Diseño y Administración de Redes

Práctica 1.2

Diseño y gestión de escenarios IPv4. NAT

Dpto. Ingeniería Electrónica y Comunicaciones
Área de Ingeniería Telemática

Departamento de
Ingeniería Electrónica
y Comunicaciones
Universidad Zaragoza

Autores:
Profesores del área de Ingeniería Telemática

1. Introducción

Esta práctica es continuación de la Práctica 1.1.

1.1. Objetivos

Tras la realización de esta práctica, el alumno deberá ser capaz de:

Configurar el acceso a Internet desde una subred privada (IPv4).

Identificar las limitaciones del NAT básico y adoptar las soluciones adecuadas.

1.2. Contenidos

Los objetivos propuestos en esta práctica pretenden afianzar y complementar los **contenidos teóricos vistos en clase**. Por lo tanto, será necesario un estudio previo de los mismos, así como la utilización de los apuntes de clase (y cualquier material adicional que el alumno considere oportuno) como apoyo a la realización práctica.

A modo de orientación, se enumeran aquellos aspectos de IPv4 que se consideran más relevantes:

Direccionamiento privado y direccionamiento público. Necesidad de NAT.
Tipos de NAT: funcionalidad.
Problemática de NAT.

Pueden resultar de utilidad los documentos RFC relativos a los distintos aspectos a analizar: http://tools.ietf.org/rfc/index.

Del mismo modo se recomienda, en caso necesario, consultar cualquier ayuda online, así como `man` de Linux, o `help` del interfaz de comandos de Windows.

Además, en los anexos se encuentra disponible información auxiliar que será necesaria durante la realización práctica:

Anexo I: Manuales de configuración y programas de análisis.

1.3. Equipos, tecnologías y herramientas

La práctica propuesta consiste en la configuración, verificación y análisis de un escenario de interconexión de redes IP. Para ello, se trabajará con máquinas Lisa_Centos y TinyCoreLinux para los diferentes equipos (vistas en la práctica de introducción de GNS3).

En cuanto a las **herramientas** necesarias para la verificación y el análisis de los escenarios, utilizaremos el software que proporciona la distribución Linux Centos para la generación de tráfico, así como las aplicaciones genéricas de comunicaciones disponibles. Como herramienta de captura usaremos ***tcpdump*** y el analizador de protocolos ***Wireshark***.

1.4. Escenarios

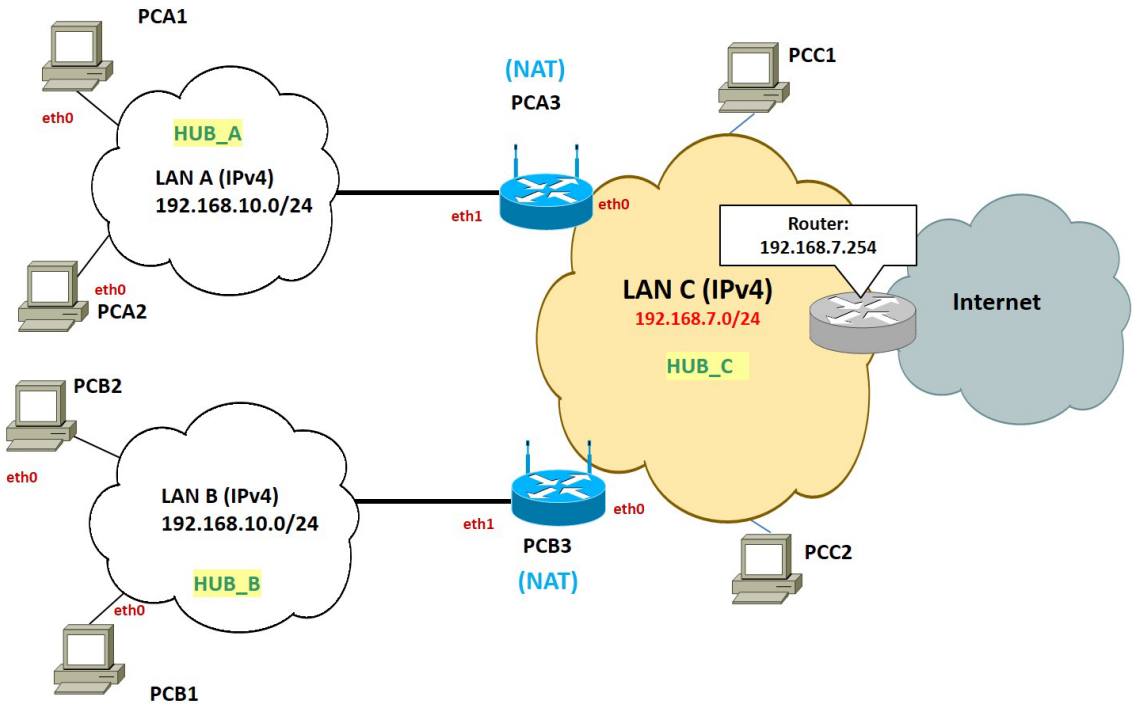


Figura 1: Escenario de interconexión de redes. Configuración IPv4 NAT.

