Filtrado de paquetes y QoS

Joan Fuster Monzó
Universidad Politécnica de Valencia

joafusmo@epsa.upv.es

Filtrado de paquetes y QoS
por Joan Fuster Monzó

Este manual pretende ser una guía práctica sobre firewalls en GNU/Linux, y sobre cómo implementarlos en cada caso a través de Netfilter/Iptables.

Por otra parte se tratará de una manera pragmática las posibilidades de QoS que nos ofrece Linux, y que podremos explotar con Iptables/Iproute2.

Tabla de contenidos

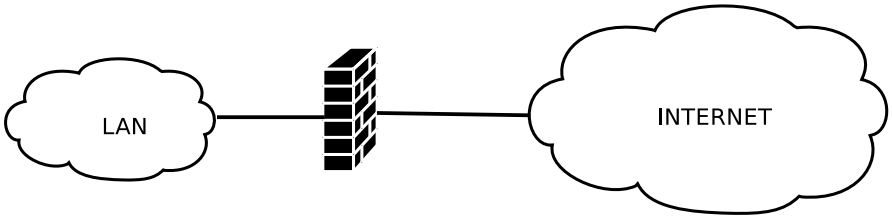
1. Introducción 1
1.1. La solución: Netfilter/Iptables 1
1.2. Términos usados 1
1.3. Arquitecturas de Firewall 2
1.3.1. Screening Router 3
1.3.2. Dual Homed-Host 3
1.3.3. Screened Host 3
1.3.4. Screened Subnet: DMZ 3
2. Configuración del kernel 5
3. Iptables 7
3.1. La tabla filter 7
3.2. La tabla nat 8
3.3. La tabla mangle 8
3.4. Parámetros 9
3.5. Extensiones de coincidencias 9
3.6. Extensiones de objetivo 9
3.7. Manejo de las reglas 13
4. Configuración de Firewalls 15
4.1. Configuraciones básicas 15
4.1.1. Ejemplo 1 15
4.1.2. Ejemplo 2 16
4.2. Configuraciones medias 17
4.2.1. Ejemplo 3 18
4.2.2. Ejemplo 4 19
4.3. Configuraciones avanzadas 22
4.3.1. Ejemplo 5 22
4.3.2. Ejemplo 6 24
5. QoS 29
5.1. Introducción 29
5.1.1. Disciplinas de colas simple 29
5.1.2. Disciplinas de colas con clase 30
5.1.3. Filtros 30
5.2. Reserva de ancho de banda según protocolo 31
5.3. Reserva de ancho de banda por IP 33
5.4. IMQ 35
5.4.1. Configuración 36
5.5. Automatización al inicio 38
A. Monitorización 41
B. Patch-o-Matic 47
C. GNU Free Documentation License 51
C.1. PREAMBLE 51
C.2. APPLICABILITY AND DEFINITIONS 51
C.3. VERBATIM COPYING 52
C.4. COPYING IN QUANTITY 53
C.5. MODIFICATIONS 53
C.6. COMBINING DOCUMENTS 54
C.7. COLLECTIONS OF DOCUMENTS 55
C.8. AGGREGATION WITH INDEPENDENT WORKS 55

C.9. TRANSLATION 55
C.10. TERMINATION 56
C.11. FUTURE REVISIONS OF THIS LICENSE 56
C.12. ADDENDUM: How to use this License for your documents 56
Bibliografía 58

Capítulo 1. Introducción

A dia de hoy, Internet es una red insegura. El número de intrusiones no deseadas cada vez es mayor, y si queremos evitar la fuga/alteración de datos, se hace necesario el uso de un firewall (sobretodo si usamos Windows).

¿Pero qué es un firewall?, básicamente un elemento que conecta dos o más redes y que analiza todo el tráfico que pasa por él. El firewall tendrá definidas unas reglas que se aplicarán a todos los paquetes, y según las cumplan o no, serán aceptados o rechazados.



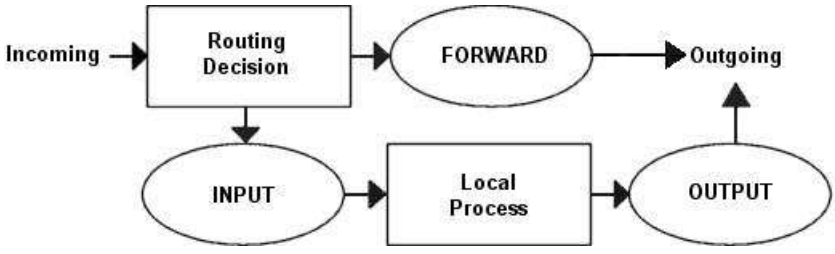
1.1. La solución: Netfi lter/Iptables

Linux nos ofrece como solución al filtrado de paquetes, Netfilter y iptables.

> **Nota:** Netfilter se integra en los kernels 2.4 y 2.6 junto con iptables.En las series 2.2 se utiliza *ipchains* y el las series 2.0 *ipfwadm.*

Toda la gestión de paquetes la lleva el kernel, y Netfilter es el encargado de aplicar las reglas de filtrado que previemente le habremos definido con iptables. En resumen, diremos que Netfilter es la parte del kernel encargada de filtrar tráfico, y iptables la herramienta con la que configuraremos Netfilter.

