Capítulo 7

Configuración da rede

Para expoñer os contidos deste capítulo imos supoñer que o alumno coñece xa os fundamentos de redes de ordenadores e imonos centrar na configuración da rede en Linux e sobre todo na configuración de equipos Linux como *router* e como cortalumes, facendo filtrado de paquetes.

Non obstante imos adicar a primeira sección a repasar algúns conceptos de TCP/IP [1] que logo necesitaremos nas seccións seguintes.

7.1. Conceptos básicos de TCP/IP

O modelo teórico no que se basa o funcionamento das actuais redes de ordenadores e Internet é o modelo OSI [2]. Deste modelo de 7 capas, para o que vamos a tratar neste capítulo interésanos sobre todo as capas 2, 3 e 4.

A capa 2 é a de "enlace de datos" e ocúpase do direccionamento físico, do acceso ao medio. Cada ordenador ou equipo conectado á rede denominarase "nodo" (*host*). No caso de redes *ethernet* cada nodo terá unha "dirección física" denominada MAC, un número de 48 bits grabado polo propio fabricante da tarxeta de comunicacións e polo tanto único.

A capa 3 é a de *rede* é ocúpase do *enrutamento* que permite enviar paquetes dunha rede a outra, pasando polas redes intermedias que sexa preciso. Para posibilitar isto cada nodo "lóxico" terá asociado unha "dirección de Internet" (IP), un conxunto de catro números no rango 0-255. Os equipos que permiten inter-

[1] http://bibing.us.es/proyectos/abreproy/11499/fichero/04+-+Administracion+de+la+red+en+GNU-Linux.pdf

[2] https://es.wikipedia.org/wiki/Modelo_OSI

conectar redes diferentes denomínanse "routers" (*gateway*). Cada equipo soe ter unha dirección IP pero tamén un nome (*hostname*) asociado. O sistema que se encarga da organización xerárquica dos nomes, e das bases de datos que permiten traducir *hostnames* a direccións IP e viceversa denomínase DNS (*domain name system*).

A capa 4 é a de *transporte* e ocúpase de levar os datos de orixe a destino, independentemente do tipo de redes físicas polas que pasen. Os conceptos co que traballa esta capa son o de "segmento" e "datagrama", según que empreguemos o protocolo TCP ou UDP. Relacionado con esta capa está tamén o concepto de "porto". Cada porto identifícase por un número enteiro e podemos velo como un "buzón" mediante o cal unha aplicación pode enviar ou recibir paquetes de datos.

En resumo, para establecer unha comunicación entre dous ordenadores conectados a Internet precisamos coñecer as súas direccións IP e os números de porto en ambos extremos. As veces a combinación de IP+porto denomínaselle *socket*. Cando unha apliación quere enviar información a outro equipo, do cal coñece a IP, o primeiro que ten que determinar é se ese equipo é da súa propia rede local ou non.

Se o equipo é da súa rede local terá que averiguar cal é a súa dirección MAC, porque a comunicación a baixo nivel nas redes *ethernet* faise de MAC a MAC. Para obter estas correspondencias IP-MAC soese empregar o protocolo ARP (*address resolution protocol*) polo cal un equipo pode lanzar unha menxe a todos os equipos da rede (*broadcast*) preguntando "¿quén ten esta dirección IP?" ao cal o equipo concreto responderá "son eu, está é a miña dirección MAC". E xa poden empezar a comunicarse. Os ordenadores soen gardar as equivalencias IP-MAC coñecidas nunha "taboa de enrutamento" durante un tempo, para non ter que estar facendo a mesma pregunta cada vez.

Se o equipo non está na mesma rede local o que fai é enviarlle os paquetes ao router (*gateway*), que será o responsable de indagar fora desa rede local ata atopar o ordenador coa IP indicada. Unha vez atopado será posible establecer un camiño, a través dos routers que sexa preciso para que os datos viaxen ata o destino.

