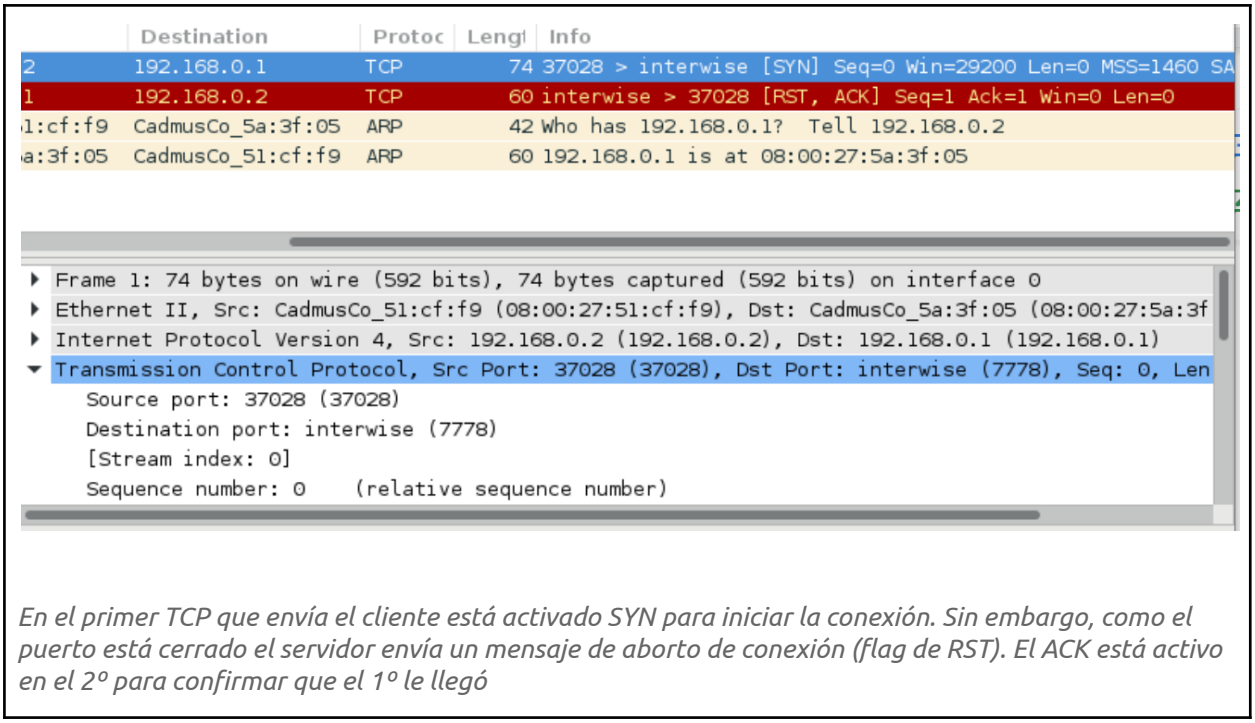
En VM1:

```
[cursoredes@localhost ~]$ ss -tano
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      100    127.0.0.1:25                     *:*
LISTEN     0      10         *:7777                   *:*                 timer:(keepalive,101ms,0)
SYN-RECV   0      0      192.168.0.1:7777             192.168.0.2:34246             timer:(on,4.301ms,3)
LISTEN     0      128        *:111                    *:*
LISTEN     0      128        *:22                     *:*
LISTEN     0      128    127.0.0.1:631                    *:*
LISTEN     0      100       ::1:25                   :::*
LISTEN     0      10        :::7777                  :::*
LISTEN     0      128       :::111                   :::*
LISTEN     0      128       :::22                    :::*
LISTEN     0      128       ::1:631                  :::*
```

Al cerrar la conexión desde el cliente:

```
[cursoredes@localhost ~]$ ss -tano
State      Recv-Q Send-Q Local Address:Port               Peer Address:Port
LISTEN     0      100    127.0.0.1:25                     *:*
LISTEN     0      128        *:111                    *:*
LISTEN     0      128        *:22                     *:*
LISTEN     0      128    127.0.0.1:631                    *:*
LAST-ACK   0      1      192.168.0.1:7777             192.168.0.2:34248             timer:(on,31sec,0)
LISTEN     0      100       ::1:25                   :::*
LISTEN     0      128       :::111                   :::*
LISTEN     0      128       :::22                    :::*
LISTEN     0      128       ::1:631                  :::*
[cursoredes@localhost ~]$
```

Ejercicio 6. Iniciar una captura con Wireshark. Intentar una conexión a un puerto cerrado del servidor (ej. 7778) y observar los mensajes TCP intercambiados, especialmente los flags TCP.