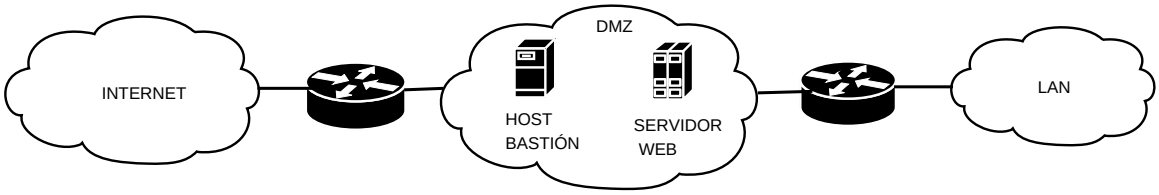
Para entender en funcionamiento de iptables primero deberemos saber cómo funciona el tratamiento de paquetes a nivel de kernel.



Cuando llega un paquete, primero se mira si va destinado a la máquina local, o no. Si está destinado a la máquina local, pasará a la cadena INPUT en caso de ser un paquete entrante, y pasará a OUTPUT en caso de ser un paquete saliente. Si el paquete no es para la máquina local, sino que va destinada a otra máquina y se tiene activado el *forwarding*, entonces pasará a la cadena FORWARD, en caso contrario se eliminará.

1.2. Términos usados

- **DMZ**: Demilitarized Zone (Zona desmilitarizada). La DMZ será una subred que se encontrará entre la red externa y la interna (puede que no). Será otra capa de seguridad antes de llegar al perímetro de seguridad. En esta zona irán ubicados los host bastión y los servidores que queremos que tengan conectividad con el exterior.



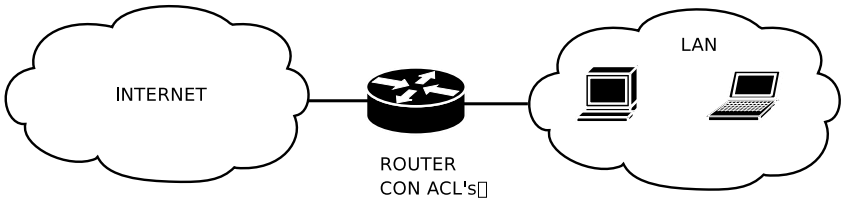
- **Host Bastión**: Punto intermedio entre la zona externa (Internet) y el perímetro de seguridad. Normalmente será un host fuertemente securizado, puesto que está conectado al exterior.
- **DoS**: Acrónimo de Denial of Service (Denegación de Servicio). Supone la pérdida de algún servicio/s en la máquina de la víctima. En caso de ser un servidor seguramente dejará de prestar servicios tales como DNS, Web, mail... Dicho ataque se produce cuando se consigue que la máquina consuma todos sus recursos mediante una tormenta de peticiones de conexión, o mediante un envio masivo de paquetes ICMP (pings).
- **Flooding**: Este procedimiento será el que cause el DoS. Consiste en un lluvia de paquetes que conseguirán consumir los recursos de la máquina. Existen flooding TCP,UDP y ICMP
- **Perímetro de Seguridad**: Zona que se quiere proteger.
- **Spoofing**: Suplantación de una identidad.
- **Escaneo de puertos**: Con un escaneo de puertos lo que conseguimos es escanear con alguna herramienta del tipo NMAP todos los puertos que están disponibles en la máquina a atacar, y saber exactamente los servicios que ofrece, y por que puertos es vulnerable.

```
    snowball:~# nmap -sS -sR -sV -O -PI -PT localhost

Starting nmap 3.50 ( http://www.insecure.org/nmap/ ) at 2004-03-30 15:08 CEST
Interesting ports on localhost (127.0.0.1):
(The 1652 ports scanned but not shown below are in state: closed)
PORT    STATE SERVICE             VERSION
13/tcp  open  daytime
22/tcp  open  ssh                 OpenSSH 3.8p1 (protocol 2.0)
25/tcp  open  smtp                Exim smtpd 3.36
37/tcp  open  time
80/tcp  open  http                Apache httpd 1.3.29 ((Debian GNU/Linux) PHP/4.3.3)
111/tcp open  rpcbind (rpcbind V2) 2 (rpc #100000)
879/tcp open  pmud                pmud (http://sf.net/projects/apmud)
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux 2.5.25 - 2.5.70 or Gentoo 1.2 Linux 2.4.19 rc1-rc7)
Uptime 0.120 days (since Tue Mar 30 12:15:31 2004)

Nmap run completed -- 1 IP address (1 host up) scanned in 11.956 seconds
```

1.3. Arquitecturas de Firewall

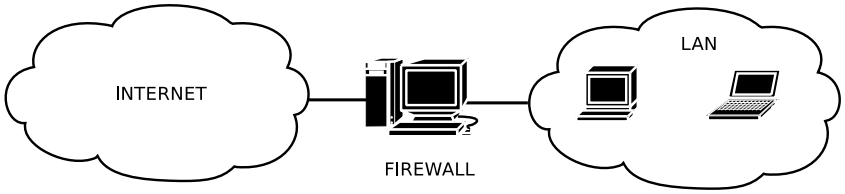
1.3.1. Screening Router

Esta es la arquitectura más simple. Se trata de tener un router que además de desempeñar tareas de routing, tiene implementadas unas ACL (Access Control List), con la que hará un filtrado básico de paquetes.



