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**.

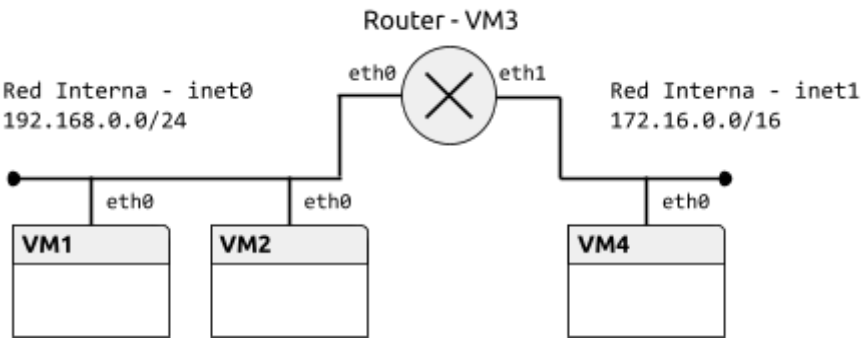
La contraseña del usuario `cursoredes` es `cursoredes`.

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`.

```
[cursoredes@localhost ~]$ ss -tln
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN   0      10          *:7777                    *:*
LISTEN   0      128         *:111                     *:*
LISTEN   0      128         *:22                      *:*
LISTEN   0      128   127.0.0.1:631                   *:*
LISTEN   0      10       :::7777                   :::*
LISTEN   0      128      :::111                    :::*
LISTEN   0      128      :::22                     :::*
LISTEN   0      128      ::1:631                   :::*

[cursoredes@localhost ~]$ ss -tln
State    Recv-Q Send-Q Local Address:Port          Peer Address:Port
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      10       :::7777                   :::*
LISTEN   0      128      :::111                    :::*
LISTEN   0      128      :::22                     :::*
LISTEN   0      128      ::1:631                   :::*
```

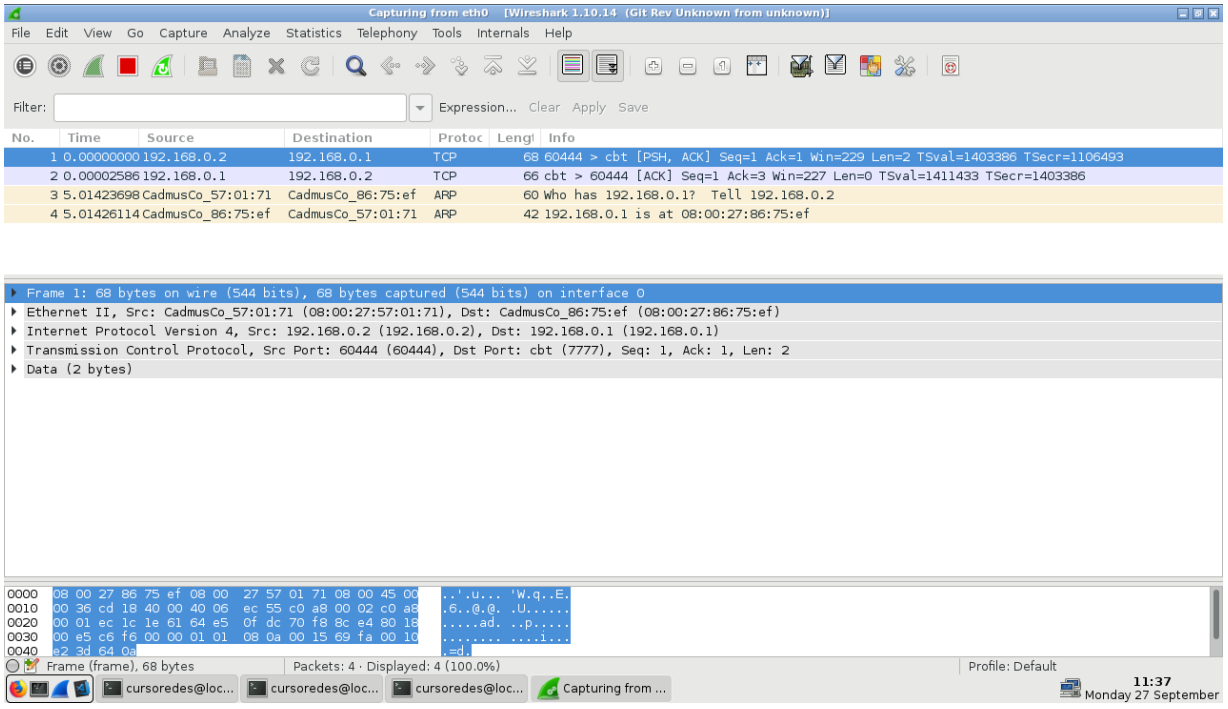
```
[cursoredes@localhost ~]$ nc localhost 7776
Ncat: Connection refused.
```

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      192.168.0.2:60444               192.168.0.1:7777
```

Cliente:
Dirección IP: 192.168.0.2
Puerto: 60444



Han sido necesarios dos mensajes (uno de envío de mensaje y otro de la confirmación del recibo)
Primer mensaje:
Se han enviado 68 bytes, pero solo 2 de datos (32 de cabera, 12 de options
Nº de secuencia del envío: 1
Nº de confirmación: 1
Flags: PSH y ACK

Segundo mensaje:
Se han enviado 66 bytes, pero 0 de datos (es un mensaje de confirmación del envío anterior)
Nº de secuencia del envío: 1
Nº de confirmación: 3 (confirma el envío de la secuencia 1 enviada anteriormente, de longitud 2)
Flags: ACK

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.

