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**.

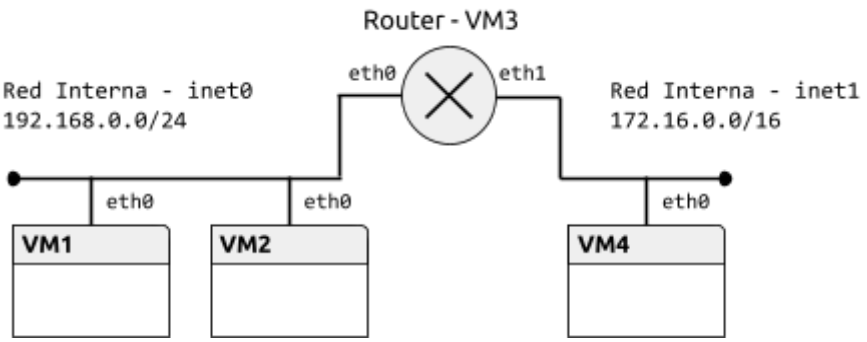
La contraseña del usuario `cursoredes` es `cursoredes`.

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.



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`.

```
[VM1]
nc -l 7777

ss -tln
State    Recv-Q   Send-Q      Local Address:Port          Peer Address:Port
LISTEN   0          100         127.0.0.1:25                *:*
LISTEN   0          10            *:7777                    *:*

nc -l 192.168.0.1 7776

ss -tln
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                  *:*

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`.

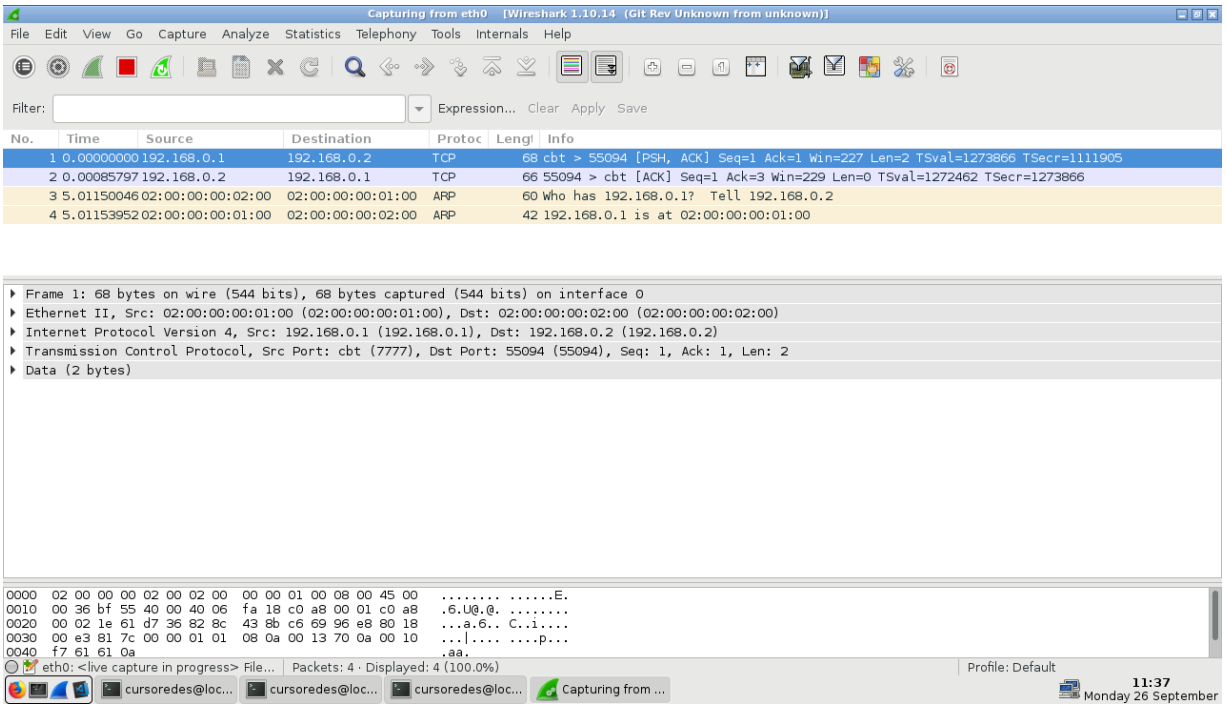
- Comprobar el estado de la conexión e identificar los parámetros (dirección IP y puerto) con el comando `ss -tn`.