La figura 1 muestra el escenario con el que se trabajará en la práctica. En él configuraremos tres redes: LAN_A, LAN_B y LAN_C. Los PC integrantes de las LAN A y B serán PCA1, PCA2, PCB1 y PCB2, que actúan como *host*, y PCA3 y PCB3 que tendrán funciones de *router* y estarán conectados también a LAN_C. LAN_A y LAN_B utilizarán el mismo direccionamiento IPv4 privado perteneciente a la red 192.168.10.0/24, asegurando que se repita, como puede ocurrir en la realidad. LAN_C utilizará el direccionamiento IPv4 privado perteneciente a la red 192.168.7.0/24, pero haciendo las veces de red pública. Se usará PCC1 y PCC2 para hacer las pruebas y comprobaciones de conexión que aparecen en el enunciado. El *router* lo usaremos para conectar nuestro escenario (con direccionamiento privado) al exterior.

2. Realización práctica en el laboratorio

**** La evaluación de la práctica requiere la entrega del escenario GNS3 con la configuración adecuada, las capturas correspondientes que avalen su correcto funcionamiento y un documento con la explicación oportuna. ****

**** Para facilitar la elaboración del documento explicativo aparecen una serie de cuestiones a lo largo del enunciado de la práctica. ****

**** Habrá cuestiones que hay que resolverlas antes de la práctica y también comprobar el grado de acierto en el transcurso de la misma. La previa se entregará individualmente y el resultado final por grupos ****

**** Las cuestiones marcadas como tipo T se resolverán después de la realización de la práctica y se entregarán individualmente ****

2.1. Conexión hacia el exterior. Utilización de NAT y TC

2.1.1. Configuración básica

El direccionamiento a asignar es el indicado en la figura 1.

PCA1 y PCB1 continúan con la configuración de la práctica anterior mientras que PCA2 y PCB2 vamos a sustituirlas para lisacentos pero manteniendo la configuración IP. Para PCA3, PCB3, PCC1 y PCC2 es necesario poner el MTU a 1500, que es su valor original y dejar las configuraciones de direccionamiento manual y encaminamiento estático (sólo vamos a configurar el *router* por defecto) que configuramos en el apartado correspondiente de la práctica anterior. Para esto, ejecutamos en los siguientes comandos:

```
Service zebra stop
Service ripd stop
ifdown eth0
ifup eth0
```

Y a continuación el script de Shell que hayamos programado para reiniciar la configuración.

Para configurar adecuadamente la traducción de direcciones (NAT) en los *router* de salida (PCA3 y PCB3), se debe utilizar el comando **`iptables`**. Para ello, consultar el Anexo I, apartado específico “Utilización de *iptables* como NAT”. Además de lo expuesto en la práctica anterior (P1.1), la configuración completa deberá tener en cuenta lo siguiente:

- En primera aproximación, no se necesita acceder desde el exterior a servidores internos de la red. Pueden realizarse conexiones desde varios equipos a la vez. El router sólo dispone de una dirección IP de la red C.
- No se activa ninguna capacidad de filtrado (*firewall*).

- Para configurar adecuadamente el control de tráfico (Traffic Control, TC) en los *router* de salida, se debe utilizar el comando **tc**.

2.1.2. Análisis NAT

Cuestión 1 (habrá que resolverla también previamente). Indica cómo debe configurarse el NAT en los *router* PCA3 y PCB3. El NAT que acabas de configurar, ¿es estático o dinámico?

Cuestión 2. Una vez configurado el escenario, verifica su funcionamiento en los casos a) y b) descritos a continuación. Captura en las interfaces oportunas e identifica el uso de direccionamiento a la entrada y salida del *router* NAT y los datos de las cabeceras ICMP y TCP implicados en el NAT.