```
[cursoredes@localhost ~]$ ss -tan
State     Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN    0      128        *:111                    *:*
LISTEN    0      128        *:22                     *:*
LISTEN    0      128   127.0.0.1:631                 *:*
TIME-WAIT 0      0     192.168.0.2:60444            192.168.0.1:7777
LISTEN    0      128       :::111                   :::*
LISTEN    0      128       :::22                    :::*
LISTEN    0      128      ::1:631                   :::*

[cursoredes@localhost ~]$ ss -o -tan
State     Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN    0      128        *:111                    *:*
LISTEN    0      128        *:22                     *:*
LISTEN    0      128   127.0.0.1:631                 *:*
TIME-WAIT 0      0     192.168.0.2:60444            192.168.0.1:7777          timer:(timewait,1.105ms,0)
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.

Desde VM1:

```
[cursoredes@localhost ~]$ sudo iptables -A INPUT -s 192.168.0.2 -p tcp --tcp-flags ALL SYN -j DROP
```

*-A INPUT es para añadir regla en la tabla INPUT, es decir, para los mensajes recibidos
*-s DIR.IP es para que la regla sea solo para esa máquina, en este caso el cliente de la VM2
*-p tcp es para el protocolo tcp
*--tcp-flags es para marcar los flags. Se ponen primero en los que te quieres fijar (en este caso ponemos ALL) y luego pones los flags que quieres que estén activados (el resto desactivados) de ese mensaje
*-j DROP es como la regla, en este caso para decir que rechace lo que cumple anteriormente

Desde VM2 (salida):

```
[cursoredes@localhost ~]$ ss -tan
State     Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN    0      128        *:111                    *:*
LISTEN    0      128        *:22                     *:*
LISTEN    0      128   127.0.0.1:631                 *:*
SYN-SENT  0      1     192.168.0.2:60446            192.168.0.1:7777
LISTEN    0      128       :::111                   :::*
```

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

Desde VM1: Borramos la regla
[cursoredes@localhost ~]$ sudo iptables -D INPUT 1

Desde VM2: Añadimos la regla
[cursoredes@localhost ~]$ sudo iptables -A OUTPUT -s 192.168.0.2 -p tcp --tcp-flags ALL ACK -j DROP

Desde VM1:
[cursoredes@localhost ~]$ ss -tan

```
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN     0      10     192.168.0.1:7776               *:*
LISTEN     0      10           *:7777                *:*
SYN-RECV   0      0      192.168.0.1:7777          192.168.0.2:60468
LISTEN     0      128          *:111                 *:*
LISTEN     0      128          *:22                  *:*
LISTEN     0      128    127.0.0.1:631               *:*
LISTEN     0      10         :::7777               :::*
LISTEN     0      128        :::111                :::*
LISTEN     0      128        :::22                 :::*
LISTEN     0      128        ::1:631               :::*
```

[cursoredes@localhost ~]$ ss -o -tan

```
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN     0      10     192.168.0.1:7776               *:*
LISTEN     0      10           *:7777                *:*            timer:(keepalive,069ms,0)
SYN-RECV   0      0      192.168.0.1:7777          192.168.0.2:60472          timer:(on,7.469ms,3)
LISTEN     0      128          *:111                 *:*
LISTEN     0      128          *:22                  *:*
LISTEN     0      128    127.0.0.1:631               *:*
LISTEN     0      10         :::7777               :::*
LISTEN     0      128        :::111                :::*
LISTEN     0      128        :::22                 :::*
LISTEN     0      128        ::1:631               :::*
```

Se han realizado 3 retransmisiones, cada algo menos de 2.5 ms

Al cerrar la conexion desde VM2:

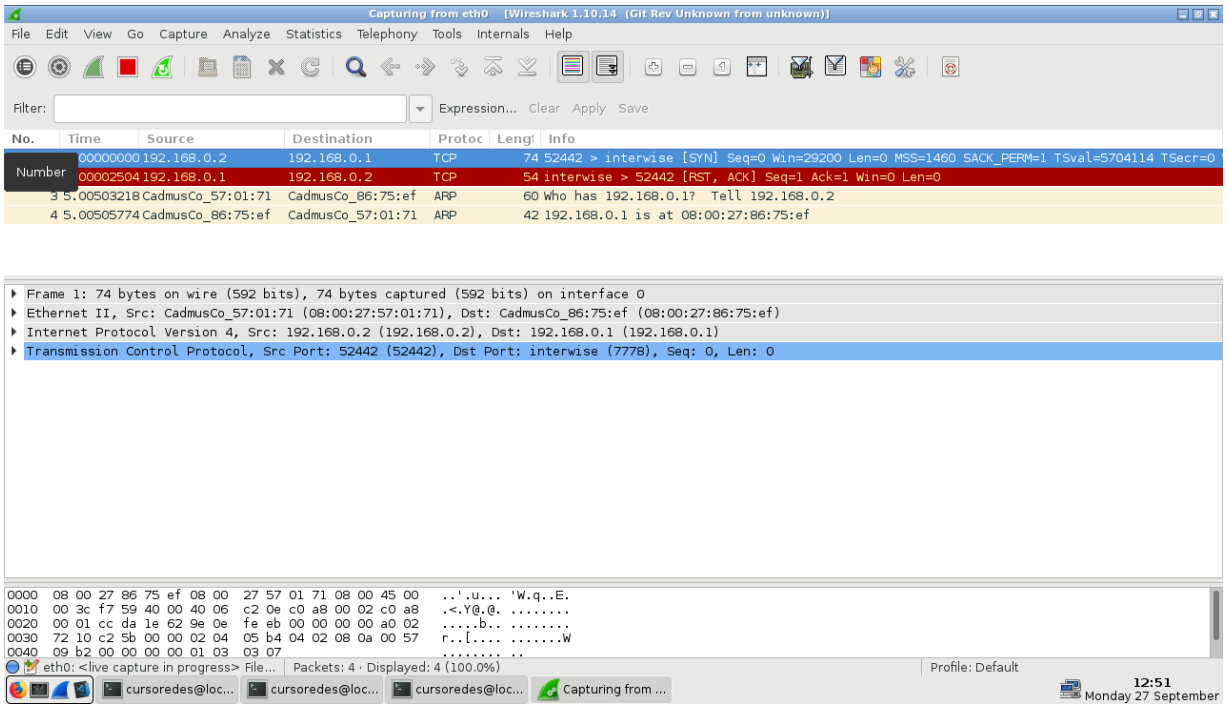
[cursoredes@localhost ~]$ ss -o -tan

```
State      Recv-Q Send-Q Local Address:Port          Peer Address:Port
LISTEN     0      10     192.168.0.1:7776               *:*
LISTEN     0      10           *:7777                *:*            timer:(keepalive,069ms,0)
SYN-RECV   0      0      192.168.0.1:7777          192.168.0.2:60472          timer:(on,7.469ms,3)
LISTEN     0      128          *:111                 *:*
LISTEN     0      128          *:22                  *:*
LISTEN     0      128    127.0.0.1:631               *:*
LISTEN     0      10         :::7777               :::*
LISTEN     0      128        :::111                :::*
LISTEN     0      128        :::22                 :::*
LISTEN     0      128        ::1:631               :::*
```