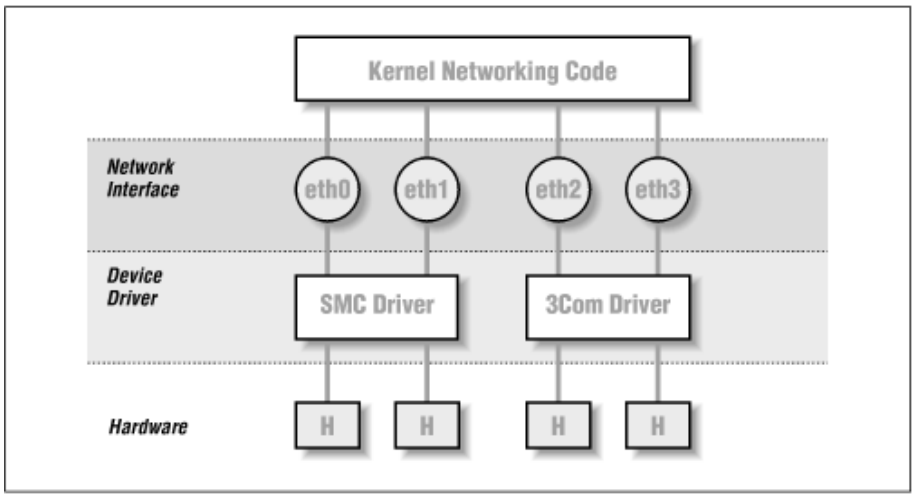

Figura 7.1: Relación entre controladores, interfaces y hardware (sacado de [3])

7.2. Configuración da rede

Cando o ordenador arranca unha das tarefas que leva a cabo é a detección do hardware conectado ao mesmo, entre eles as tarxetas de rede. No caso dos ordenadores con Ubuntu actualmente é o sistem *udev* o que se encarga desta detección e de escribir en determinados arquivos a información necesaria para definir a *interface de rede* asociada.

Na figura 7.1 representase gráficamente o proceso. A partires dunha tarxeta física, é preciso que o *kernel* conte co correspondente *driver* para poder enviar e recibir datos da mesma. Cada tarxeta levará asociado unha *interface de rede*, cun nome único, que será a que empregarán as capas superiores para xestionar a comunicación a través de dita tarxeta.

A asignación de nomes aos diferentes interfaces de rede cambiou recentemente, a lo menos en Ubuntu. Anteriormente as interfaces *ethernet* denominábanse *eth0*, *eth1*, etc. Actualmente emprégase un esquema denominado *predictable network interface names* [4] que soluciona certos problemas do esquema anterior.

Por exemplo, no meu equipo, a tarxeta de rede denomínase *enp1s0* que significa que se trata dunha tarxeta *ethernet*, conectada no bus PCI 1 e no *slot* 0.

Outra información que precisamos para configurar a rede é o propio nome

[3] http://es.tldp.org/Manuales-LuCAS/GARL2/garl-2.0.pdf

[4] https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-understanding_the_predictable_network_interface_device_names

do equipo, polo cal será coñecido na red. Ese nome estará definido no arquivo */etc/hostname*.

Por último precisamos indicar a dirección IP e o resto de parámetros de rede precisos: máscara de rede e IP do *gateway* ou porta de enlace. Tradicionalmente esta configuración atopábase no arquivo */etc/network/interfaces*, pero por exemplo o meu ordenador so contén o seguinte:

```
auto lo
iface lo inet loopback
```

Como vemos so contén información da conexión denominada *loopbak*, que é unha interface *virtual* do equipo consigo mesmo. Isto é debido a que actualmente Ubuntu emprega un servizo denominado *Networkmanager* [5] para xestionar a configuración da rede.

Iso fai o que a información que debería estar no arquivo */etc/network/interfaces*, agora aparece na carpeta */etc/NetworkManager/system-connections/*. En concreto no meu equipo existe un arquivo denominado *Profile 1* que contén o seguinte:

```
[connection]
id=Profile 1
uuid=92ef64ee-a6fd-4011-8731-c05162a414dc
type=ethernet
permissions=

[ethernet]
cloned-mac-address=40:B0:34:46:CD:BE
mac-address=40:B0:34:46:CD:BE
mac-address-blacklist=

[ipv4]
address1=193.147.25.120/24,193.147.25.1
dns=193.146.32.86;193.146.86.228;
dns-search=
method=manual
```

[5] https://help.ubuntu.com/community/NetworkManager#User_Settings_and_System_Settings

Entre a información que contén o arquivo está a dirección MAC da tarxeta ethernet. Na parte de *ipv4*, no campo *address1* temos a dirección IP do equipo (193.147.25.120) e a dirección da porta de enlace *193.147.25.1*. Aínda que parece que non se explicita a máscara de rede, o */24* da dirección de rede estanola indicandola de xeito indirecto, serían 24 bits a 1 e o resto a 0 (255.255.255.0).