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

a) Prueba una conexión desde los dos equipos de la LAN (PC1 y PC2) hacia el exterior **mediante el protocolo** ICMP (**ping** a PCC1 o 2). Captura mediante *Wireshark* en las interfaces oportunas y verifica el funcionamiento del NAT de acuerdo a lo indicado.

b) Prueba una conexión desde los dos equipos de la LAN (PC1 y PC2) hacia el exterior **mediante el protocolo TCP** (se puede acceder al *ssh* de PCC1 o PCC2, con el comando **$ ssh alumno@192.168.7.??**, password *alumno)*. Captura mediante *Wireshark* en las interfaces oportunas y verifica el funcionamiento del NAT. Vamos a forzar que en los dos equipos PC1 y PC2 se abra el mismo puerto origen (para que coincidan en el NAT del *router*) mediante el siguiente comando en PCA1 y en PCA2:

```
$ iptables -t nat -A POSTROUTING -o eth0 -p tcp --dport 22 –j
SNAT --to 192.168.N0.[1ó2]:2016
```

Ahora vamos a activar el servicio ftp en los *host* LisaCentos (**$ service vsftpd start**) y ejecutar el comando: **$ setsebool -P ftp_home_dir on** (este comando tarda unos 2 minutos en ejecutarse, paciencia). Como comprobación podemos ejecutar **$ getsebool –a | grep ftp** [1].

Cuestión 3. Conéctate con el servidor ftp de PCC1:

```
$ ftp 192.168.7.??
```

name = *anonymous*

[1] El comando **setsebool** sirve para establecer políticas de seguridad en Security-Enhanced Linux (https://github.com/SELinuxProject).

password = <una dirección de e-mail>

(la dirección de correo puede ser inexistente)

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

Realiza las siguientes pruebas:

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.

   Nota: El cliente ftp por defecto se inicia en modo pasivo, para cambiar al modo activo, es necesario ejecutar el comando ***passive***. Dicho comando también sirve para volver al modo pasivo:

   `ftp>passive` → passive mode off

   `ftp>passive` →passive mode on

b) Incorpora el módulo **`ip_nat_ftp`**.

   `$ modprobe 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.

Se desea ahora establecer una conexión ssh desde PCA1 (cliente) a PCB1 (servidor) de otra bancada (B). Tener en cuenta que lo que se pretende ahora es permitir que una petición de conexión que llegue a la salida del NAT (pública), pueda asociarse al servidor ssh interno (privada). Consultando de nuevo el Anexo I, apartado específico "Utilización de *iptables* como NAT", configura apropiadamente el NAT en el *router* correspondiente (PC3).

Cuestión 4 (habrá que resolverla también previamente). Indica cómo has configurado el NAT y qué comandos has utilizado. Tal y como has configurado el NAT, ¿es estático o dinámico?

Cuestión 5. Tomando como base las capturas, verifica y demuestra el funcionamiento de la conexión al servicio ssh.

2.1.3. Ancho de banda útil

Mantenemos los valores de MTU del apartado anterior.

Se propone realizar una estimación del ancho de banda disponible entre dos equipos. Para ello se va a utilizar la herramienta iperf (en centos 8 es iperf3 en lugar iperf, consultar el Anexo I.2.C para ver los detalles).

El objetivo es estimar el máximo ancho de banda que podemos conseguir para una transferencia fiable entre PCA1 y PCC1.

En primer lugar, lo haremos mediante un servicio fiable que utiliza TCP como nivel de transporte. El mecanismo de control de congestión de TCP garantiza que la aplicación utilice únicamente el ancho de banda que la red tiene disponible (el máximo posible, si no existe otro tráfico en la red, o el que, si hay tráfico, queda disponible sin saturar la red).

Establecer en PCC1 el servidor TCP de iperf: iperf –s (para servidor tcp)

Realizar la conexión con iperf como cliente desde PCA1. En iperf, el tráfico de datos se genera desde el cliente hacia el servidor. Vamos a comenzar generando tráfico TCP y en paralelo debemos capturar el tráfico mediante tcpdump o wireshark para comparar los resultados.

iperf –c @IP_servidor –t 30

A continuación vamos a generar tráfico UDP desde PCA1 con diferentes tamaños. Pero antes establecemos en PCC1 el servidor UDP de iperf: iperf -s -u (para servidor udp). En primer lugar generamos a 20 Mbps (y deberemos comprobar que estamos saturando el enlace obteniendo un resultado de transferencia inferior a este valor) con diferentes tamaños de trama: desde un valor pequeño, como 100, hasta 1472 (asegurando que no se fragmente).

iperf –c @IP_servidor –t 10 –u –b 20000000 –l 100

iperf –c @IP_servidor –t 10 –u –b 20000000 –l 200

iperf –c @IP_servidor –t 10 –u –b 20000000 –l 300

iperf –c @IP_servidor –t 10 –u –b 20000000 –l 600

iperf –c @IP_servidor –t 10 –u –b 20000000 –l 1200

iperf –c @IP_servidor –t 10 –u –b 20000000 –l 1472

Para cada tamaño obtenemos una tasa de transmisión que deberemos anotar. Además queremos medir la eficiencia a la vez que comprobamos si existen otras limitaciones (como por ejemplo el *througput* de pps enviados). Para los cálculos podemos utilizar la herramienta de análisis proporcionada con wireshark.