Desde VM2: Borramos la regla
[cursoredes@localhost ~]$ sudo iptables -D OUTPUT -s 192.168.0.2 -p tcp --tcp-flags ALL SYN,ACK -j DROP
(vale con sudo iptables -D OUTPUT 1)

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.

Desde VM2:
[cursoredes@localhost ~]$ nc 192.168.0.1 7778
Ncat: Connection refused.



Mensaje 1:
Flag activado - SYN (intento de establecer conexion)

Mensaje 2:
Flags activados - RST (aborto de conexion) y ACK (confirmacion de la recepcion del mensaje)

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`).

En VM2:
[cursoredes@localhost ~]$ sudo iptables -A INPUT -s 192.168.0.1 -p tcp --tcp-flag ALL SYN,ACK -j DROP

[cursoredes@localhost ~]$ 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

En VM3:
[cursoredes@localhost ~]$ nc 192.168.0.1 22
SSH-2.0-OpenSSH_7.4

*Deja conectar

En VM1:
[cursoredes@localhost ~]$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.0.1 netmask 255.255.255.0 broadcast 0.0.0.0
inet6 fe80::a00:27ff:fe86:75ef prefixlen 64 scopeid 0x20<link>
ether 08:00:27:86:75:ef txqueuelen 1000 (Ethernet)
RX packets 27888371 bytes 1673323130 (1.5 GiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 27086247 bytes 1625203920 (1.5 GiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Nº de paquetes recibidos(RX): 27.888.371

En VM2:
--- 192.168.0.1 hping statistic ---
6804541 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