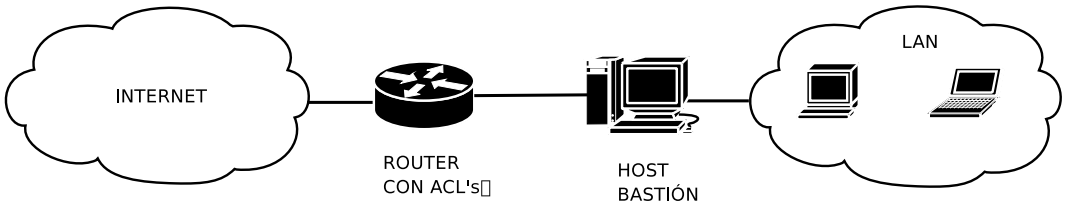
1.3.2. Dual Homed-Host

En este caso, en vez de utilizar un router con ACL's, utilizaremos un host con 2 interfaces de red (o más). El *forwarding* estará desactivado para evitar que las 2 redes se vean directamente. En caso de querer salida a Internet para determinados servicios, se implementará un *proxy* para cada servicio.



1.3.3. Screened Host

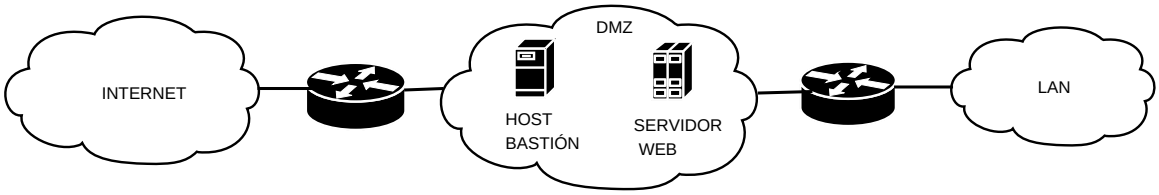
Sería la combinación de los dos anteriores. El filtrado de paquetes lo realizaría el router, mientras que en el host bastión residirían los *proxys* de las aplicaciones.



1.3.4. Screened Subnet: DMZ

Es más conocida por DMZ, y es la más utilizadas de todas las arquitecturas, aunque la más complicada. Una de sus virtudes es que no centraliza la seguridad, es decir, un ataque exitoso al host bastión o al router, no

desmoronará toda nuestra seguridad, ya que aún deberá atravesar otra red.



Capítulo 2. Configuración del kernel

Antes de comenzar a utilizar iptables para configurar nuestro firewall, deberemos configurar el kernel adecuadamente para que soporte el filtrado y unas cuantas opciones más. Así que lo primero será bajarse las fuentes del kernel y configurarlas.

```
snowball:/usr/src# wget ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.3.tar.bz2
snowball:/usr/src# tar xvjf linux-2.6.3.tar.bz2
snowball:/usr/src# ln -s /usr/src/linux-2.6.3 linux
snowball:/usr/src# cd linux
snowball:/usr/src# make menuconfig
```

Una vez tengamos abierta la configuración del kernel, iremos a `Device Drivers` -> `Networking Support` -> `Networking Options` -> `[*] Network Packet Filtering` -> `IP: Netfilter Configuration`

- `<M> Connection tracking (required for masq/NAT)`
- `<M> FTP protocol support`
- `<M> IRC protocol support`
- `<M> IP tables support (required for filtering/masq/NAT)`
- `<M> limit match support`
- `<M> IP range match support`
- `<M> MAC address match support`
- `<M> Packet type match support`
- `<M> netfilter MARK match support`
- `<M> Multiple port match support`
- `<M> TOS match support`
- `<M> Owner match support`
- `<M> Packet filtering`
- `<M> REJECT target support`
- `<M> Full NAT`
- `<M> MASQUERADE target support`
- `<M> REDIRECT target support`
- `<M> Packet mangling`
- `<M> TOS target support`
- `<M> MARK target support`
- `<M> CLASSIFY target support`
- `<M> LOG target support`

Estas serían las opciones más interesantes en cuanto a Netfilter se refiere, ahora pasaremos a configurar el QoS.

Ahora vamos a `Device Drivers` -> `Networking Support` -> `Networking Options` -> `QoS and/or fair queueing`

- `[*] QoS and/or fair queueing`
- `<*> CBQ packet scheduler`
- `<*> HTB packet scheduler`
- `<*> The simplest PRIO pseudoscheduler (NEW)`
- `<*> SFQ queue (NEW)`
- `<*> Ingress Qdisc`
- `<*> QoS support`
- `<*> Rate estimator`
- `<*> Ingress Qdisc`
- `<*> Packet classifier API`
- `<*> Ingress Qdisc`
- `<*> TC index classifier`
- `<*> Routing table based classifier`
- `<*> Firewall based classifier`
- `<*> U32 classifier`
- `<*> Special RSVP classifier`
- `<*> Traffic policing (needed for in/egress)`

Una vez tengamos guardada la configuración del kernel, pasaremos a compilar:

```
snowball:/usr/src/linux# make && make modules_install
```

Una vez hecho esto, cogeríamos la imagen del kernel que se encuentra en `/usr/src/linux/arch/i386/boot/bzImage` y la meteríamos en lilo o grub, reiniciaríamos y ya tendríamos el sistema listo para utilizar.