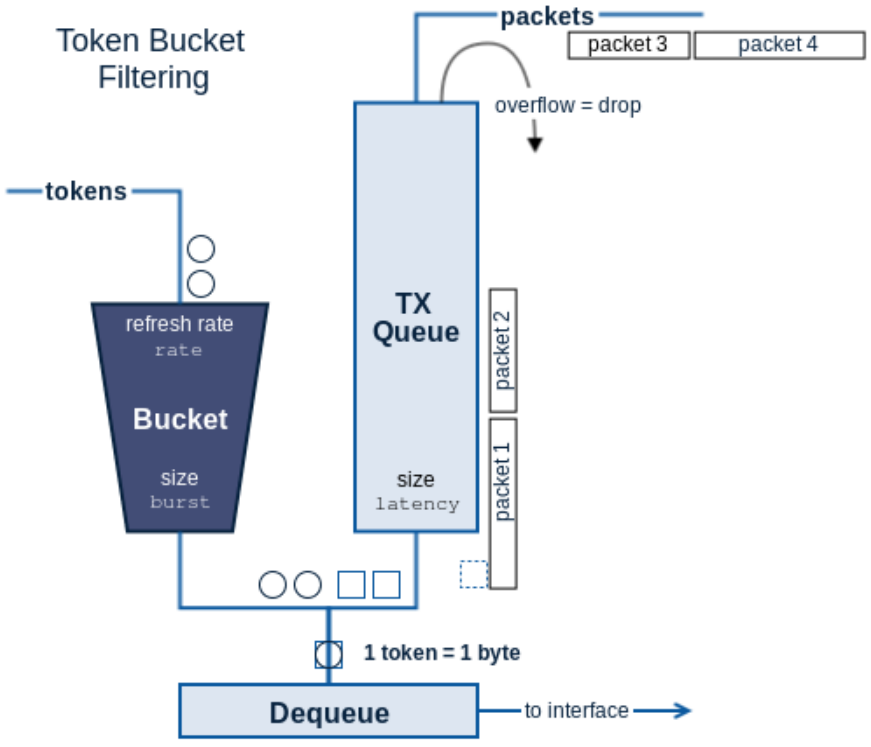
Cuestión 6. En base a las capturas del tráfico, calcula el trhougput (en pps) y el ancho de banda (en bps) total, útil a nivel IP y útil a nivel de aplicación, que se consigue en el escenario propuesto para cada uno los casos de generación de tráfico y compáralos con el dato de ancho de banda dado por la herramienta iperf. Analiza si la limitación aparece en el valor del trhougput o el de ancho de banda y si es a causa del generador de tráfico o del router.

Cuestión 7 tipo T. Calcula teóricamente el valor máximo que se obtendría para la cuestión 6. Teniendo en cuenta que la medida la realizamos sobre tecnología Ethernet 10Mbps y en modo half-dúplex (conexión a través de HUB donde datos y ACKs en TCP comparten el mismo medio.

Nota: Para los cálculos teóricos hay que tener en cuenta que Ethernet incluye un tiempo de espera vacío entre tramas transmitidas por el mismo equipo denominado Interframe Gap – IFG (equivalente al tiempo que cuesta transmitir 96 bits (12 bytes) en el medio, dependiente de la velocidad de la interfaz - 9.6 µs en 10 Mbit/s Ethernet, 960 ns en 100 Mbit/s (fast) Ethernet, etc.)

2.1.4. Análisis TC

La herramienta TC opera con una estructura de buffer para control de tráfico.



Visualization of the Token Bucket Filter (TBF) queuing discipline.
https://www.excentis.com/blog/use-linux-traffic-control-impairment-node-test-environment-part-2

Vamos a ejecutar el siguiente comando en las máquinas PC3.

```
$ tc qdisc add dev eth0 root tbf rate 1.0mbit limit 50kb burst 1.6kb
```

Este comando limita el ancho de banda a 1 Mbps y define un *burst* (ráfaga) en el que cabe una trama de 1500 bytes (establecemos `1.6kb` para asegurar que cabe una trama, pero sólo una). Si queremos comprobar si se ha actualizado la configuración, basta con ejecutar: `$ tc qdisc show`.