É posible desactivar o servizo *Networkmanager* se preferimos retornar a xestión por defecto de Linux. En Ubuntu fariamos:

```
systemctl stop NetworkManager.service
systemctl disable NetworkManager.service
```

Nese caso teriamos que modificar o arquivo */etc/network/ interfaces* para que recolla a configuación desexada. Un exemplo típico sería o seguinte.

```
iface eth1 inet static
address 192.168.1.11
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
dns-nameservers 80.58.61.254
```

Para completar este apartado imos mencionar outros arquivos de configuración que conteñen información de interese para as comunicacións por rede. Un deles é o arquivo */etc/nsswitch.conf*, que é un arquivo de configuración das bases de datos do sistema. En concreto interésanos a liña que comeza pola palabra "hosts". Será algo como isto:

```
hosts:     files mdns4_minimal [NOTFOUND=return] dns myhostname
```

que indica a orde en que se buscará os nomes asociados a IPs e viceversa. Vemos que en primeiro lugar se empregará o método "files". Eso significa que se vai a empregar o arquivo */etc/hosts* que conterá pares de valores *IP-nome*. No meu equipo ese arquivo so contén a liña

```
127.0.0.1 localhost
127.0.1.1 anton-PC1
```

Se o nome ou IP que estamos buscando non é ningún deses empregaremos o "dns"que accederá a xerarquía de servidores DNS para averiguar esa información. O método "mdns4_minimal", corresponde ao *multicast DNS* , do que non imos a falar aqui.

Outro dos arquivos a ter en conta é o */etc/resolv.conf* que debería conter as IPs dos servidores DNS ao cal consultar, un exemplo podería ser o seguinte:

```
nameserver 80.58.32.220
nameserver 80.58.31.250
```

Nas versións actuais de Ubuntu, como xa comentamos esa información está tamén no carpeta */etc/NetworkManager/system-connections/*.

7.2.1. Configuración da rede: comando ip

Ata fai uns anos a xestión da rede en Linux facíase empregando o paquete *net-tools* que contiña comandos como *ifconfig, netstat, route* ou *arp*. Actualmente a maioría de distribucións optaron por mudarse a un novo paquete, denominado *iproute2* [6] [7], moito mais avanzado.

Neste apartado imos explicar o funcionamento do comando principal do paquete, denominado *ip*, que engloba o funcionamento dos anteriormente mencionados. A sintaxe deste comando é a seguinte:

```
ip [ OPCIÓNS ] OBXECTO [ COMANDO [ ARGUMENTOS]]
```

As opcións son comúns a todos os obxectos, algunhas delas son:

- -s: da mais "estatísticas", mais datos.
- -r: traduce as IPs a nomes.
- -j: saca a información en formato JSON.

A parte fundamental do comando é o "obxecto" que poderiamos entendelo como o "aspecto" da rede sobre o que queremos preguntar ou modificar, que soe ter relación cunha determinada taboa. A continuación imos repasar os "obxectos" mais importantes.

[5] https://albertomolina.wordpress.com/2008/07/07/utilizando-mdns-en-una-red-local/
[6] https://www.zeppelinux.es/como-utilizar-iproute2-en-lugar-de-ifconfig/
[7] https://es.wikipedia.org/wiki/Iproute2

address

Empregaremos este obxecto parta obter información das direccións IP asociadas aos diferentes dispositivos [8]. Por exemplo:

```
ip address show
```

Neste caso *show* é o "comando", que pode levar argumentos. Por exemplo, para obter a IP asociado a interfaz *enp1s0* fariamos:

```
ip address show enp1s0
```

obtendo unha salida como a seguinte:

```
2: enp1s0: ... mtu 1500 qdisc fq_codel state UP ...
   link/ether 40:b0:35:47:cd:be brd ff:ff:ff:ff:ff:ff
   inet 193.147.24.114/24 brd 193.147.24.255 ...
```

Actualmente é posible asociar mais dunha dirección IP ao mesmo interfaz. O equipo recibirá os paquetes dirixidos a todas as IP. Para engadir unha segunda dirección IP executariamos o seguinte:

```
ip address add 193.147.24.115/24 dev enp1s0
```

O comando oposto, para quitar unha dirección IP sería *delete*.