En el primer TCP que envía el cliente está activado SYN para iniciar la conexión. Sin embargo, como el puerto está cerrado el servidor envía un mensaje de aborto de conexión (flag de RST). El ACK está activo en el 2º para confirmar que el 1º le llegó

Introducción a la seguridad en el protocolo TCP

Diferentes aspectos del protocolo TCP pueden aprovecharse para comprometer la seguridad del sistema. En este apartado vamos a estudiar dos: ataques DoS basados en TCP SYN *flood* y técnicas de exploración de puertos.

Ejercicio 7. El ataque TCP SYN *flood* consiste en saturar un servidor mediante el envío masivo de mensajes SYN.

- (Cliente VM2) Para evitar que el atacante responda con un mensaje RST (que liberaría la conexión), bloquear con `iptables` los mensajes SYN+ACK del servidor.
- (Cliente VM2) Usar el comando `hping3` (estudiar la página de manual) para enviar mensajes SYN al puerto 22 del servidor (`ssh`) lo más rápido posible (*flood*).
- (Servidor VM1) Estudiar el comportamiento de la máquina, en términos del número de paquetes recibidos. Comprobar si es posible la conexión al servicio `ssh` desde Router.

Repetir el ejercicio desactivando el mecanismo SYN *cookies* en el servidor con el comando `sysctl` (parámetro `net.ipv4.tcp_syncookies`).

Parte 1:

1.Para bloquear mensajes SYN + ACK:

sudo iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -s 192.168.0.1 -j DROP

2.Para enviar mensajes lo más rápido posible:

hping3 -p 22 -S --flood 192.168.0.1

3.Nº de paquetes recibidos y comprobación de conexión desde Router:

En Router: sí se estableció la conexión
[cursoredes@localhost ~]$ nc 192.168.0.1 22
SSH-2.0-OpenSSH_7.4

En VM2: Se transmitieron 121916 paquetes
--- 192.168.0.1 hping statistic ---
121916 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

Parte2:

1. *Para desactivar las cookies:*

   sudo sysctl net.ipv4.tcp_syncookies=0

2. *El resto igual salvo que en este caso el Router no se pudo conectar al puerto ssh del servidor:*
   [cursoredes@localhost ~]$ nc 192.168.0.1 22
   Ncat: Connection timed out

--- 192.168.0.1 hping statistic ---
1176771 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

Nota: Wireshark no debe estar activo cuando se envían paquetes lo más rápido posible (*flooding*).

Ejercicio 8. (Técnica CONNECT) Netcat permite explorar puertos usando la técnica CONNECT que intenta establecer una conexión a un puerto determinado. En función de la respuesta (SYN+ACK o RST), es posible determinar si hay un proceso escuchando.

- (Servidor VM1) Abrir un servidor en el puerto 7777.
- (Cliente VM2) Explorar, de uno en uno, el rango de puertos 7775-7780 usando nc, en este caso usar las opciones de exploración (-z) y de salida detallada (-v).
- Con ayuda de Wireshark, observar los paquetes intercambiados.

[cursoredes@localhost ~]$ nc -zv 192.168.0.1 7775
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -zv 192.168.0.1 7776
Ncat: Version 7.50 (https://nmap.org/ncat)