Podemos comprobar que el valor que aplica para la latencia es:

$$\frac{(limit-burst)*8*1024}{rate} = \frac{(50.000-1.600)*8*1024}{1.000.000}$$

A continuación ejecutamos en PCC1: `$ iperf -s -u`

Y ahora generamos tráfico UDP desde PCA1 con un tamaño de trama de 1472 bytes y un ancho de banda de 2 Mbps que, según hemos visto en la práctica, no está condicionada por el límite de paquetes por segundo del generador:

```
$ iperf –c @IP_servidor –t 10 –u –b 2000000 –l 1472
```

Quitamos el `tc`, para dejar el equipo en la situación anterior:

```
$ tc qdisc del dev eth0 root tbf rate 1.0mbit limit 50kb burst 1.6kb
```

Por último vamos a generar tráfico UDP desde PCA1 con diferentes tamaños pequeños de trama (100, 200 y 300 bytes) y ancho de banda de 200 kbps. Pero antes modificamos `tc` para que en el *burst* quepa sólo una trama (Por ejemplo, para 100 bytes, podemos usar `0.15kb` y así aseguramos que cabe sólo una trama. esto es importante porque si no, el `tc` transmite con otros valores). Este es el ejemplo para 100 bytes:

```
$ tc qdisc add dev eth0 root tbf rate 100kbit limit 50kb burst 0.15kb

$ iperf –c @IP_servidor –t 10 –u –b 200000 –l 100
```

Cuestión 8 tipo T. Calcula teóricamente el máximo ancho de banda total, el útil a nivel IP y el útil a nivel de aplicación, que se conseguiría en el escenario propuesto para los casos propuestos de generación de tráfico UDP y teniendo en cuenta que suponemos la transmisión sobre tecnología Ethernet 10Mbps y lo expuesto en la nota de la cuestión 7 tipo T.

Cuestión 9 tipo T. En base a la captura en los interfaces LANA y LANC, del tráfico generado por *iperf,* calcula el ancho de banda total, el útil a nivel IP y el útil a nivel de aplicación. Analiza si hay pérdidas de tramas y retardo en la transmisión.

Si se considera necesario para una mejor comprensión, se pueden modificar los parámetros de `tc` e `iperf` y repetir el cálculo de anchos de banda.

Anexo I. Manuales de configuración y programas de análisis

Configuración de NAT: `iptables`

`iptables` es el comando utilizado para configurar en Linux tanto el filtrado de paquetes (firewall) como la traducción de direcciones (NAT, Network Address Translation[2]). A continuación se muestra una breve descripción de su funcionamiento. No obstante, dadas las múltiples funciones que proporciona esta herramienta y, por lo tanto, la complejidad asociada, se especifican únicamente los comandos necesarios en relación a su funcionalidad como NAT. La información completa puede consultarse en la página principal de **`iptables`** (https://linux.die.net/man/8/iptables), donde pueden encontrarse diversos tutoriales y manuales de ayuda. Igualmente, pueden consultarse otras web:

http://www.netfilter.org/projects/iptables/index.html

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch14_:_Linux_Firewalls_Using_iptables

Funcionamiento básico de iptables

Todos los paquetes inspeccionados por **`iptables`** pasan por una secuencia de **tablas** (**`table`**) para su procesado. Cada una de estas tablas se dedica a un tipo particular de actividad sobre el paquete y se controla por una cadena (*CHAIN*) específica de transformación o filtrado del paquete.

Hay tres tablas en total, con sus cadenas asociadas:

- ***mangle** table*: responsable de alterar campos específicos del paquete. Tiene tres cadenas:
  - **PREROUTING** *chain:* Altera los paquetes antes de encaminarlos.
  - **INPUT** *chain*: Altera los paquetes destinados al propio equipo.
  - **FORWARD** *chain*: Altera los paquetes que deben ser reenviados a otro equipo.
  - **OUTPUT** *chain*: Altera los paquetes originados por el propio equipo.
  - **POSTROUTING** *chain:* Altera los paquetes tras el encaminamiento.
- ***filter** table*: responsable del filtrado de paquetes, usado para establecer las reglas y políticas de paso del cortafuegos o *firewall*. Tiene tres cadenas en las que se aplican las reglas y políticas (reglas por defecto):
  - **FORWARD** *chain*: Filtra los paquetes que llegan del exterior y deben ser enrutados hacia otro equipo.
  - **INPUT** *chain*: Filtra los paquetes destinados al propio equipo.
  - **OUTPUT** *chain*: Filtra los paquetes originados por el propio equipo.
- ***nat** table*: responsable de la traducción de direcciones (NAT). Tiene dos cadenas asociadas:

[2] RFC 3022, https://tools.ietf.org/html/rfc3022

- **PREROUTING** *chain*: realiza NAT sobre paquetes antes de encaminarlos (cambios sobre la dirección destino, para que el encaminamiento se decida según la nueva dirección).
- **POSTROUTING** *chain*: realiza NAT sobre paquetes tras el encaminamiento (una vez decidida la interfaz de salida o la entrada del paquete al equipo). Se utiliza cuando es necesario modificar la dirección fuente del paquete.

Cuando un paquete entra en el equipo donde se ha implementado **`iptables`** (típicamente un *firewall*), alcanza el hardware y es procesado en el kernel por su driver correspondiente. Después, el paquete empieza a recorrer una serie de etapas en el kernel antes de ser enviado a la aplicación adecuada (localmente), reenviado hacia otro host, o cualquier otra operación.

Si asumimos que el servidor sabe enrutar un paquete, y que las reglas del firewall permiten su transmisión, este sería el orden de las cadenas en las distintas situaciones:

- Paquetes que llegan de fuera, destinados al sistema local:

  PREROUTING -> INPUT

- Paquetes que llegan de fuera, destinados a otra máquina:

  PREROUTING -> FORWARD -> POSTROUTING

- Paquetes generados localmente:

  OUTPUT -> POSTROUTING

De acuerdo a las tablas y cadenas antes mencionadas, el camino de procesado que sigue dicho paquete se corresponde con el esquema de la Figura I.1.