En VM1:
[cursoredes@localhost ~]$sudo sysctl net.ipv4.tcp_syncookies=0

```
En VM2:
[cursoredes@localhost ~]$ sudo hping3 -p 22 -S --flood 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

En VM3:
[cursoredes@localhost ~]$ nc 192.168.0.1 22
Ncat: Connection timed out.

En VM1:
[cursoredes@localhost ~]$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.1  netmask 255.255.255.0  broadcast 0.0.0.0
        inet6 fe80::a00:27ff:fe86:75ef  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:86:75:ef  txqueuelen 1000  (Ethernet)
        RX packets 41727703  bytes 2503683118 (2.3 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 33676868  bytes 2020641186 (1.8 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Nº paquetes recibidos: 41.727.703

En VM2:
--- 192.168.0.1 hping statistic ---
7271585 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

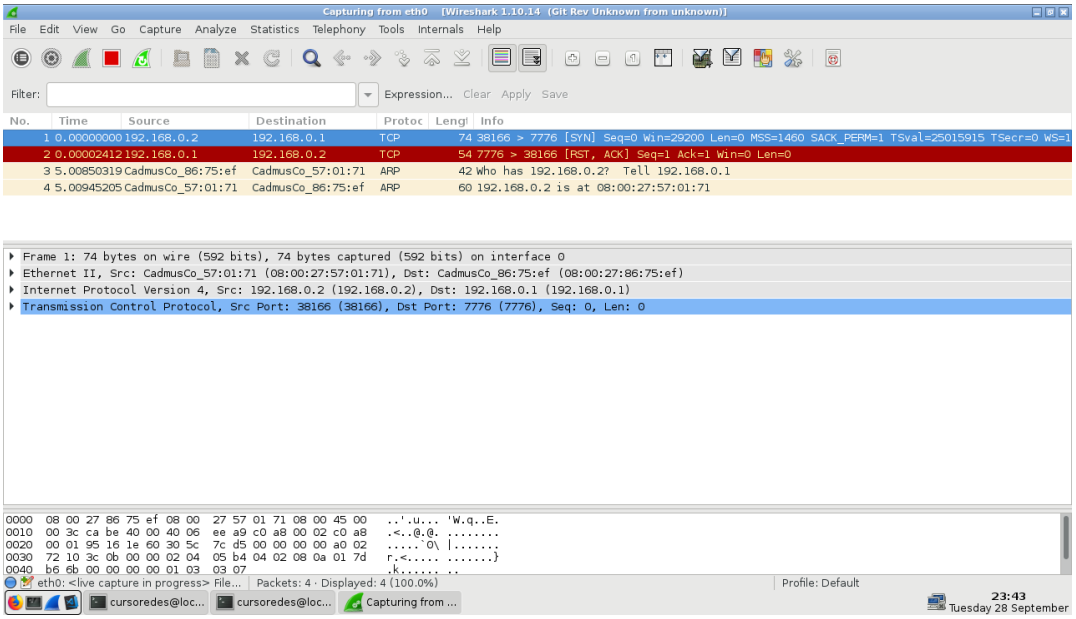
Nota: Wireshark no debe estar activo cuando se envían paquetes lo más rápido posible (*flooding*).

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.

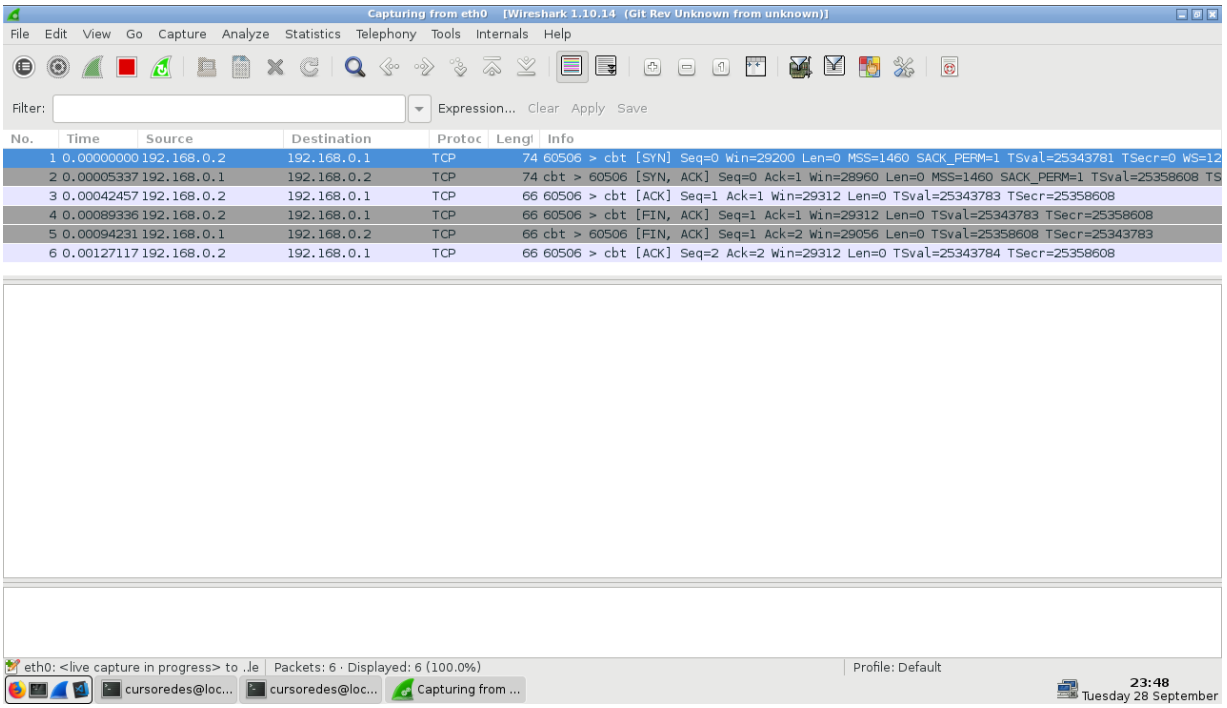
- (Servidor VM1) Abrir un servidor en el puerto 7777.
- (Cliente VM2) Explorar, de uno en uno, el rango de puertos 7775-7780 usando nc, en este caso usar las opciones de exploración (-z) y de salida detallada (-v).
- Con ayuda de Wireshark, observar los paquetes intercambiados.

```
En VM1:
[cursoredes@localhost ~]$ nc -l 7777

En VM2:
[cursoredes@localhost ~]$ nc -z -v 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 -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.
```