- Iniciar una captura con Wireshark. Intercambiar un único carácter con el cliente y observar los mensajes intercambiados (especialmente los números de secuencia, confirmación y flags TCP)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

determinar cuántos bytes (y número de mensajes) han sido necesarios.

[VM2]
Nc 192.168.0.1 7777

ss -tn
State Recv-Q Send-Q Local Address:Port Peer Address:Port
ESTAB 0 0 192.168.0.2:55094 192.168.0.1:7777

Entre los cuatro mensajes han sido necesarios 236 bytes. Los números de secuencia son Seq = 1 y los de confirmación son ACK = 1 y ACK = 3. El resto de flags se ven en la captura.



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.

[VM2]

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 0 192.168.0.2:55100 192.168.0.1:7777 timer:(timewait,55sec,0)
*LISTEN 0 100 ::1:25 :::**
*LISTEN 0 128 :::111 :::**
*LISTEN 0 128 :::22 :::**
*LISTEN 0 128 ::1:631 :::**

$ 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            192.168.0.2:55100          192.168.0.1:7777          timer:(timewait,49sec,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.

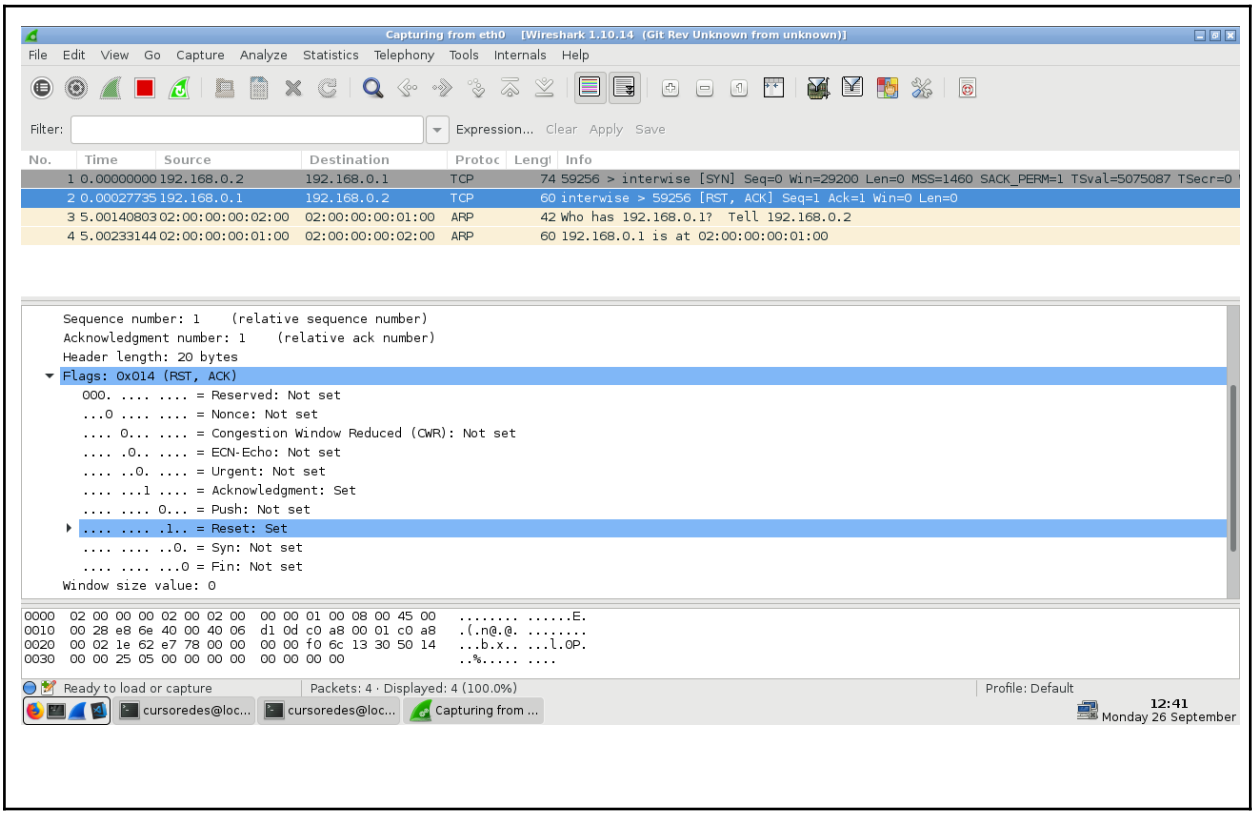
```
[VM1]
sudo iptables -A INPUT -p  tcp  --tcp-flags SYN,ACK,FIN,RST  SYN -j DROP

[VM2]
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      192.168.0.2:55116          192.168.0.1:7777

[VM2]
sudo iptables -A INPUT -p  tcp  --tcp-flags SYN,ACK,FIN,RST  SYN,ACK -j DROP

[VM1]
ss -tan
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                 *:*
SYN-RECV   0      0          192.168.0.1:7777            192.168.0.2:55120
```

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.