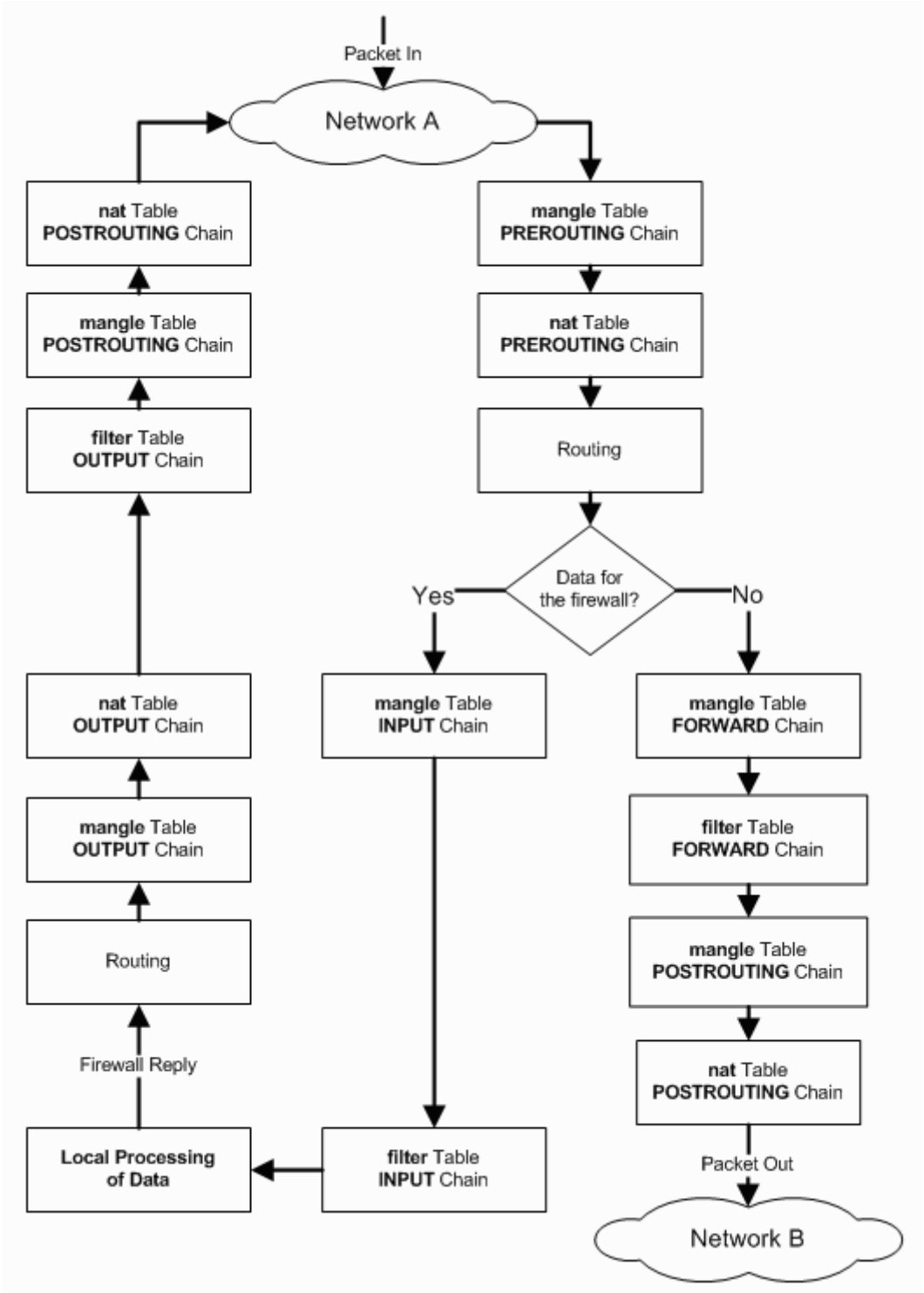


Figura I.1: Camino de procesado que sigue un paquete a través de las tablas y cadenas de *iptables*

El procesado realizado sobre el paquete responde al cumplimiento o no de las reglas configuradas en las diferentes tablas/cadenas. Una regla cualquiera se especifica de la siguiente forma (sin seguir estrictamente dicho orden):

```
$ iptables –t <table> <command> <CHAIN> <matching criteria> –j <target>
```

- **<table>** hace referencia a una de las tablas antes mencionadas (si no se especifica, por defecto será *filter*)
- **<command>** especifica el comando de tratamiento de la regla (**-A** añadir; **-D** borrar, etc.)
- **<CHAIN>** hace referencia a la cadena (**INPUT**, **PREROUTING**, etc.) asociada.
- **<matching criteria>** Criterios que debe cumplir un paquete para aplicarle el objetivo especificado en **<target>**, como por ejemplo, que sea un paquete ICMP, o tráfico TCP web (dirigido al puerto 80, etc.); que tenga determinada dirección fuente; que se dirija a una interfaz de salida en concreto.
- **<target>** Objetivo sobre el paquete (**- ACCEPT** seguir con su procesado; **- DROP** tirarlo; **- SNAT** aplicarle traducción de dirección fuente; etc.)

Criterios típicos (matching criteria):

-p <protocol-type>	Protocolo. Tipo incluye **icmp**, **tcp**, **udp**, y **all**
-p tcp --sport <port>	Puerto fuente TCP. Puede ser uno, o un rango: **pini:pfin**
-p tcp --dport <port>	Puerto destino TCP. Puede ser uno, o un rango: **pini:pfin**
-p tcp --syn	Indicar *flag* SYN de TCP presente (identificar inicio de conexión TCP)
-p udp --sport <port>	Puerto fuente UDP. Puede ser uno, o un rango: **pini:pfin**
-p udp --dport <port>	Puerto destino UDP. Puede ser uno, o un rango: **pini:pfin**
-p icmp --icmp-type <type>	Tipos ICMP más habituales: **echo-reply** and **echo-request**
-s <ip-address>	Dirección IP fuente
-d <ip-address>	Dirección IP destino
-i <interface-name>	Interfaz por la que entra el paquete
-o <interface-name>	Interfaz por la que sale el paquete

Módulos adicionales del kernel requeridos por iptables

Para activar ciertas funcionalidades de *iptables* es necesario incorporar algunos módulos adicionales. Para instalar (desinstalar) dichos módulos, basta la utilización de la herramienta **modprobe (-r)**:

$ modprobe ip_conntrack / modprobe –r ip_conntrack

Carga o elimina el módulo **ip_conntrack**, necesario para hacer seguimiento de las conexiones.
Por defecto está cargado en las máquinas del laboratorio.

$ modprobe ip_conntrack_ftp / modprobe –r ip_conntrack_ftp

Carga o elimina el módulo **ip_conntrack_ftp**, necesario para hacer seguimiento específico de las conexiones FTP (requiere que el módulo **ip_conntrack** esté instalado previamente).
Por defecto está cargado en las máquinas del laboratorio.