O comando de *net-tools* equivalente a este sería o *ifconfig*

neighbour

Este comando emprégase para acceder a taboa de "veciñanza", ou sexa á taboa ARP, a que contén os pares dirección IP - dirección MAC coñecidas. O comando de *net-tools* equivalente é *arp*.

Para ver o contido da táboa ARP facemos:

```
ip neighbour show
```

co cal obtermos algo como o seguinte:

[8] https://www.baturin.org/docs/iproute2/#Table%20of%20contents

```
193.147.24.1 dev enp1s0 lladdr 68:bc:0c:2d:68:43 REACHABLE
193.147.24.209 dev enp1s0 lladdr 40:b0:35:40:d9:3f DELAY
193.147.24.245 dev enp1s0 lladdr 00:1f:f2:42:6b:2a STALE
193.147.24.212 dev enp1s0 lladdr 18:d6:cb:02:42:2f REACHABLE
```

O último campo danos información de interés [9] pero que non imos explicar aqui para non alongarnos.

De igual modo que no caso anterior temos os comandos *add* e *delete* para engadir manualmente unha dirección a táboa, ou para eliminala.

link

Este é un comando moi potente que se emprega para ver e configurar aspectos físicos e lóxicos da rede. Aqui so imos comentar o mais básico. Para ver os datos básicos das conexións empregamos o argumento "show" ou "list".

```
ip link show
```

co que obtemos algo como:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state ...
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc  ...
link/ether 40:b55:35:56:ca:bb brd ff:ff:ff:ff:ff:ff
```

Tamén nos sirve para activar ou desactivar unha interface cos argumentos "dev XXX up/down". Por exemplo:

```
ip link dev enp1s0 down
```

Outras cousas que permite facer: cambiar o nome da MAC, o nome do interface, permitir ou non ARP, crear *bridges interfaces*, etc.

route

O último comando que imos ver, aínda que hai mais, será "route", que sustitue precisamente ao comando *route* de *net-tools*. Este comando permite visualizar e modificar a táboa de enrutamento da rede. Para vela facemos:

[9] https://www.ccexpert.us/routing-tcp-ip-2/neighbor-unreachability-detection.html

```
ip route show
```

co cal obteremos algo como:

```
default via 193.147.12.1 dev enp1s0 proto static metric 100
169.254.0.0/16 dev enp1s0 scope link metric 1000
193.147.12.0/24 dev enp1s0 ... scope link src 193.147.87.105
```

No que nos indica a través de que interface (*enp1s0*), en que ámbito (*link*) e con que dirección IP fonte se enviarán os diferentes paquetes. A primeira liña indica a ruta por defecto para as IPs descoñecidas na rede (via 193.147.12.1). Na seguinte sección explicaremos mellor isto.

Tamén se poden engadir novas rutas co argumento "add" ou eliminalas con "delete". Vexamos un exemplo:

```
ip route add 192.168.1.0/24 via 192.168.1.1
```

que engade unha ruta para a subrede indicada a través (via) a porta de enlace 192.168.1.1

Para rematar un último exemplo. Para coñecer a routa concreta cara aunha IP determinada podemos empregar o argumento "get".

```
ip route get 193.13.1.1
```

obtendo no meu caso:

```
193.13.1.1 via 193.147.12.1 dev enp1s0 src 193.147.12.105 uid 1000
```

que indica que os paquetes serán enviados a porta de enlace 193.147.12.1

7.2.2. Obter información da rede: comando ss

O comando *ss*[10] é o substituto de *netstat* nas *net-tools*. Emprégase para obter información de tipo estatístico sobre as conexión existentes. A sintaxe mais básica, simplemente *ss*, obten un listado con todas as conexións establecidas indicando os portos orixe e destino.

Este listado soe ser moi extenso, pero podemos restrinxilo mediante opcións como "-t" para incluir so as conexión de tipo TCP, "state connected" para excluir

[10] https://www.linux.com/tutorials/introduction-ss-command/

as conexións pechadas ou en espera, "src IP" ou "dst IP" para filtrar so as que proveñan ou vaian cara determinadas IPs.

Outras opcións interesantes son "-r" que permite resolver as direccións IP e poñer no seu canto os nomes dos equipos e "-s" que nos amosa unhas estatísticas globais. Vexamos un exemplo.