```
[cursoredes@localhost ~]$ nc -z -v 192.168.0.1 7778
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connection refused.
[cursoredes@localhost ~]$ nc -z -v 192.168.0.1 7779
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connection refused.
```

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

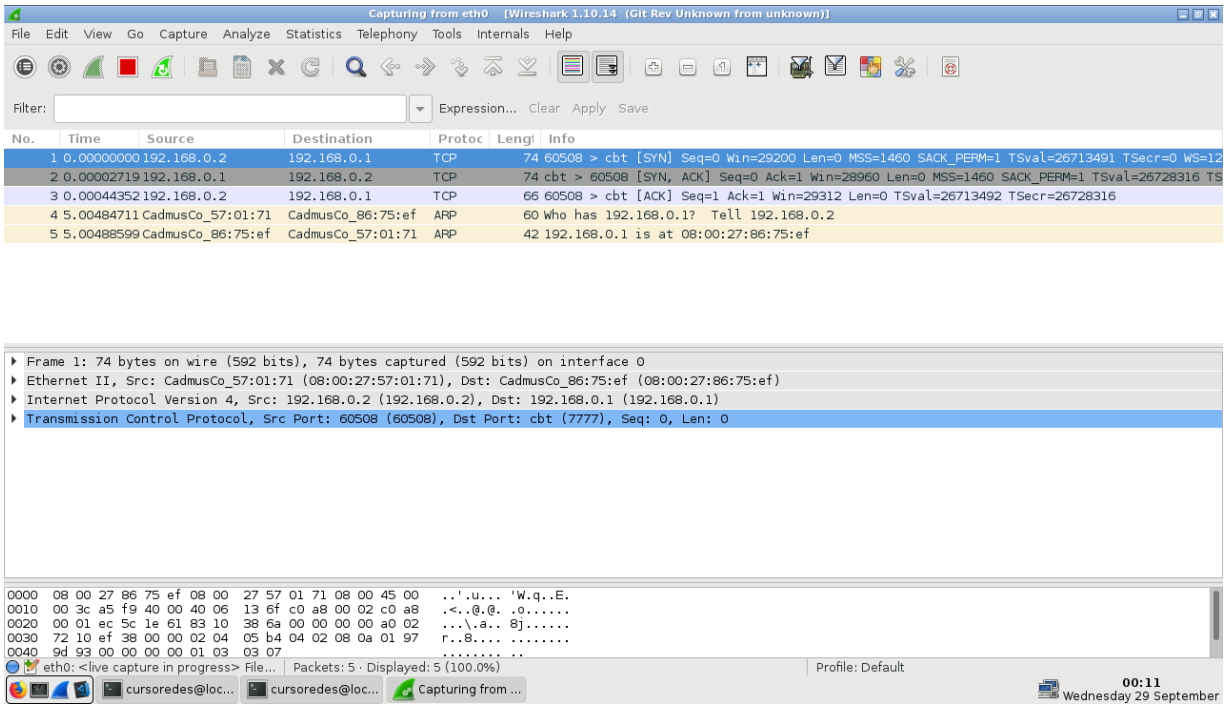
El comportamiento de la conexión TCP se puede controlar con varias opciones que se incluyen en la cabecera en los mensajes SYN y que son configurables en el sistema operativo por medio de parámetros del kernel.

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`	Activar la escala de ventana definida, para admitir ventanas de recepción TCP que superen los 64 KB.	1
`net.ipv4.tcp_timestamps`	Activa marca de tiempo TCP, un método más preciso que el tiempo de espera de retransmisión. Se utiliza para habilitar el cálculo de RTT	1
`net.ipv4.tcp_sack`	Activar la respuesta selectiva para mejorar el rendimiento respondiendo a los mensajes recibidos fuera de orden, tal que el emisor solo envíe los segmentos de mensaje que faltan.	1

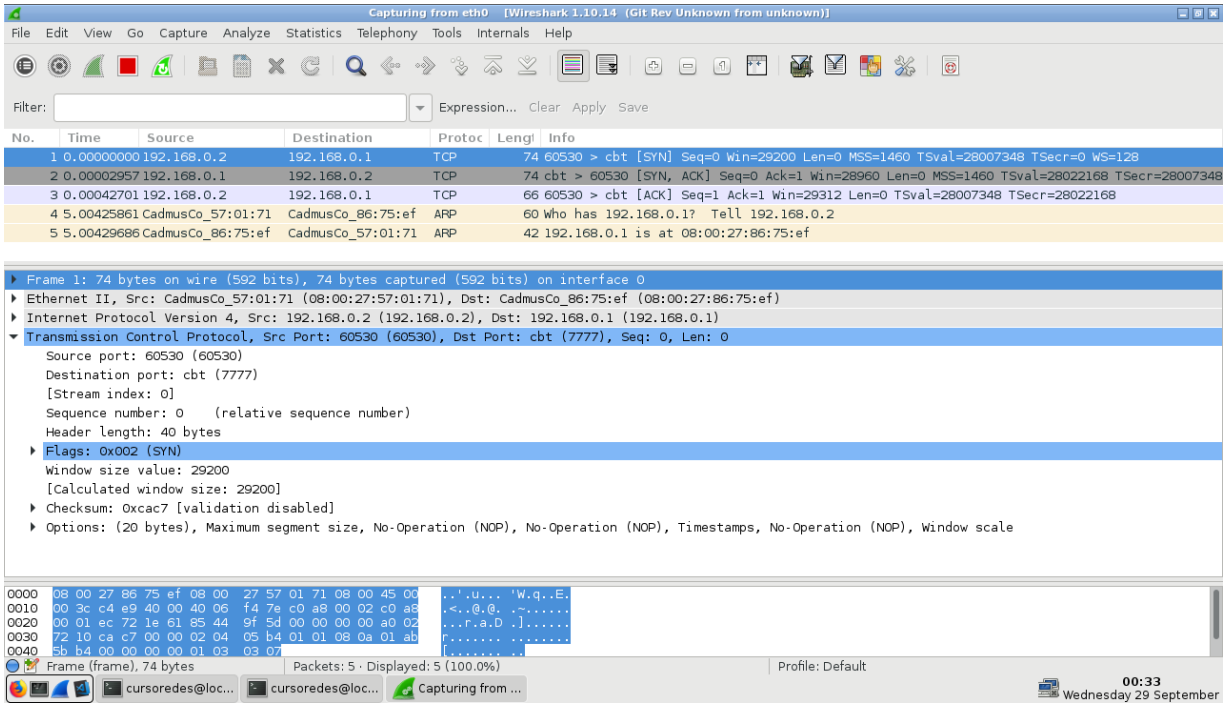
Ejercicio 10. Iniciar una captura de Wireshark. Abrir el servidor en el puerto 7777 y realizar una conexión desde la VM cliente. Estudiar el valor de las opciones que se intercambian durante la conexión. Variar algunos de los parámetros anteriores (ej. no usar ACKs selectivos) y observar el resultado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

Normal:

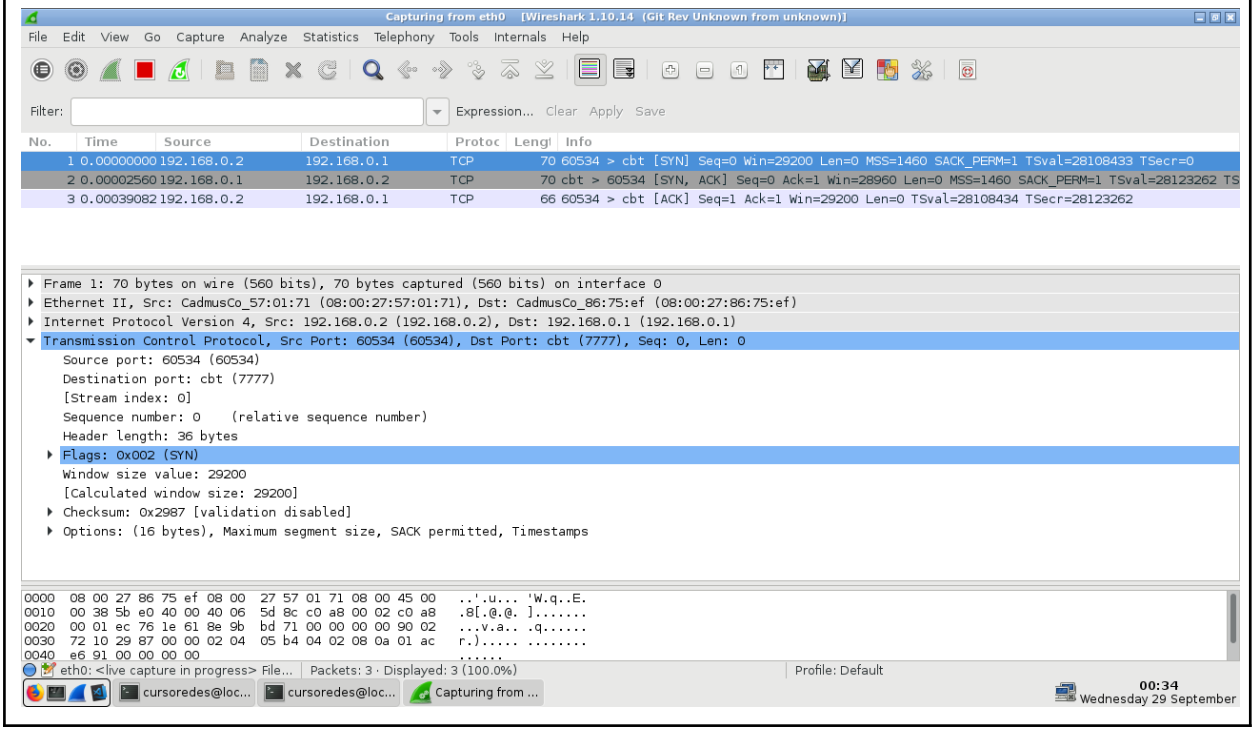


En VM1 y VM2:

[cursoredes@localhost ~]$ sudo sysctl net.ipv4.tcp_sack=0
net.ipv4.tcp_sack = 0



En VM1 y VM2:
[cursoredes@localhost ~]$ sudo sysctl net.ipv4.tcp_window_scaling=0
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 tiempo (en segundos) que pasa desde el último paquete enviado 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 keepalive se pueden enviar antes de determinar que la conexión TCP no es válida	9
`net.ipv4.tcp_keepalive_intvl`	El tiempo (en segundos) para volver a enviar un mensaje de sondeo keepalive cuando no recibo respuesta	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:

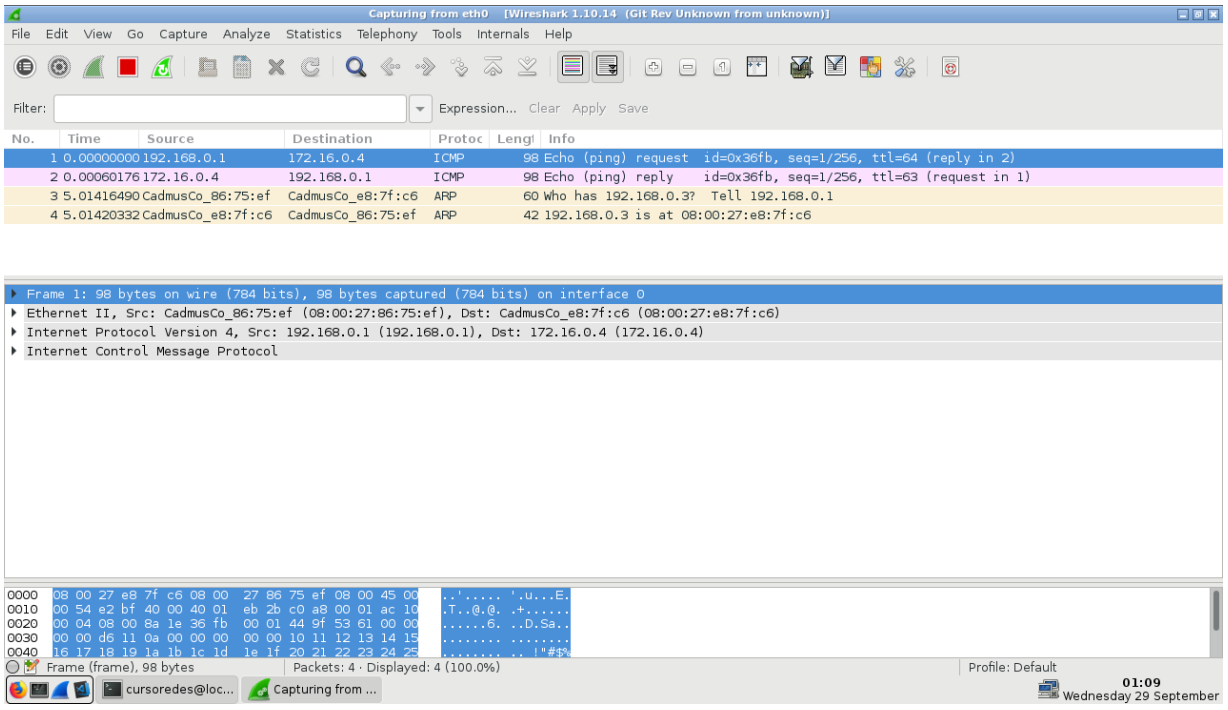
- (Router) Usando `iptables`, configurar Router para que haga SNAT (*masquerade*) sobre la interfaz `eth1`. Iniciar una captura de Wireshark en cada interfaz de red.
- (VM1) Comprobar la conexión con VM4 usando la orden `ping`.
- (Router) Analizar con Wireshark el tráfico intercambiado, especialmente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

En VM3:
[cursoredes@localhost ~]$ sudo iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE

En VM1:
[cursoredes@localhost ~]$ ping -c 1 172.16.0.4
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=1.15 ms

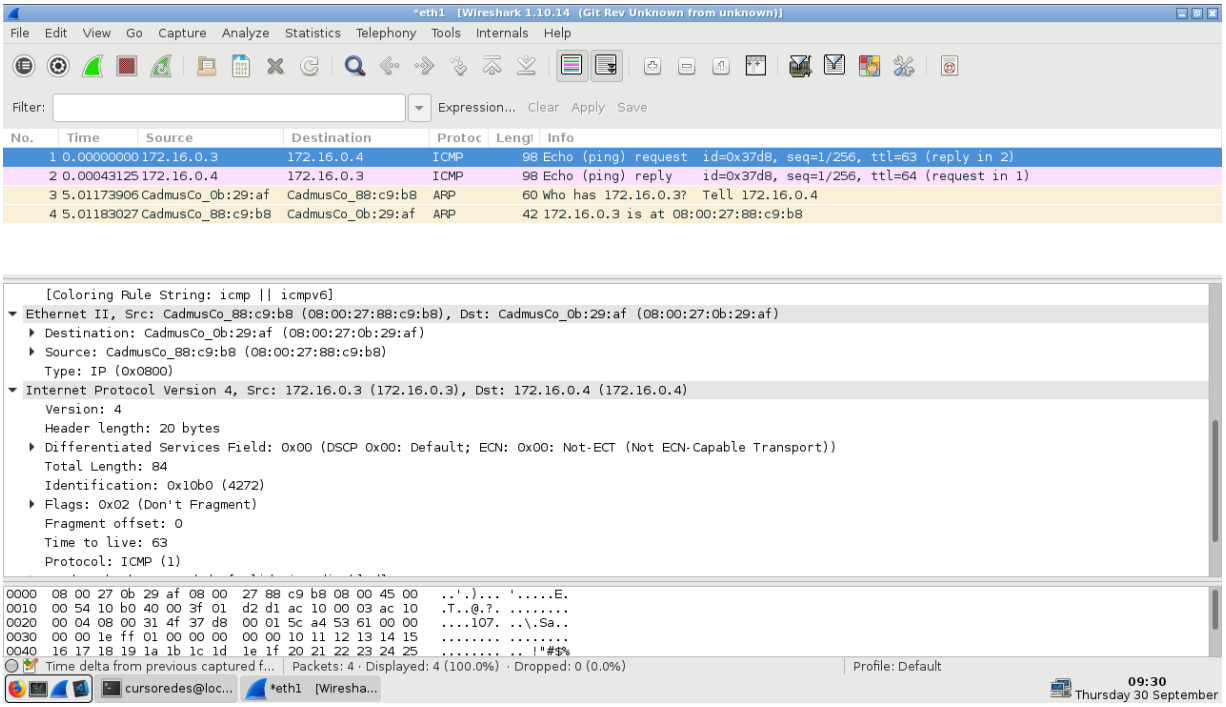
--- 172.16.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.158/1.158/1.158/0.000 ms

En eth0:



Observamos que en eth0 todo va de forma normal, se ven las direcciones privadas de ambas máquinas (tanto origen como destino), pero en eth1 observamos que la dirección IP de origen que aparece es la del router cuando debería aparecer la de VM1. Eso se debe a que hemos enmascarado la dir. IP privada de VM1 y mostramos la del router que es la pública

En eth1:



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?