Capítulo 3. Iptables

Una vez tengamos el kernel compilado con las opciones necesarias, ya podemos pasar a configurar nuestro firewall con iptables. Cuando definamos una regla de filtrado, deberemos especificar unos criterios que se aplicarán al paquete, y en caso de que se cumplan, un objetivo. Es decir, si llega un paquete ICMP (criterio) de la dirección IP x.y.z.w (otro criterio), elimínalo (objetivo). Esa regla cuando la definamos, la meteremos dentro de una cadena predefinida por iptables, o una definida por el propio usuario. Por ejemplo, si queremos evitar que los usuarios de la LAN hagan pings al exterior, definiríamos una regla de filtrado en la cadena de salida de esta forma: si llegan paquetes ICMP (criterio) desde cualquier IP de la LAN (criterio) con destino al exterior (cadena FORWARD), elimínalo (objetivo).

Los objetivos básicos son:

- **ACCEPT**. Acepta el paquete.
- **DROP**. Rechaza el paquete.
- **RETURN**. Cuando un paquete entra en una regla que tiene como objetivo RETURN, pueden suceder dos cosas: si la cadena es una subcadena de otra, entonces deja de examinarse la subcadena y se sigue con la cadena principal; y si la cadena es la principal y tiene como objetivo RETURN, entonces se aplicará la política por defecto, normalmente ACCEPT o DROP.
- **QUEUE**. Pasa el paquete a espacio de usuario, donde otro programa los puede recoger y analizar.

En iptables existen tres tipos de tablas, cada una con unas cadenas internas predefinidas. Dependiendo de lo que queramos hacer, necesitaremos utilizar unas tablas junto con sus cadenas. Los 3 tipos de tablas son: *filter*, *nat* y *mangle*.

En cada tabla, nos podremos encontrar con alguna de estas cadenas predefinidas:

- **INPUT**. Esta cadena se utiliza para los paquetes de entrada.
- **OUTPUT**. Esta cadena se utiliza para los paquetes salientes.
- **FORWARD**. Si los paquetes van destinados a otra máquina que no sea la local, se utilizara esta cadena.
- **PREROUTING**. Se utiliza para el manejo de paquetes antes de que sean enrutados.
- **POSTROUTING**. El manejo de paquetes se realiza después de su enrutamiento.

Atención

El *forwarding* debe estar activado para que podamos utilizar la tabla FORWARD, de lo contrario, todo paquete que tenga un destino distinto de la máquina local será eliminado. Activaremos el forwarding mediante:

```
zen:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

3.1. La tabla filter

Esta es la tabla por defecto de iptables. Es la que utilizaremos para definir reglas de filtrado. Sus cadenas son:

- **INPUT**
- **OUTPUT**
- **FORWARD**

y sus objetivos serán:

- **ACCEPT**
- **DROP**
- **QUEUE**
- **RETURN**

3.2. La tabla nat

Cuando queramos utilizar cualquier tipo de NAT, esta será la tabla a utilizar. Sus cadenas son:

- **PREROUTING**
- **OUTPUT**
- **POSTROUTING**

y sus objetivos son:

- **DNAT**
- **SNAT**
- **MASQUERADING**

3.3. La tabla mangle

En esta tabla podremos modificar parámetros de los paquetes como el TOS (Type Of Service) de IPv4. Está será la tabla que utilizaremos para el marcado de paquetes y posterior priorización para aplicar QoS. Esta tabla contiene todas las cadenas anteriores:

- **INPUT**
- **OUTPUT**
- **FORWARD**
- **PREROUTING**
- **POSTROUTING**

sus principales objetivos son:

- **TOS**
- **MARK**

3.4. Parámetros

Con los siguientes parámetros podremos definir reglas en base a determinadas características.

- **-p**: Tipo de protocolo a tratar entre: *tcp*, *udp*, *icmp* o *all* (los tres).
- **-s**: Dirección origen.
- **-d**: Dirección destino.
- **-i**: Interfaz de entrada.
- **-o**: Interfaz de salida.
- **-j**: Objetivo para una regla.

> **Nota:** Tanto en los parámetros como en las extensiones de coincidencia, podemos utilizar el símbolo **!** para invertir la regla. Este ejemplo, aceptará cualquier protocolo que NO sea *icmp*.
>
> ```
> zen:~# iptables -A INPUT -p ! icmp -j ACCEPT
> ```

3.5. Extensiones de coincidencias

Una forma de afinar nuestra configuración con iptables es usando los módulos de extensiones de coincidencias. Con la opción `-p` podemos especificar el protocolo a tratar como TCP, ICMP o UDP y hacer una clasificación más exaustiva en cuanto a flags TCP, tipo de ICMP, puertos UDP...

Otra forma es mediante la opción `-m` que cargará el módulo en cuestión junto con sus opciones listas para utilizar. Esta opción es bastante útil ya que podemos utilizar tanto los módulos que vienen *de serie* con iptables, como otros que van apareciendo a posteriori.

3.6. Extensiones de objetivo

Esta es la parte más importante de iptables, ya que cuando tengamos un paquete que coincida con nuestras reglas, le deberemos decir que tiene que hacer con él. Sobra decir que no es lo mismo un `-j ACCEPT` que un `-j DROP`. Nos centraremos en los objetivos más comunes.

LOG. Lo utilizaremos cuando queramos que quede constancia en el syslogd (puede ser otro) del paso de algún paquete. Básicamente se puede utilizar para el *debuging* del firewall, o simplemente si queremos tener constancia de un conato de ataque.

> **Aviso**
>
> Para el primer caso, bastará tener una única regla con el objetivo LOG, pero para el segundo, deberemos tener dos reglas idénticas, una con el objetivo LOG y la siguiente con el objetivo DROP.