```
ss -rst state established src 193.147.87.105
```

obtendo como resultado

```
Total: 1503
TCP:   66 (estab 26, closed 9, orphaned 0, timewait 1)

Transport       Total     IP        IPv6
RAW             1         0         1
UDP             10        6         4
TCP             57        49        8
INET            68        55        13
FRAG            0         0         0

Recv-Q  Send-Q      Local Address:Port          Peer Address:Port
0       0           pc1.esei.uvigo.es:41920    ws-in.1e100.net:https
0       0           pc1.esei.uvigo.es:58906    mad07s10.1e100.net:https
0       0           pc1.esei.uvigo.es:57374    40.74.32.146:https
```

7.3. Router Linux

Nas conexións entre ordenadores as veces distínguense tres tipos de "ambito" [11]. Un é o do propio *host* e corresponde as conexións do equipo consigo mesmo, empregando diferentes portos. É o coñecido *loopback* do que xa falamos antes. O segundo sería o dos equipos conectados á propia rede local. Nese caso o equipo pode conectarse directamente ao destino sin mais que averiguar a súa dirección MAC empregando o protocolo *ARP*.

Cando un equipo indica como dirección IP de destino unha que non esté na súa rede local, non é posible obter a MAC de destino, e por iso eses paquetes son enviados á "porta de enlace" ou *gateway*, que será un equipo que terá outras

[11] http://linux-ip.net/html/routing-intro.html

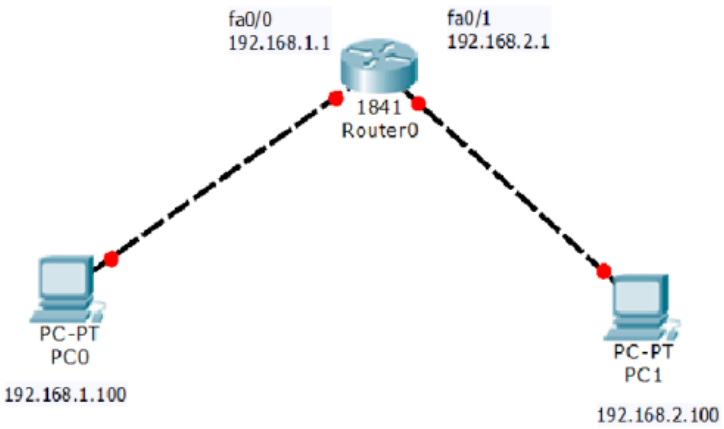

Figura 7.2: Esquema de redes interconectadas por un router (sacado de [12])

conexións, ademais da que ten coa nosa rede local, a través das cales se supón que o paquete poderá chegar ao su destino.

Na figura 7.2 amósase un esquema con un router interconectado dúas redes locais, supoñamos que sexan (192.168.1.0/24) e (192.168.2.0/24).

Chamaremos *router* a calquera equipo conectado a duas ou mais redes, capaz de encamiñar paquetes dunha rede a outra, empregando información da capa de rede, ou sexa direccións IP. Comercialmente existen dispositivos específicos para operar como router pero calquera máquina Linnux, con duas tarxetas de rede pode operar como tal. Nesta sección imos explicar como se configura un deses equipos.

Para que un equipo poda configurarse como router requírese un paso previo: activar no mesmo o reenvio de paquetes IP (*IP forwarding*), ou sexa a posibilidade de recibir paquetes IP e reenvialos cara outras redes, modificando para iso os atributos precisos. Nos equipos actuais iso faise editando o arquivo */etc/sysctl.conf* e descomentando a liña

```
net.ipv4.ip_forward = 1
```

O seguinte paso é incluir nova reglas de enrutamento en ambas interfaces de rede para que os paquetes de cada rede, que non se podan resolver na mesma, sexan enviados polo router a outra rede. Poderiamos facer o seguinte:

```
ip route add 192.168.1.0/24 dev fao/0
```

[12] https://linuxtiwary.com/2015/03/19/how-to-connect-two-different-network-using-router/