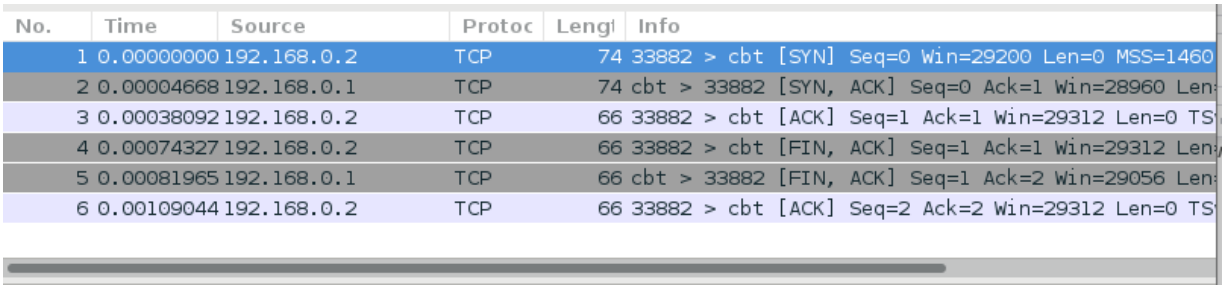
Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -zv 192.168.0.1 7777
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connected to 192.168.0.1:7777.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.

[cursoredes@localhost ~]$ nc -zv 192.168.0.1 7778
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -zv 192.168.0.1 7779
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.

[cursoredes@localhost ~]$ nc -zv 192.168.0.1 7780
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.



Se observa una finalización 3-way.

Opcional. La herramienta Nmap permite realizar diferentes tipos de exploración de puertos, que emplean estrategias más eficientes. Estas estrategias (SYN *stealth*, ACK *stealth*, FIN-ACK *stealth*…) se basan en el funcionamiento del protocolo TCP. Estudiar la página de manual de `nmap` (PORT SCANNING TECHNIQUES) y emplearlas para explorar los puertos del servidor. Comprobar con Wireshark los mensajes intercambiados.

Opciones y parámetros de TCP

El comportamiento de la conexión TCP se puede controlar con varias opciones que se incluyen en la cabecera en los mensajes SYN y que son configurables en el sistema operativo por medio de parámetros del kernel.

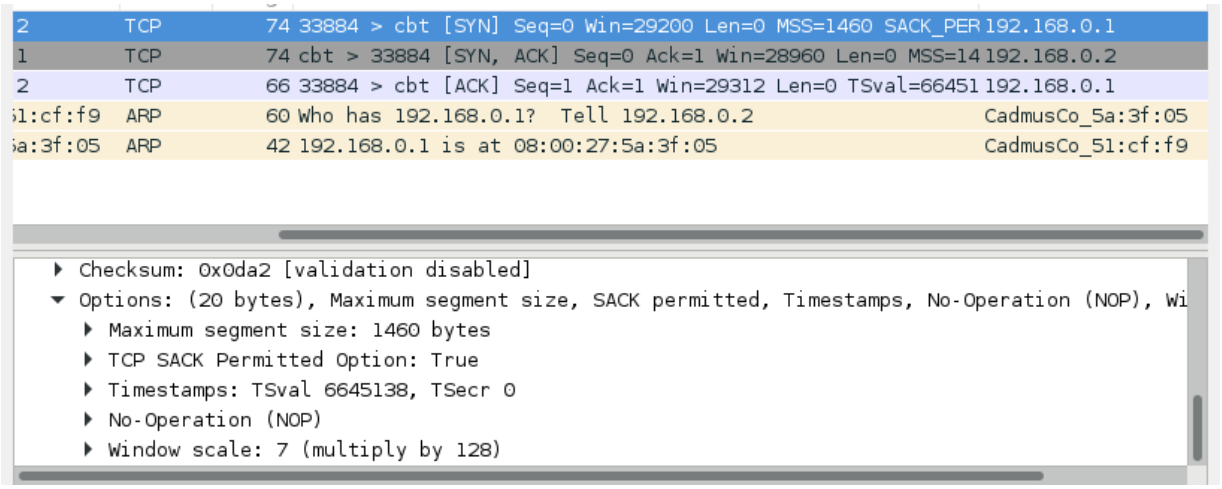
Ejercicio 9. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten modificar algunas opciones de TCP:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_window_scaling`	Permite aumentar el tamaño máximo de la ventana de recepción (que es 64kBytes)	1
`net.ipv4.tcp_timestamps`	Marca de tiempo que se usa para 2 métodos distintos :RTTM (Medición de	1

	tiempo de ida y vuelta) y PAWS (Protección contra secuencias envueltas)	
net.ipv4.tcp_sack	Activar los ACK selectivos	1