```
snowball:~# iptables -A INPUT -p icmp -j LOG --log-prefix "PAQUETES ICMP FILTRADOS "
--log-level debug --log-ip-option
```

Aquí definimos una regla para los paquetes ICMP (`-p icmp`) de entrada (`-A INPUT`). Cuando el sistema reciba paquetes ICMP, entonces iptables logeará (`-j LOG`) los paquetes. Con `--log-prefix` conseguiremos que cuando logee los paquetes añana el prefijo indicado en los logs. Esto es especialmente útil si tenemos un archivo grande de logs, en donde luego con `grep` podremos clasificar fácilmente los logs que nos interesen. Con `--log-level` le indicamos el rigor con el que queremos que guarde el log (man `syslog.conf` para más opciones). Por último, le indicamos que queremos que guarde todas las opciones de la cabecera IP con `--log-ip-option`.

Podemos comprobar que la regla funciona haciendo pings a la máquina en cuestión.

```
snowball:~# ping -c 1 localhost
snowball:~# tail /var/log/syslog

Mar 21 11:58:38 snowball kernel: PAQUETES ICMP FILTRADOS IN=lo
OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=IPv6/IPv4 ID=0 DF
PROTO=ICMP TYPE=8 CODE=0 ID=2082 SEQ=1

Mar 21 11:58:38 snowball kernel: PAQUETES ICMP FILTRADOS IN=lo
OUT= MAC=00:00:00:00:00:00:00:00:00:00:00:00:08:00 SRC=127.0.0.1
DST=127.0.0.1 LEN=84 TOS=0x00 PREC=0x00 TTL=IPv6/IPv4 ID=28729
PROTO=ICMP TYPE=0 CODE=0 ID=2082 SEQ=1
```

En caso de querer que los logs aparezcan en `/var/log/syslog` , deberíamos modificar `/etc/syslog.conf`.

MARK. Lo utilizaremos para el marcaje de paquetes. Este será el objetivo que se urilizará junto con `tc` para definir las reglas de QoS. Destacar que cuando marcamos un paquete no estamos cambiando ningún valor de la cabecera IP, sino el valor del paquete de cara al kernel. El valor que tomará MARK será un entero entre 0 y 65535. Para modificar la cabecera IP, podemos cambiar el valor del Type Of Service con el objetivo TOS.