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`).

Copiar los comandos iptables y hping3 utilizados. Describir el comportamiento de la máquina con y sin el mecanismo SYN cookies.

[VM2]
sudo iptables -A INPUT -p tcp --tcp-flags ALL SYN,ACK -j DROP
sudo hping3 --flood -p 22 -S 192.168.0.1
HPING 192.168.0.1 (eth0 192.168.0.1): S set, 40 headers + 0 data bytes

hping in flood mode, no replies will be shown
^C
--- 192.168.0.1 hping statistic ---
2236540 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
Se conecta sin problemas al servicio ssh desde Router.
[VM3]
nc 192.168.0.1 22
SSH-2.0-OpenSSH_7.4

Desactivando el mecanismo SYN cookies en el servidor:

[VM1]
sudo sysctl net.ipv4.tcp_syncookies=0

[VM2]
sudo hping3 --flood -p 22 -S 192.168.0.1

[VM3]
nc 192.168.0.1 22
Ncat: Connection timed out.

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

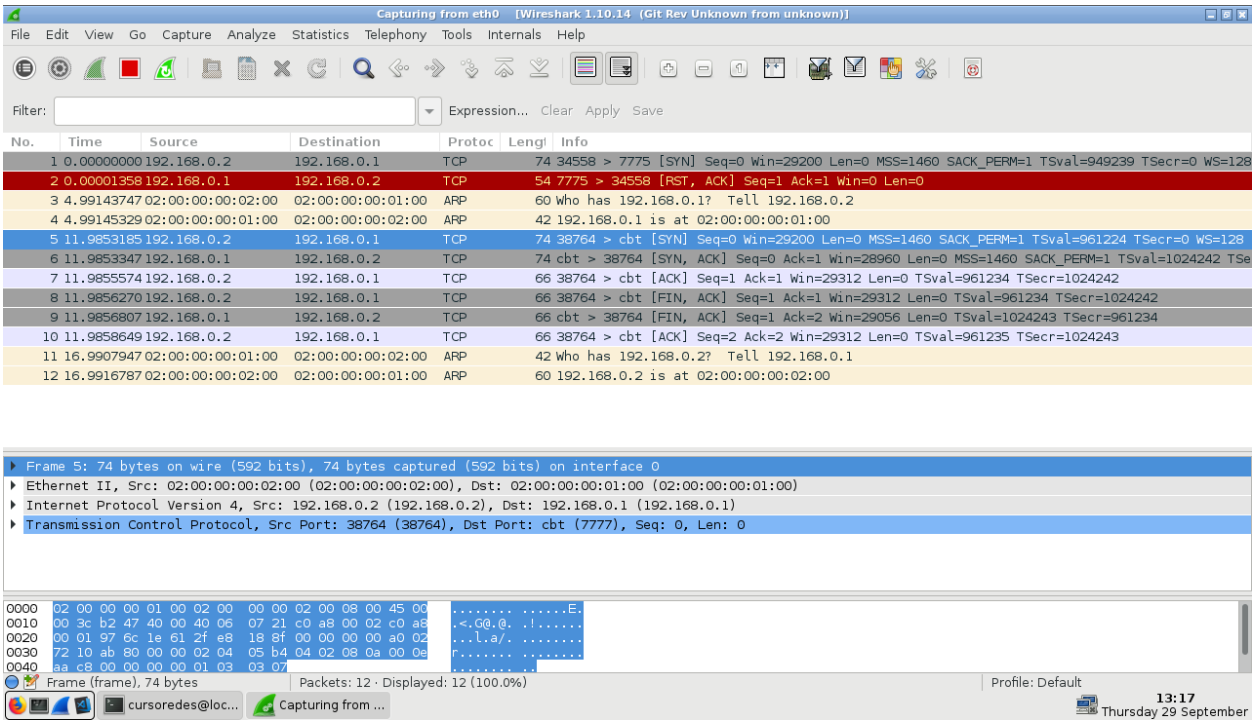
Ejercicio 8. (Técnica CONNECT) Netcat permite explorar puertos usando la técnica CONNECT que intenta establecer una conexión a un puerto determinado. En función de la respuesta (SYN+ACK o RST), es posible determinar si hay un proceso escuchando.