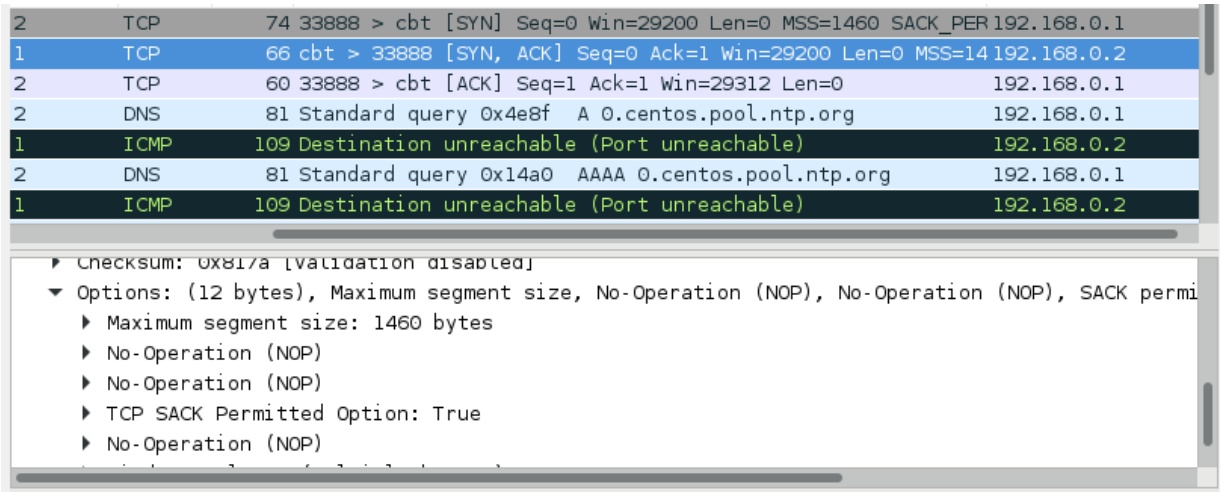
Ejercicio 10. Iniciar una captura de Wireshark. Abrir el servidor en el puerto 7777 y realizar una conexión desde la VM cliente. Estudiar el valor de las opciones que se intercambian durante la conexión. Variar algunos de los parámetros anteriores (ej. no usar ACKs selectivos) y observar el resultado en una nueva conexión.

Sin variación de parámetros:

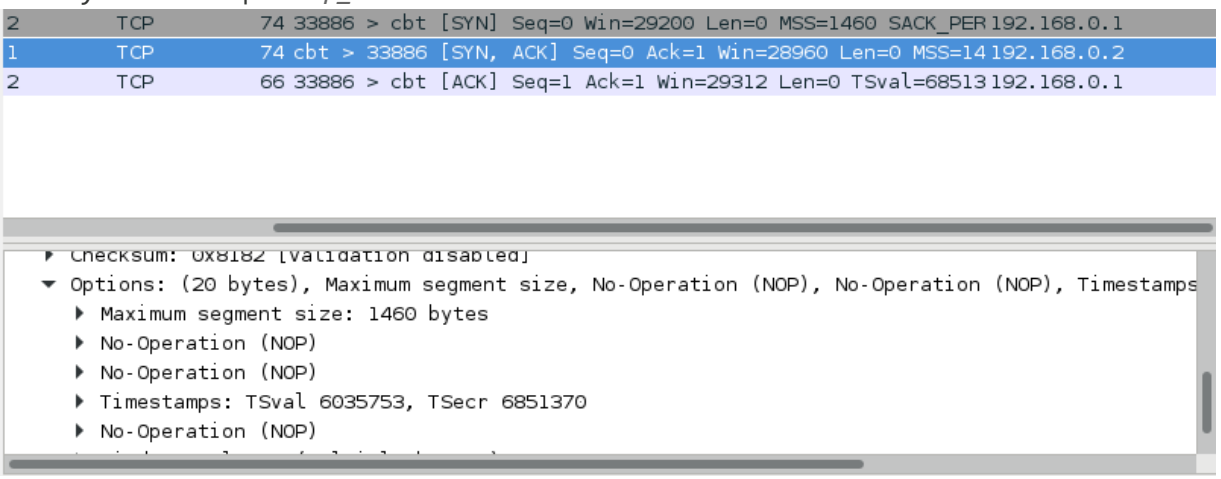


Variando `tcp_timestamps` del servidor:

```
sudo sysctl net.ipv4.tcp_timestamps = 0
```