```
snowball:~# iptables -t mangle -A PREROUTING -p tcp --dport 143 -j MARK \
--set-mark 2
```

Le indicamos a la tabla mangle que todo paquete IMAP (`-p tcp --dport 143`) que reciba la máquina deberá ser marcado (`-j MARK --set-mark 2`) para su posterior clasificación y envio (`-A PREROUTING`).

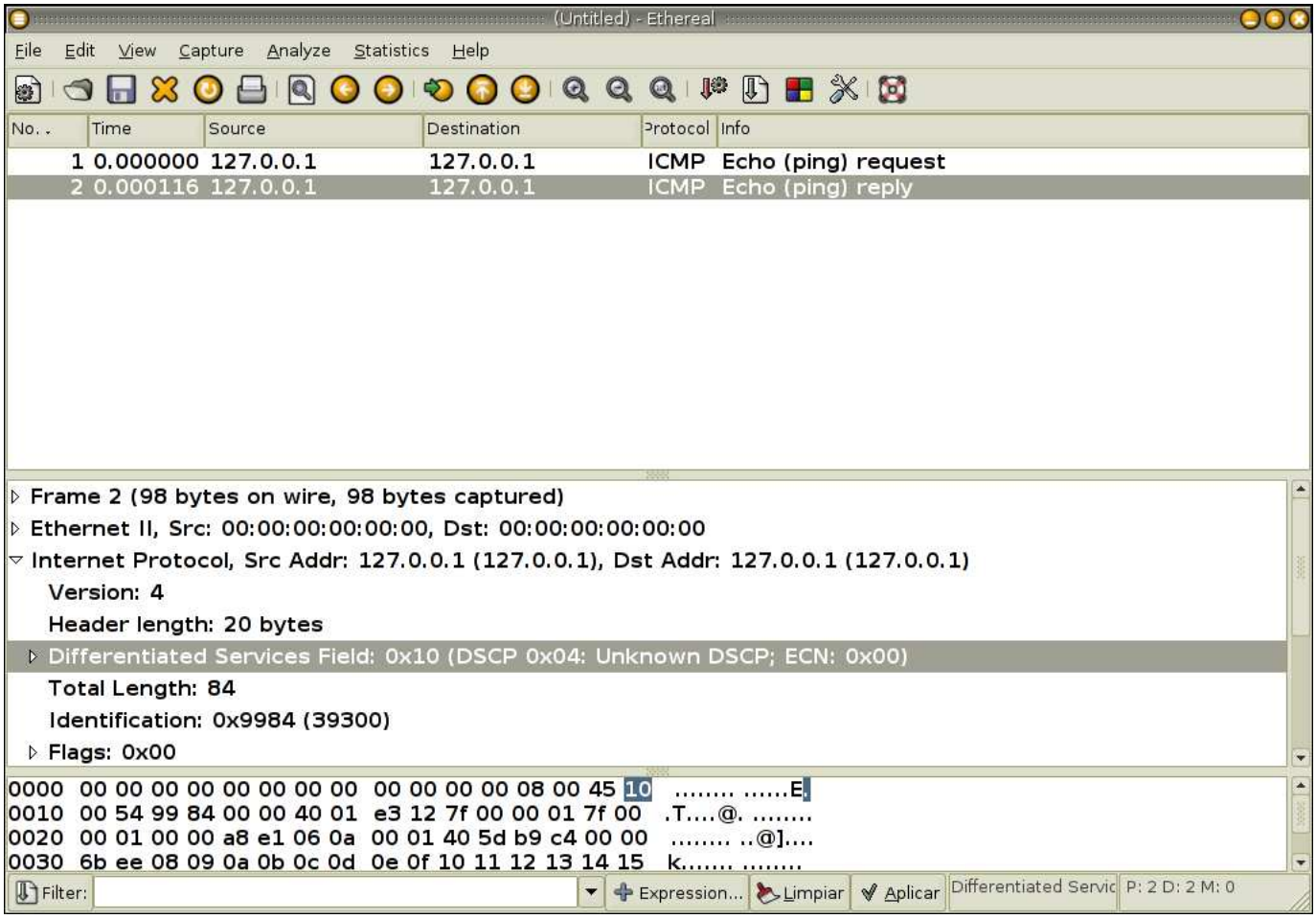
TOS. Lo utilizaremos para cambiar el valor de 8 bits del campo TOS dentro de la cabecera IPv4. Este campo indica la prioridad del tráfico, y se aplicará en routers que tengan soporte para QoS. Los valores son:

- Minimize-Delay - 0x10
- Maximize-Throughput - 0x08
- Maximize-Reliability - 0x04
- Minimize-Cost 2 - 0x02
- Normal-Service 0 - 0x00

Un ejemplo sería:

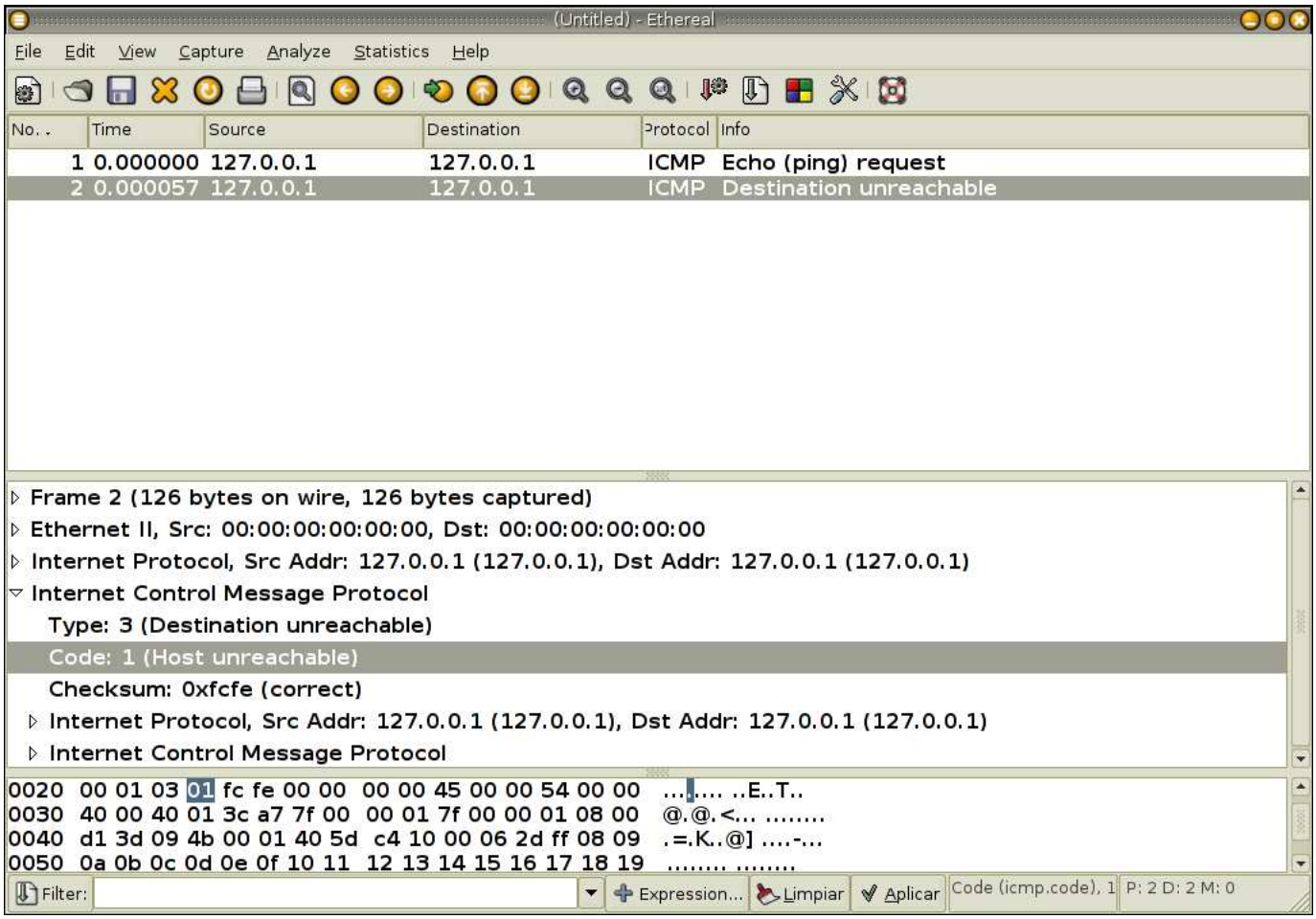
```
snowball:~# iptables -t mangle -A PREROUTING -p icmp -j TOS --set-tos Minimize-Delay
```

Marcamos todos los paquetes ICMP con el TOS `Minimize-Delay`. También los podríamos haber marcado con 0x10. Podemos comprobar que ha cambiado el valor del TOS haciendo un ping y capturando el paquete con un sniffer.



REJECT. Es igual que DROP, per más "educado". Cuando descarta el paquete puede devolver un error ICMP como *icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited o icmp-host-prohibited*.