```
ip route add 192.168.2.0/24 dev fao/1
```

Hai que advertir que as regras así escritas perderanse no momento en que se apague o ordenador. Se queremos que sexan permanentes podemos escribilas no propio arquivo */etc/network/interfaces*. Nese caso incluiriamos as liñas:

```
up ip route add 192.168.1.0/24 dev fao/0
up ip route add 192.168.2.0/24 dev fao/1
```

Dende logo este é o caso mais sinxelo. Un caso habitual, por exemplo, é o dos routers que conectan unha rede local con Internet. Neses casos é preciso indicar que os paquetes que vaian para as IPs descoñecidas sexan enviados "via" o *gateway* cara Internet. Neses casos veremos regras como a seguinte:

```
ip route add default via 205.7.14.18 dev enp1s1
```

que indica que a ruta por defecto para todas as IPs descoñecidas será enviar os paquetes polo dispositivo "enp1s1", no cal se supón que o noso equipo ten a dirección 205.7.14.18.

Ata agora todas as regras vistas empregaban como criterio para o encamiñamento dos paquetes a dirección destino dos mesmos. Actualmente as redes son moi complexas polo cal este criterio volveuse insuficiente. Esto, unido as posibilidades que ofrecen os actuais nucleos de Linux, fai que sexa posible construir routers en Linux, capaces de encamiñar paquetes empregando outros criterios como a dirección de orixe dos mesmos ou calquera outro atributo do paquete.

Mais aínda, como veremos, será posible inspeccionar os paquetes recibidos para establecer regras de encamiñamento baseadas en calquera campo de información dos mesmos, e o mais potente, alterar ditos campos de información.

Algunhas destas funcións poderían facerse co propio comando *ip route*, pero outras non. Por iso, imos introducir outro paquete dispoñible en Linux deseñado específicamente para interceptar e manipular paquetes de rede chamado *Netfilter* [13]. A ferramenta mais coñecida deste paquete é a denominada *iptables* [14]. Esta é unha ferramenta que permite ao usuario definir regras para especificar a política de filtrado dos paquetes que circulen pola rede.

[13] https://es.wikipedia.org/wiki/Netfilter
[14] https://wiki.archlinux.org/index.php/Iptables_(Espa%C3%B1ol)

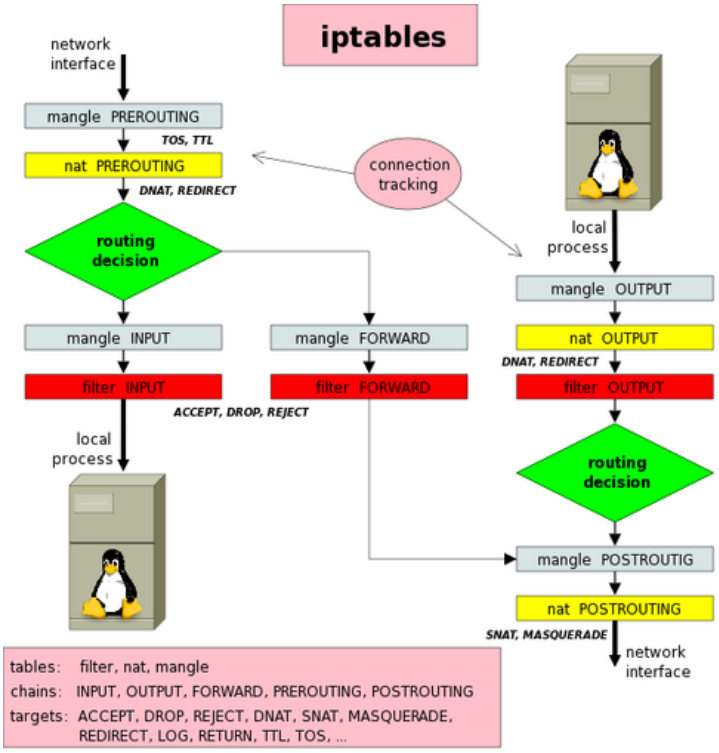

Figura 7.3: Esquema de funcionamento de Iptables (sacado de [15])

7.4. Iptables

Iptables permite definir "regras" para decidir que facer cos paquetes de datos. As regras agrúpanse en "cadeas" e estas en "taboas". Cada táboa está asociada a un tipo diferente de procesamento dos paquetes.

Existen 3 táboas:

- Táboa de filtros (*filter table*). Contén regras de filtrado para decidir que paquetes pasan ou non.
- Táboa de traducción de direccións de rede (*NAT table*). Contén regras para cambiar as direccións IP ou portos de orixen ou destino.
- Táboa de modificación dos paquetes (*mangle table*). Permite modificar outros datos do paquete.

No esquema amosado na figura 7.3 aprecianse as regras *filter* en roxo, as *NAT* en amarelo e as *mangle* en gris.

Falemos agora das cadeas (*chains*). Hai 5 tipos de cadeas.

[15] http://grimore.org/fuss/iptables

- *PREROUTING*. Contén regras que se aplican antes de tomar a decisión sobre o encamiñamento do paquete. Como se aprecia na figura pode conter unha táboa *mangle* e/ou *NAT* (para facer DNAT ou "destination nat" como veremos despois.

- *INPUT*. Se as regras de encamiñamento deciden que ese paquete é para o noso sistema este pasará a través da cadea INPUT, que pode conter unha táboa *mangle* e outra *filter* .

- *FORWARD* Pola contra, se as regras de ecamiñanto deciden que ese paquete debe ser reenviado a outro equipo, antes de facelo pasan polas regras da cadea FORWARD, tanto dunha posible táboa *mangle* como *filter*.

- *OUTPUT*. Os paquetes de saída do noso equipo, antes de pasar pola regras de encamiñamento pasarán pola cadea OUTPUT, que pode conter táboas dos 3 tipos posible.

- *POSTROUTING*. Os paquetes de saída, despois de pasar as regras de encamiñamento, antes de sair definitivamente do noso equipo, aínda pasan pola cadea POSTROUTING que poderá conter táboas *mangle* e *NAT*.

O funcionamento é similiar en todas as cadeas. Cando o paquete chega á cadea vai pasando secuencialmente polas regras da mesma. Se cumpre as propiedades indicadas na regra aplícaselle as condicións de destino indicadas na mesma e xa non se miran o resto de regras da cadea. En caso de que non se cumpra ningunha regra aplicaráselle ao paquete a "política por defecto " (*default policy*), que poderá ser aceptar (*accept*) os paquetes ou descartalos (*drop*).

Fagamos unha parada antes de seguir. Para ver as regras definididas actualmente no noso equipo podemos facer:

```
iptables -L
```

obtendo un resultado como:

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Non temos ningunha regra definida pero temos a política de aceptar todo. Se quixesemos cambiar a política por defecto en algunha cadea poderiamos facer por exemplo:

```
iptables --policy INPUT DROP
```

A política por defecto sería como a última regra da cadea, aquela que se aplica se un paquete non cumpre as condición de ningunha regra.

Falemos agora dos posibles destinos que se poden especificar nunha regra. Os mais comúns son:

- *ACCEPT* (aceptar). Acéptase o paquete.
- *DROP* (descartar). Descártase o paquete sin mais, sin notificalo ao remitente.
- *REJECT* (rechazo). Recházase o paquete, notificándollo ao remitente. Como se indica na figura 7.3 estas son regras típicas das táboas de filtrado.
- *DNAT*. Cambia a dirección IP de destino do paquete, e opcionalmente o porto. Se nos fixamos na figura 7.3 vemos que esto so se pode facer antes do encamiñamento.
- *SNAT*. Cambia a dirección IP de orixe do paquete, e opcionalmente o porto. Só se pode empregar na cadea POSTROUTING.
- *MASQUERADE*. Esta é unha variante de SNAT que se emprega por exemplo coa conexión a un proveedor de internet que nos da unha IP dinámica. En vez de especificar unha dirección IP determinada esta obterase a partires da dirección de IP do interfaz de rede.

Hai que comentar que tanto para SNAT como MASQUERADE, o router ten que manter información sobre a IP real de orixe, porque, cando cheguen paquetes de volta, deberá ser capaz de "reconstruilos", poñendolle a IP real e enviandos pola interface de rede correspondente, para que cheguen ao remitente.

Faltanos comentar unha característica moi interesante de Iptables. Se volvemos a mirar a figura 7.3 vemos que aparece a palabra *connection tracking* enlazando a entrada e a saída. Eso significa que o sistema é capaz de manter información sobre o estado das conexións, para poder indicar que determinada

regra se aplique aos paquetes que están nun determinado "estado". Esto vais mais alá do comentado no parágrafo anterior. Os posibles estados son [16]:

- *NEW*. Un paquete será etiquetado con ese estado se é o primeiro dunha conexión establecida entre dous equipos.
- *ESTABLISHED*. Aplícase aos seguintes paquetes desa conexión. Eso pode facerse se o router leva conta das conexión establecidas en cada momento.
- *RELATED*. Refírese a conexión "derivadas" dunha conexión xa establecida. Eso pode pasar por exemplo nunha conexión FTP, que emprega dous canles simultáneos.
- *INVALID*. Paquetes que non se poden asignar a outro estado por diversas razóns. Soe sor boa idea rechazar todos os paquetes deste tipo.

Se queremos que as regras que escribimos permenezan cando se reinicie o equipo debemos empregar o comando:

```
/sbin/iptables-save > file
```

Sendo file o arquivo onde se gardarán as regras. Por exemplo en Ubuntu suxirese que sexa */etc/iptables.rules*. A continuación deberemos colocar en algun script que se execute o iniciar o comando:

```
/sbin/iptables-restore < file
```

A continuación imos a presentar unha serie de regras a modo de exemplo (sacadas de [17]).

```
iptables -A INPUT -i lo -j ACCEPT
```

Para engadir novoas regras podemos empregar o comando "-A" (*append* que engadirá a regra ao final ou "-I [num]" para coloca ao principio ou diante da regra número [num]. Neste caso imos engadila ao final da cadena INPUT. O parámetro "-i lo" indica que a regra se aplique aos paquetes que veñan do interfaz de rede "lo". As regras sempre rematan con "-j TARGET" que indican que facer cos

[16] https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-firewalls-iptables_and_connection_tracking

[17] https://www.hostinger.com/tutorials/iptables-tutorial

paquetes que cumplan os requisitos da regra, neste caso aceptalos. En definitiva a regra indica que se acepten todos os paquetes do *loopback*.

Se queremos permitir o tráfico de entrada para HTTP (80), HTTPS (443) e SSH (22) podemos empregar as opcións "-p tcp" (protocolo TCP) e "-dport n" (*destination port*).

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

Tamén podemos poñer condicións baseadas na IP de orixe. Por exemplo, para descartar os paquetes procedentes dunha determinada IP faremos:

```
iptables -A INPUT -s 192.168.1.3 -j DROP
```

Finalizamos cun exemplo mais complexo (sacado de [18]). Neste caso é un script completo que, ao executalo, configura as regras para activar un cortalumes.

```
#!/bin/bash

iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -m conntrack --ctstate NEW ! -i eth1 -j ACCEPT
iptables -A FORWARD -i eth1 -o eth0 -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
iptables -A FORWARD -i eth1 -o eth1 -j REJECT
iptables -P INPUT DROP
iptables -P FORWARD DROP
echo "Firewall Activado"
```

A primeira regra indica que se acepta todo o tráfico local. A segunda acepta todo o tráfico relativo a conexións establecidad ou relacionadas. A opción "-m conntrack" indica que imos empregar a extensión de Iptables capaz de seguir os estados das conexións. Con "–ctstate ESTABLISHED,RELATED" indicamos que esta regra se aplica aos paquetes con eses estados.

A terceira regra indica que se acepten as concexións novas (NEW) só se non veñen da interface *eth1* (que se supón que será a interface pública). A cuarta é unha regra para a cadea FORWARD e indica que se acepten todas os paquetes que veñan da interface *eth1* cara *eth0* sempre que correspondas a conexións establecidas. A quinta, outra regra para a cadea FORWARD, indica que se acepten todas as conexións saíntes.

[18] https://es.wikipedia.org/wiki/Netfilter

A sexta é unha regra para a táboa NAT da cadea POSTROUTING, ou sexa para facer "sNAT". Indica que aos paquetes saíntes (-o *eth1*) se lle aplique o destino MASQUERADE, ou sexa, que se lles cambie a súa IP orixe pola do propio router nesa interface de rede.

A séptima indica que se rechacen todos os paquetes saíntes. Fixarse en que esta regra so se aplicará aos que non cumpriron as condicións das regras anteriores. As dúas últimas regras establecen a política por defecto (-P) para as cadeas INPUT e FORWARD, que será rechazar (DROP).