- (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.

[VM1]
nc -l 192.168.0.1 7777

[VM2]
nc -z -v 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.

En el resto de puertos (escribimos el ejemplo del puerto 7775) dan todos esta misma salida.
nc -z -v 192.168.0.1 7775
Ncat: Version 7.50 (https://nmap.org/ncat)
Ncat: Connection refused.



Captura de Wireshark, donde primero hemos explorado el puerto 7775 (mensaje 2 en rojo) y luego el puerto 7777, que sí está conectado (a partir del mensaje 5).

Opcional. La herramienta Nmap permite realizar diferentes tipos de exploración de puertos, que emplean estrategias más eficientes (SYN *stealth*, ACK *stealth*, FIN-ACK *stealth*…). Estas estrategias 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.

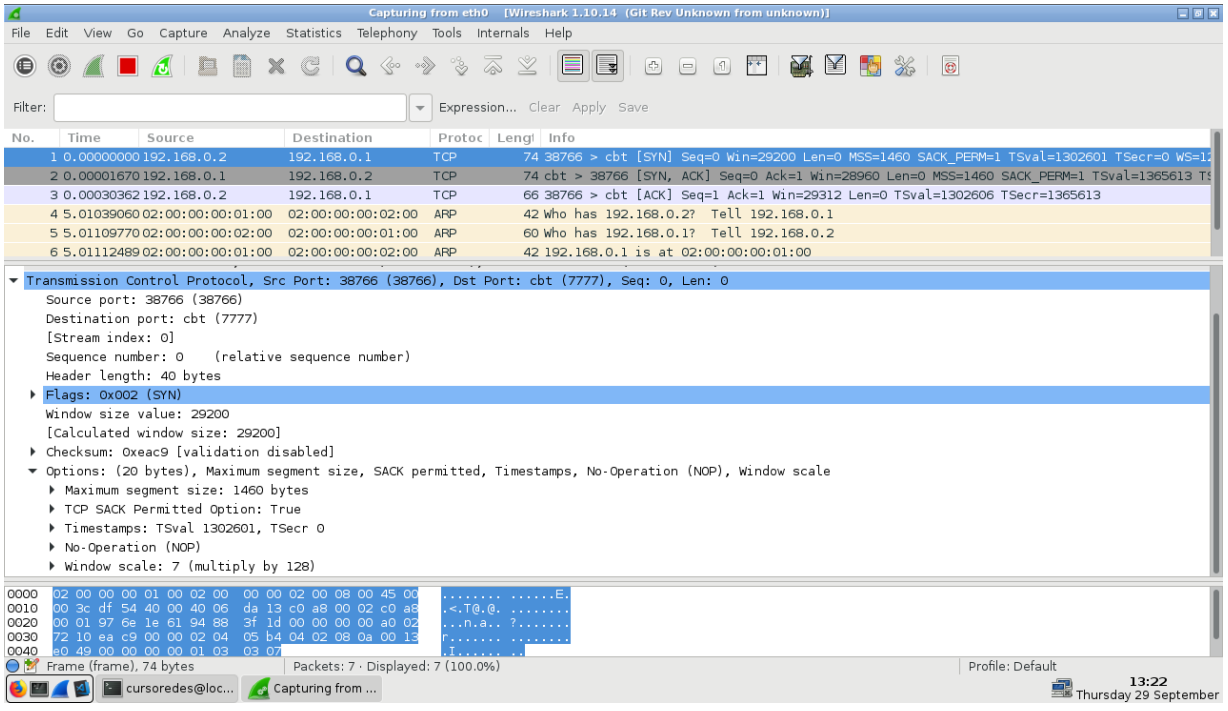
Ejercicio 9. Con ayuda del comando `sysctl` y la bibliografía recomendada, completar la siguiente tabla con parámetros que permiten modificar algunas opciones de TCP:

Parámetro del kernel	Propósito	Valor por defecto