```
snowball:~# iptables -A INPUT -p icmp -j REJECT --reject-with icmp-host-unreachable
snowball:~# ping -c 1 localhost

PING localhost (127.0.0.1) 56(84) bytes of data.

--- localhost ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```



DNAT. Objetivo válido para la tabla nat y las cadenas PREROUTING y OUTPUT. Con DNAT, podemos cambiar la dirección de destino del paquete. Esto es especialmente útil cuando queremos redirigir las peticiones a un servidor interno de la LAN con direcciones IP privadas según especifica el RFC 1918.

```
snowball:~# iptables -t nat -A PREROUTING -p tcp -d 81.65.123.90 --dport 80 \
-j DNAT --to-destination 192.168.1.10:80
```

Cuando recibimos (`-A PREROUTING`) una petición web (`-p tcp --dport 80`) a la dirección IP pública (`-d 81.65.123.90`), la enviamos a la dirección IP privada de la LAN (`--to-destination 192.168.1.10:80`).

REDIRECT. Este objetivo sólo es válido en las cadenas PREROUTING y OUTPUT de la tabla nat. Es una especialización de DNAT. En DNAT podemos elegir la IP de destino, en REDIRECT siempre será la del propio host.

```
snowball:~# iptables iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT \
--to-ports 8080
```

Cuando recibimos (`-A PREROUTING`) una petición web (`-p tcp --dport 80`), la redirigimos al puerto del *proxy* (`-j REDIRECT --to-ports 8080`).

SNAT. Objetivo valido también en la tabla nat, pero únicamente en la cadena POSTROUTING. Este objetivo es muy utilizado sobretodo para todos aquellos que tiene una única conexión a Internet y varias máquinas detrás.

```
snowball:~# iptables iptables -t nat -A POSTROUTING -o eth0 -j SNAT \
--to-source 81.65.123.90
```

Cuando un paquete de la LAN es enrutado (`-A POSTROUTING`) hacia la ethernet que tiene salida a Internet (`-o eth0`), entonces cambia la IP original, por la que nos suministra nuestro ISP (`-j SNAT --to-source 81.65.123.90`) para poder tener salida a Internet.

MASQUERADE. Es una particularización de SNAT, y por tanto únicamente utilizable en la cadena POSTROUTING de la tabla nat. Se utiliza en caso de que la dirección IP pública sea asignada dinámicamente (que suele ser lo más normal), con lo que nos ahorramos la opción `--to-source`.

```
snowball:~# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Este es el ejemplo más utilizado a nivel mundial ;-) .Realizará la misma tarea que el SNAT, pero no hará falta que le indiquemos IP.

> **Nota:** En caso de tener una conexión a Internet con una direccióon IP fija asignada por nuestro ISP, utilizaremos SNAT.

3.7. Manejo de las reglas

Las operaciones básicas que se pueden hacer a una cadena son:

- **-N** Crea una nueva cadena
- **-X** Borra una cadena definida por el usuario
- **-A** Añade una regla nueva al final de la cadena
- **-D** Elimina reglas de una cadena
- **-I** Inserta una regla en la posición de la cadena que le indiquemos

- **-R** Mueve la regla dentro de la cadena
- **-F** Elimina TODAS las reglas de la cadena
- **-L** Lista todas las reglas de la cadena
- **-Z** Pone a cero los contadores de paquetes de cada regla
- **-P** Cambia la política de cada cadena

Capítulo 4. Configuración de Firewalls

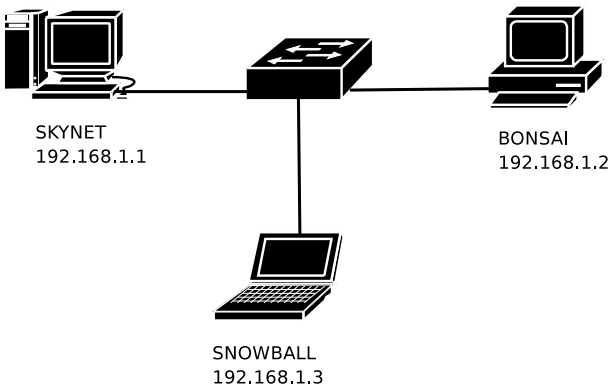
En este capítulo veremos ejemplos de reglas de iptables aplicados a arquitecturas diferentes, con necesidades diferentes, que irán aumentando en complejidad. Por convenio, se utilizarán los siguentes datos en todas las configuraciones:

1. Políticas por defecto **DROP** (salvo el la cadena OUTPUT del primer ejemplo).
2. Red **10.0.0.0/8** para las LAN
3. Red **172.16.0.0/16** para las DMZ
4. Red**192.168.0.0/30** para enlaces punto a punto (evitará el *spoofing* si no se considera el *subnet zero)*

4.1. Configuraciones básicas

Primero comenzaremos con un ejemplo muy simple en la que vamos a securizar mínimamente el host 192.168.1.1 que está conectado a otros 2 host.

4.1.1. Ejemplo 1



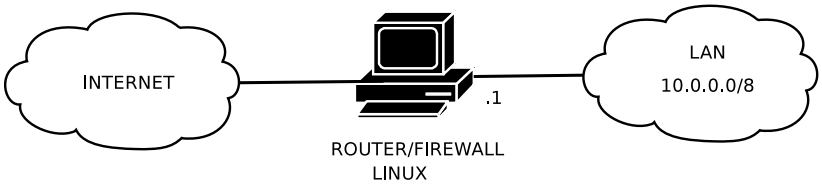