Variando el tcp_sack en servidor:
`sudo sysctl net.ipv4.tcp_sack = 0`



Ejercicio 11. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten configurar el temporizador *keepalive*:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_keepalive_time`	El intervalo(en segundos) para que TCP envíe mensajes de detección keepalive. Se utiliza para confirmar si la conexión TCP es válida.	7200
`net.ipv4.tcp_keepalive_probes`	Nº de mensajes de sondeo que se envían antes de determinar que la conexión TCP no es válida. Se envían tras agotarse el tiempo keepalive_time.	9
`net.ipv4.tcp_keepalive_intvl`	Intervalo de tiempo (en segundos) para reenviar el mensaje de cada prueba de keealive_probes	75

Traducción de direcciones (NAT) y reenvío de puertos (*port forwarding*)

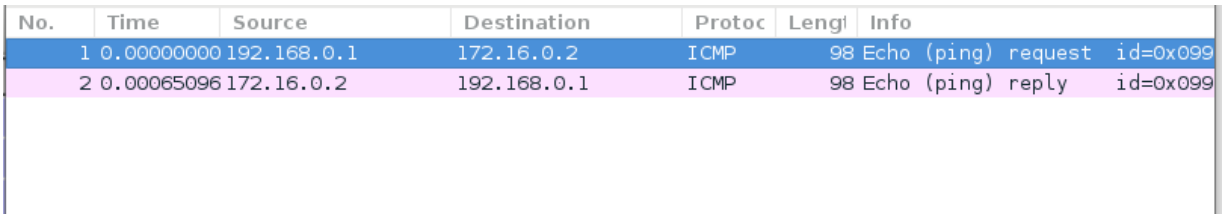
En esta sección supondremos que la red que conecta Router con VM4 es pública y que no puede encaminar el tráfico `192.168.0.0/24`. Además, asumiremos que la dirección IP de Router es dinámica.

Ejercicio 12. Configurar la traducción de direcciones dinámica en Router:

- (Router) Usando `iptables`, configurar Router para que haga SNAT (*masquerade*) sobre la interfaz `eth1`. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Comprobar la conexión con VM4 usando la orden `ping`.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes

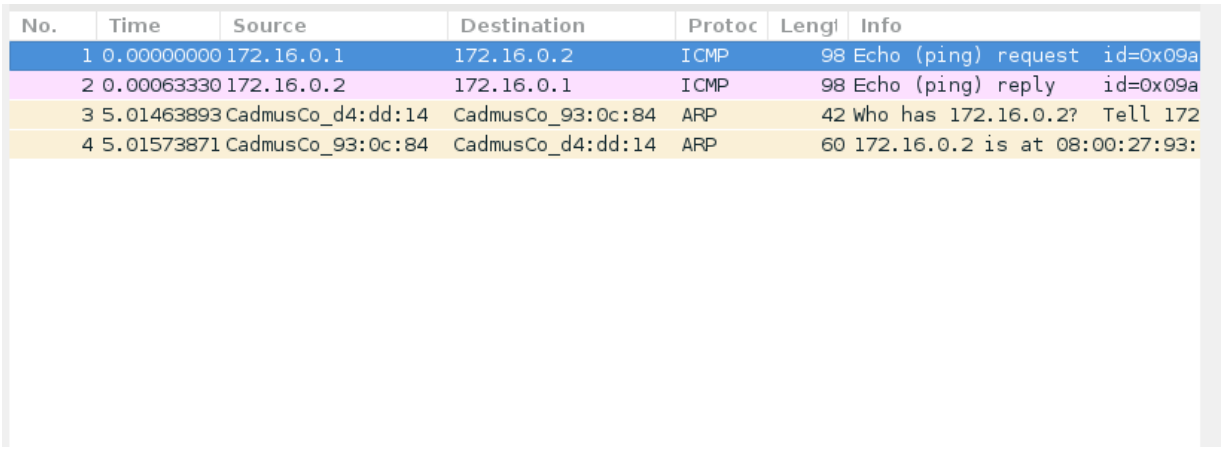
En Router hemos utilizado:
sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