$ modprobe iptable_nat
Carga el módulo de **iptables** para la realización de NAT.
Por defecto está cargado en las máquinas del laboratorio.

$ modprobe ip_nat_ftp
Carga el módulo **ip_nat_ftp**, necesario para permitir la utilización de FTP a través de un NAT.
Por defecto no cargado en las máquinas del laboratorio.

Utilización de **iptables** *como NAT*

Dado que nos centraremos en la funcionalidad NAT de **iptables**, a continuación se muestran los ejemplos básicos de configuración, obviando todo lo relativo a reglas de filtrado.

$ iptables –t nat –L <chain>
permite visualizar las reglas existentes en la tabla NAT. Opcionalmente se puede visualizar únicamente la cadena de interés

$ iptables –t nat –F <chain>
permite borrar todas las reglas existentes en la tabla NAT. Opcionalmente se puede visualizar únicamente una cadena.

En ambos casos, la cadena **<chain>** puede omitirse, refiriéndose en ese caso la instrucción a visualizar o borrar todas las cadenas.

Source NAT: modifica la dirección fuente del paquete. La dirección IP pública a asignar se indica explícitamente. Dicha modificación se realiza en la cadena POSTROUTING, justo antes de hacer finalmente el envío (requiere el criterio **–o <interfaz de salida>**). Esto implica que cualquier otra funcionalidad (encaminamiento, filtrado de paquetes) "verá" el paquete sin modificar.

$ iptables –t nat –A POSTROUTING –o eth0 –j SNAT --to 1.2.3.4
cambia la dirección fuente del paquete al valor **1.2.3.4**, antes de enviarlo por la interfaz de salida *eth0*.

$ iptables –t nat –A POSTROUTING –o eth0 –j SNAT --to 1.2.3.4-1.2.3.6
cambia la dirección fuente del paquete a cualquiera de los valores en el rango especificado, antes de enviarlo por la interfaz de salida *eth0*.

$ iptables –t nat –A POSTROUTING –o eth0 –j SNAT –p tcp --to 1.2.3.4:1-1023

cambia la dirección fuente del paquete al valor **1.2.3.4**, y el puerto (para el protocolo TCP) a cualquier valor entre **1** y **1023**, antes de enviarlo por la interfaz de salida *eth0*.

Masquerading: Es un tipo especializado de Source NAT. Su uso está especialmente indicado cuando las direcciones IP públicas se asignan dinámicamente (como en una conexión ADSL). No requiere la identificación explícita de la dirección, ya que usará como dirección fuente la existente en la interfaz de salida. Además, si la conexión cae, se olvidan las asociaciones realizadas, lo que facilita la recuperación una vez se recupera una nueva IP.

$ iptables –t nat –A POSTROUTING –o ppp0 –j MASQUERADE
modifica cualquier dirección origen por la dirección existente en la interfaz **ppp0**.

Destination NAT: En este caso, la modificación de direccionamiento IP se realiza en la cadena PREROUTING, justo antes de que el paquete entre en el resto de funcionalidades de procesado del paquete (encaminamiento, filtrado de paquetes). Dichas funcionalidades "verán" por lo tanto el paquete dirigido a su destino "real". Requiere el criterio **–i <interfaz de entrada>**.

$ iptables –t nat –A PREROUTING –i eth0 –j DNAT --to 5.6.7.8
cambia la dirección destino del paquete (recibido de la interfaz de entrada *eth0*) al valor **5.6.7.8**

$ iptables –t nat –A PREROUTING –i eth0 –j DNAT --to 5.6.7.8-5.6.7.10
cambia la dirección destino del paquete (recibido de la interfaz de entrada *eth0*) a cualquiera de los valores en el rango especificado.

$ iptables –t nat –A PREROUTING –i eth0 –p tcp --dport 80 –j DNAT --to 5.6.7.8:8080
cambia la dirección destino del paquete (recibido de la interfaz de entrada *eth0*) relativo a tráfico http (TCP, puerto **80**) al valor **5.6.7.8**, con puerto **8080**.

$ iptables –t nat –A PREROUTING –d 5.6.7.8 –p tcp --dport 80 –j DNAT --to 192.168.10.1
cambia la dirección destino del paquete (dirigido a **5.6.7.8:80**, a la salida del NAT) al valor **192.168.10.1**, encaminándolo por tanto después a dicha dirección privada.

Consulta del contenido de las tablas de NAT dinámico

Se trata de consultar el contenido de las tablas de NAT que permiten hacer la traducción de direcciones dinámica. Para ello se ejecuta el comando:

$ cat /proc/net/nf_conntrack