`net.ipv4.tcp_window_scaling`	Habilitar la escala de ventana definida por RFC 1323, para admitir ventanas TCP que superen los 64 KB, este valor debe estar habilitado (1 significa habilitar), y la ventana TCP máxima es 1 GB, y entrará en vigencia cuando ambos lados de la conexión TCP estén habilitados.	1
`net.ipv4.tcp_timestamps`	Marca de tiempo TCP (se agregan 12	1

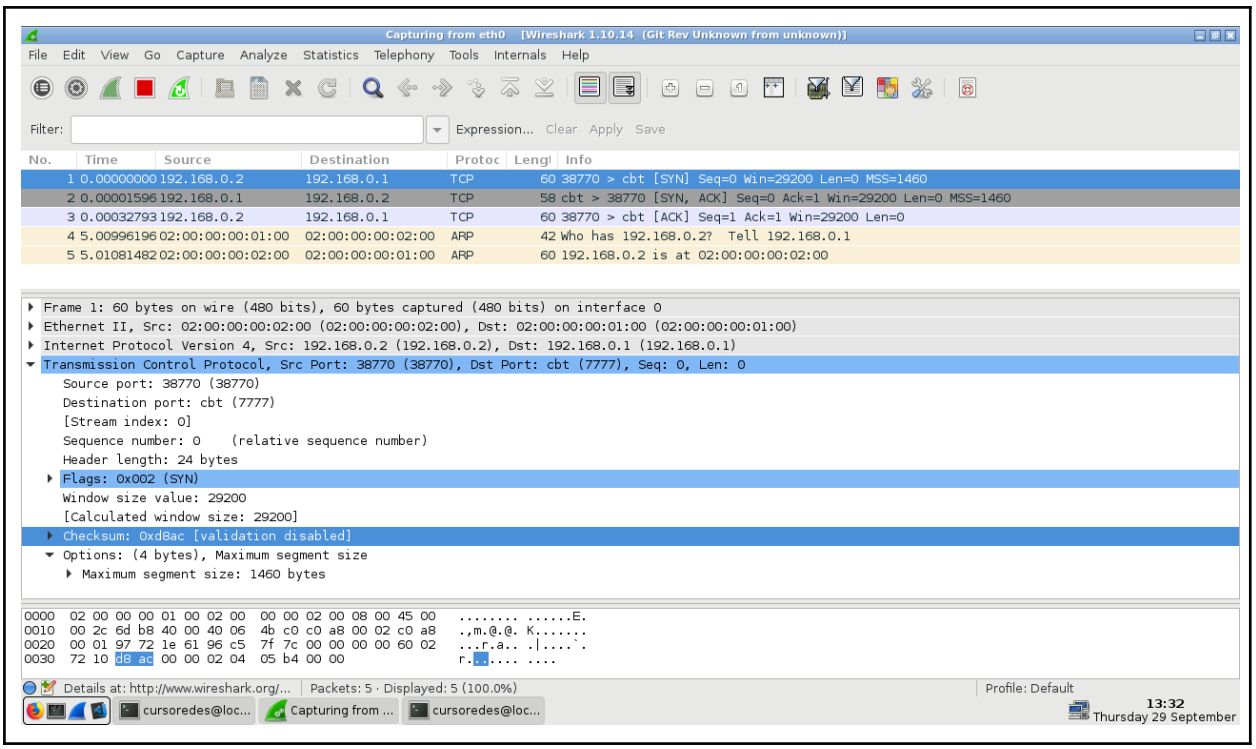
	bytes al encabezado TCP), un método más preciso que el tiempo de espera de retransmisión (consulte RFC 1323) se utiliza para habilitar el cálculo de RTT. Esta opción debe habilitarse para un mejor rendimiento.	
`net.ipv4.tcp_sack`	Habilita la respuesta selectiva (1 significa habilitar), mejora el rendimiento respondiendo selectivamente a los mensajes recibidos fuera de orden, de modo que el remitente sólo envía los segmentos de mensaje que faltan, (para la comunicación WAN) esta opción debe estar habilitada, pero aumentará el uso de la CPU	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.

Copiar una captura de pantalla de Wireshark donde se muestren las opciones TCP.
Primero, realizamos una conexión entre el servidor y el cliente con los parámetros anteriores en su valor por defecto. Vemos que sus valores aparecen en los mensajes enviados.