Interfaz eth0:



No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	192.168.0.1	172.16.0.2	ICMP	98	Echo (ping) request id=0x099
2	0.00065096	172.16.0.2	192.168.0.1	ICMP	98	Echo (ping) reply id=0x099

Interfaz eth1:



No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	172.16.0.1	172.16.0.2	ICMP	98	Echo (ping) request id=0x09a
2	0.00063330	172.16.0.2	172.16.0.1	ICMP	98	Echo (ping) reply id=0x09a
3	5.01463893	CadmusCo_d4:dd:14	CadmusCo_93:0c:84	ARP	42	Who has 172.16.0.2? Tell 172
4	5.01573871	CadmusCo_93:0c:84	CadmusCo_d4:dd:14	ARP	60	172.16.0.2 is at 08:00:27:93:

Ejercicio 13. Comprueba la salida del comando `conntrack -L` o, alternativamente, el contenido del fichero `/proc/net/nf_conntrack` en Router mientras se ejecuta el `ping` del ejercicio anterior. ¿Qué parámetro se utiliza, en lugar del puerto origen, para relacionar las solicitudes con las respuestas?

Adjuntar la salida del comando conntrack y responder a la pregunta.

[cursoredes@localhost ~]$ sudo conntrack -L
icmp 1 29 src=192.168.0.1 dst=172.16.0.2 type=8 code=0 id=2700 src=172.16.0.2 dst=172.16.0.1 type=0 code=0 id=2700 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.

Ejercicio 14. Acceso a un servidor en la red privada:

- (Router) Usando `iptables`, reenviar las conexiones (DNAT) del puerto 80 de Router al puerto 7777 de VM1. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Arrancar el servidor en el puerto 7777 con nc.
- (VM4) Conectarse al puerto 80 de Router con nc y comprobar el resultado en VM1.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente los puertos y direcciones IP origen y destino en ambas redes.

Adjuntar el comando iptables utilizado y capturas de pantalla de Wireshark.

En Router:
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to 192.168.0.1:7777

En VM1:
Nc -l 7777

En VM4:
nc 172.16.0.1 80

En Wireshark capturo a la vez tanto interfaz 0 como interfaz 1

No.	Time	Source	Destination	Protoc	Lengt	Info
1	0.00000000	172.16.0.2	172.16.0.1	TCP	74	36282 > http [SYN] Seq=0 Win=
2	0.00006520	172.16.0.2	192.168.0.1	TCP	74	36282 > cbt [SYN] Seq=0 Win=2
3	0.00072209	192.168.0.1	172.16.0.2	TCP	74	cbt > 36282 [SYN, ACK] Seq=0
4	0.00074685	172.16.0.1	172.16.0.2	TCP	74	http > 36282 [SYN, ACK] Seq=0
5	0.00138568	172.16.0.2	192.168.0.1	TCP	66	36282 > cbt [ACK] Seq=1 Ack=1
6	0.00136859	172.16.0.2	172.16.0.1	TCP	66	36282 > http [ACK] Seq=1 Ack=
7	5.00783128	CadmusCo_93:0c:84	CadmusCo_d4:dd:14	ARP	60	Who has 172.16.0.1? Tell 172
8	5.00787913	CadmusCo_d4:dd:14	CadmusCo_93:0c:84	ARP	42	172.16.0.1 is at 08:00:27:d4:
9	5.00994013	CadmusCo_5a:3f:05	CadmusCo_6d:33:25	ARP	60	Who has 192.168.0.3? Tell 19
10	5.00998427	CadmusCo_6d:33:25	CadmusCo_5a:3f:05	ARP	42	192.168.0.3 is at 08:00:27:6d

Se observa como el router redirige las conexiones de su puerto 80 al 7777 de VM1 pues en el segundo TCP, que se corresponde con la interfaz 0 tenemos origen VM4 y destino VM1