```
[cursoredes@localhost ~]$ sudo conntrack -L
icmp     1 29 src=192.168.0.1 dst=172.16.0.4 type=8 code=0 id=15316
src=172.16.0.4 dst=192.168.0.1 type=0 code=0 id=15316 mark=0 use=1
conntrack v1.4.4 (conntrack-tools): 1 flow entries have been shown.
```

Se utiliza el parámetro **id** para relacionar solicitudes con respuestas. En este caso id=15316

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.

En VM3:

```
[cursoredes@localhost ~]$ iptables –t nat –A PREROUTING –p tcp --dport 80 -j DNAT –-to
192.168.0.1:7777
```

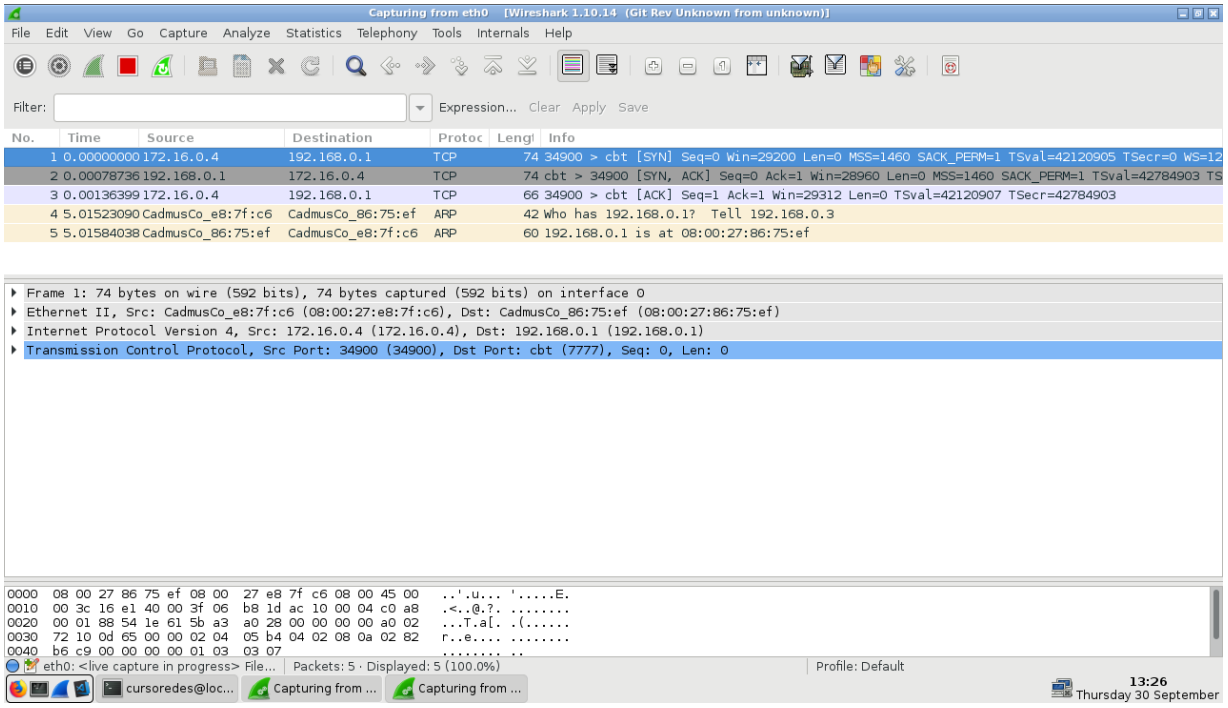
En VM1:

```
[cursoredes@localhost ~]$ nc -l 7777
```

En VM4:

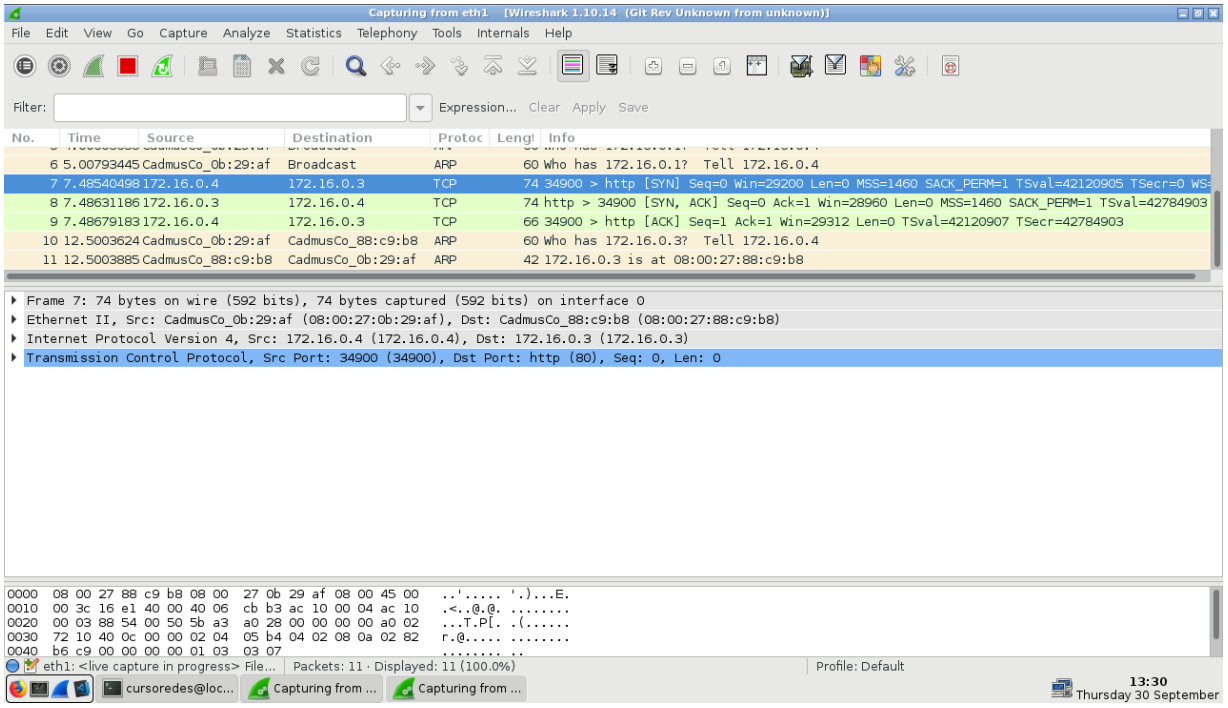
```
[cursoredes@localhost ~]$ nc 172.16.0.3 80
```

En eth0:



Observamos que es una conexión normal en eth con los mensajes propios entre la VM1 y la VM4. Vemos que se conecta VM4 con VM1 al puerto 7777 y las direcciones IP de estas máquinas. Todo es como si la conexión de VM4 hubiese sido solicitada a VM1 en ese puerto directamente.

En eth1:



Sin embargo, aquí observamos (en eth1) que la solicitud de conexión que ha realizado VM4 es al router (vemos claramente esto en la dir. IP) y al puerto 80. Por tanto, vemos claramente como se ha redireccionado la conexión.