Ahora, al desactivar todas las opciones de timestamp, sack y window_scale, vemos que no aparecen como información opcional en los mensajes TCP:



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`	¿Cuántos mensajes de sondeo keepalive se pueden enviar antes de determinar que la conexión TCP no es válida?	9
`net.ipv4.tcp_keepalive_intvl`	Cuando el mensaje de la sonda no recibe una respuesta, el intervalo de tiempo (en segundos) para reenviar el mensaje.	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:

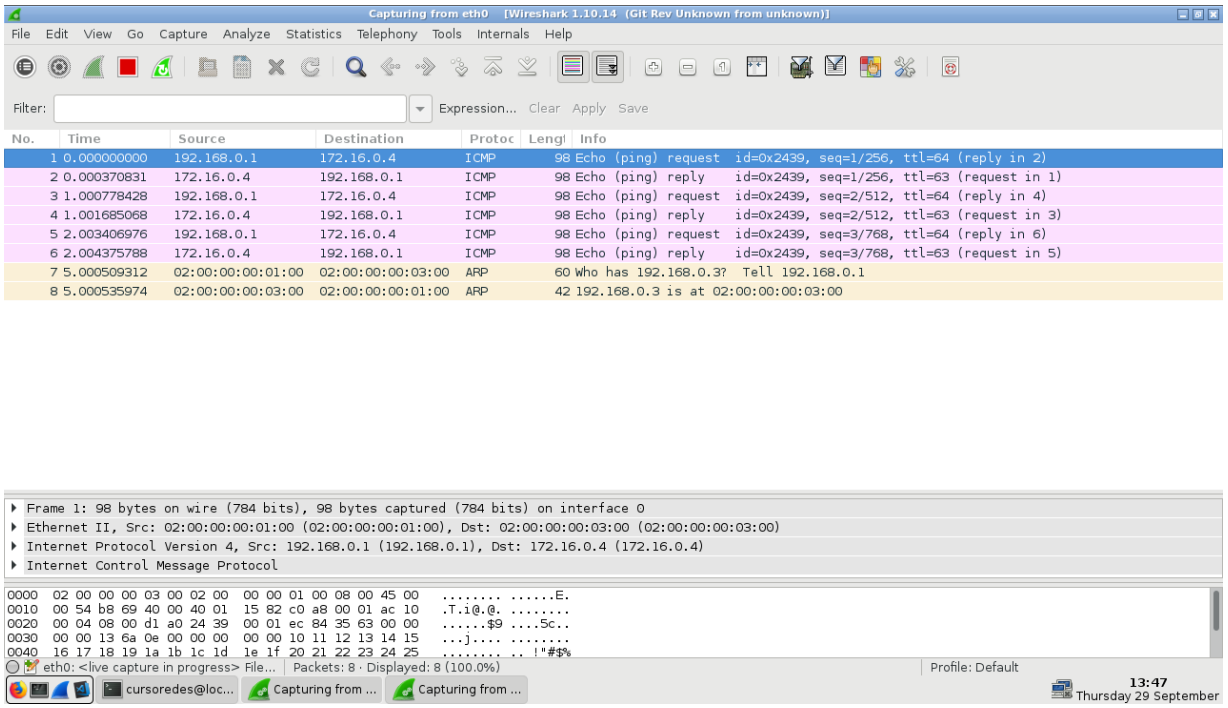
- (Router) Usando `iptables`, configurar Router para que haga SNAT (*masquerade*) sobre la interfaz `eth1`. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Comprobar la conexión con VM4 usando la orden `ping`.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente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

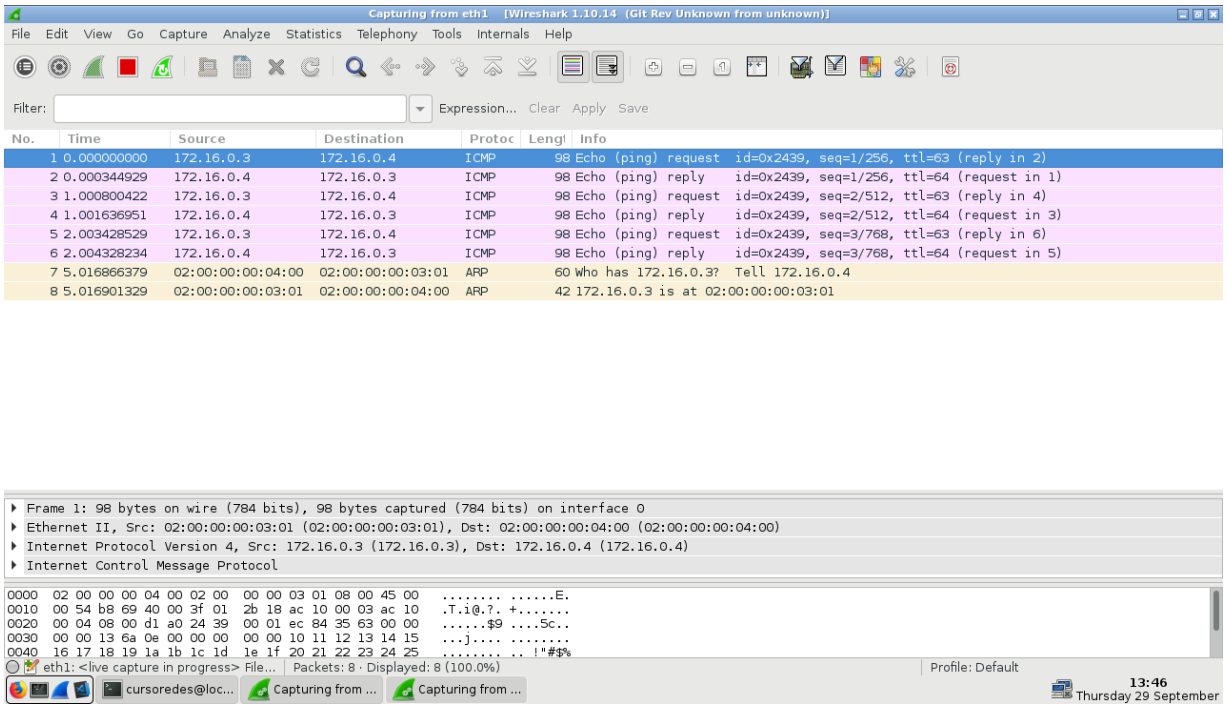
direcciones IP origen y destino en ambas redes

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

[VM3]
sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

[VM1]
Ping 172.16.0.4
PING 172.16.0.4 (172.16.0.4) 56(84) bytes of data.
64 bytes from 172.16.0.4: icmp_seq=1 ttl=63 time=1.68 ms





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?

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

[VM3]
sudo conntrack -L
icmp 1 29 src=192.168.0.1 dst=172.16.0.4 type=8 code=0 id=9330 src=172.16.0.4 dst=172.16.0.3 type=0 code=0 id=9330 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.

Se usa el parámetro ICMP 9330.

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.

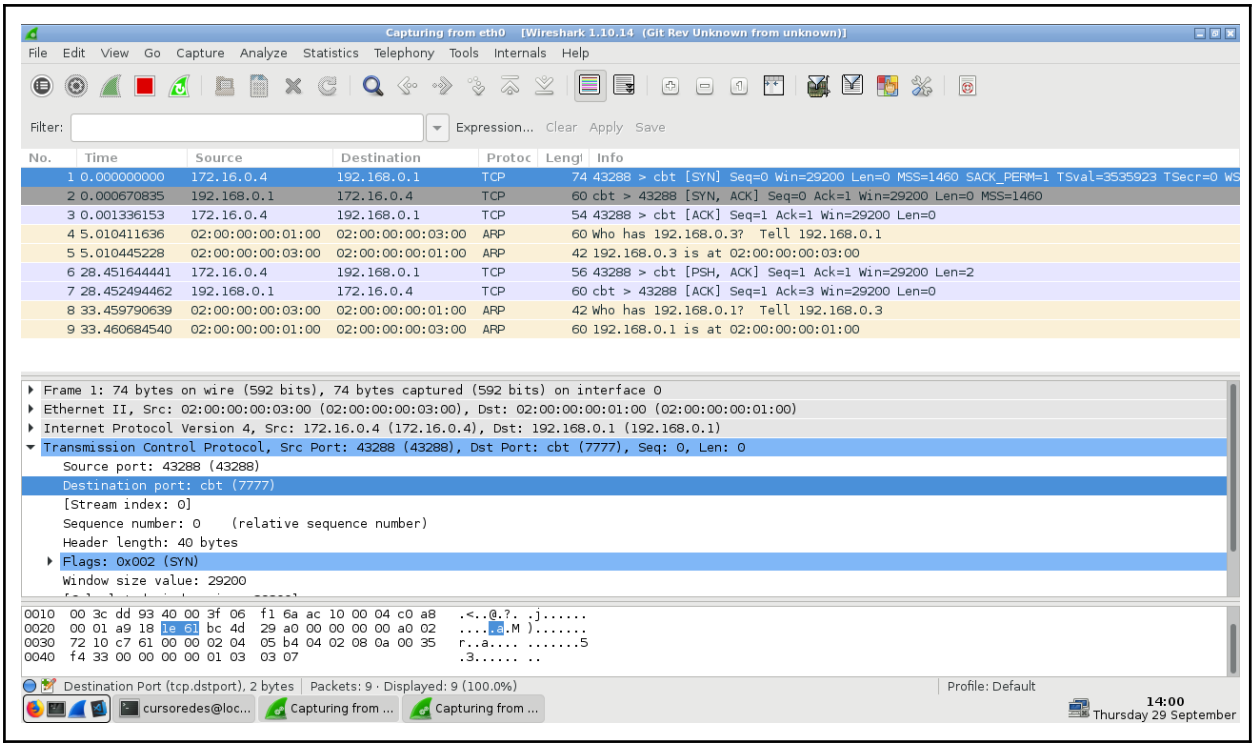
Copiar el comando iptables utilizado y capturas de pantalla de Wireshark.

[VM3]
iptables –t nat –A PREROUTING –d 192.168.0.3 –p tcp --dport 80 -j DNAT —to 192.168.0.1:7777

[VM1]
nc -l 192.168.0.1 7777

[VM4]
Nc 192.168.0.3 80

Vemos que al enviar un carácter desde VM4 llega hasta VM1, el servidor con puerto 7777.


Capturing from eth0 [Wireshark 1.10.14 (Git Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Expression... Clear Apply Save
No. Time Source Destination Protoc Lengt Info
1 0.000000000 172.16.0.4 192.168.0.1 TCP 74 43288 > cbt [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3535923 TSecr=0 WS
2 0.000670835 192.168.0.1 172.16.0.4 TCP 60 cbt > 43288 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460
3 0.001336153 172.16.0.4 192.168.0.1 TCP 54 43288 > cbt [ACK] Seq=1 Ack=1 Win=29200 Len=0
4 5.010411636 02:00:00:00:01:00 02:00:00:00:03:00 ARP 60 Who has 192.168.0.3? Tell 192.168.0.1
5 5.010445228 02:00:00:00:03:00 02:00:00:00:01:00 ARP 42 192.168.0.3 is at 02:00:00:00:03:00
6 28.451644441 172.16.0.4 192.168.0.1 TCP 56 43288 > cbt [PSH, ACK] Seq=1 Ack=1 Win=29200 Len=2
7 28.452494462 192.168.0.1 172.16.0.4 TCP 60 cbt > 43288 [ACK] Seq=1 Ack=3 Win=29200 Len=0
8 33.459790639 02:00:00:00:03:00 02:00:00:00:01:00 ARP 42 Who has 192.168.0.1? Tell 192.168.0.3
9 33.460684540 02:00:00:00:01:00 02:00:00:00:03:00 ARP 60 192.168.0.1 is at 02:00:00:00:01:00
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Ethernet II, Src: 02:00:00:00:03:00 (02:00:00:00:03:00), Dst: 02:00:00:00:01:00 (02:00:00:00:01:00)
Internet Protocol Version 4, Src: 172.16.0.4 (172.16.0.4), Dst: 192.168.0.1 (192.168.0.1)
Transmission Control Protocol, Src Port: 43288 (43288), Dst Port: cbt (7777), Seq: 0, Len: 0
Source port: 43288 (43288)
Destination port: cbt (7777)
[Stream index: 0]
Sequence number: 0 (relative sequence number)
Header length: 40 bytes
Flags: 0x002 (SYN)
Window size value: 29200
0010 00 3c dd 93 40 00 3f 06 f1 6a ac 10 00 04 c0 a8 .<..@.?. .j......
0020 00 01 a9 18 1e 61 bc 4d 29 a0 00 00 00 00 a0 02 .....a.M).......
0030 72 10 c7 61 00 00 02 04 05 b4 04 02 08 0a 00 35 r..a.... 5
0040 f4 33 00 00 00 00 01 03 03 07 .3...... ..
Destination Port (tcp.dstport), 2 bytes
Packets: 9 · Displayed: 9 (100.0%)
Profile: Default
cursoredes@loc...
Capturing from ...
Capturing from ...
14:00
Thursday 29 September