```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
BONSAI="192.168.1.2"
SNOWBALL="192.168.1.3"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

$IPT -F
$IPT -X
$IPT -Z
```

```
#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT ACCEPT

#EMPEZAMOS CON LAS REGLAS DE FILTRADO
-------------------------------------------------------------------------

#PERMITIREMOS QUE SKYNET PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT

#SÓLO SNOWBALL PODRÁ ACCEDER A SKYNET PARA ADMINISTRARLO

$IPT -A INPUT -s $SNOWBALL -j ACCEPT

#PERMITIMOS A BONSAI UTILIZAR EL SERVIDOR WEB Y EL FTP

$IPT -A INPUT -s $BONSAI -m multiport --destination-ports 80,443 -j ACCEPT

$IPT -A INPUT -s $BONSAI -p tcp --dport 20:21 -j ACCEPT

#PARA TODO LO DEMÁS -> DROP
```

> **Aviso**
>
> Hay que tener en cuenta que el ORDEN de las reglas es vital. Una regla en un mal lugar, puede echarnos al suelo toda la seguridad del firewall, o hacer que el firewall no haga lo que se supone que debe hacer.

4.1.2. Ejemplo 2

En este ejemplo, los usuarios de la LAN, tendrán salida a internet mediante NAT a través del router/firewall.



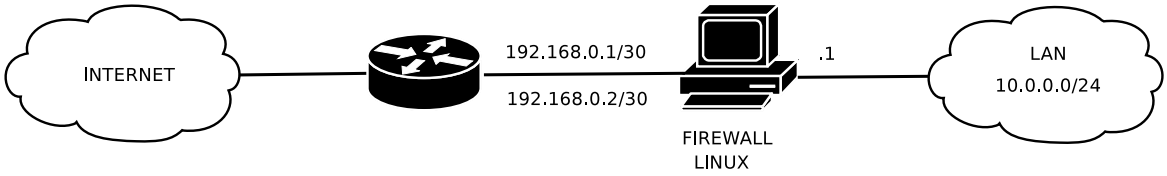

```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
LAN="10.0.0.0/8"
ADMIN="10.0.0.10"
IF_EXT="eth0"
```

```
UP_PORTS="1024:65535"
DNS_SERVER="195.235.113.3"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

for TABLE in filter nat
do
 $IPT -t $TABLE -F
 $IPT -t $TABLE -X
 $IPT -t $TABLE -Z
done

#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#ACTIVAMOS EL NAT Y EL REENVIO DE PAQUETES

$IPT -t nat -A POSTROUTING -s $LAN -o $IF_EXT -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward

#PERMITIREMOS QUE EL ROUTER/FIREWALL PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT
$IPT -A OUTPUT -o lo -j ACCEPT

#EMPEZAMOS CON LAS REGLAS DE FILTRADO
------------------------------------------------------------------------

#PERMITIMOS QUE ÚNICAMENTE EL ADMINISTRADOR PUEDA CONECTAR POR SSH

$IPT -A INPUT -s $ADMIN -p tcp --dport 22 -j ACCEPT
$IPT -A OUTPUT -d $ADMIN -p tcp --sport 22 -j ACCEPT

#PERMITIMOS LAS CONSULTAS DNS

$IPT -A FORWARD  -p udp -s $LAN  -d $DNS_SERVER --dport 53 -j ACCEPT
$IPT -A FORWARD  -p udp -d $LAN  -s $DNS_SERVER --sport 53 -j ACCEPT

#DAMOS SALIDA AL TRÁFICO WEB

$IPT -A FORWARD -s $LAN -p tcp -m multiport --destination-ports 80,443 -j ACCEPT
$IPT -A FORWARD -d $LAN -p tcp -m multiport --source-ports 80,443 -j ACCEPT

#PERMITIMOS EL INTERCAMBIO DE FICHEROS MEDIANTE FTP

$IPT -A FORWARD -p tcp -s $LAN --sport $UP_PORTS --dport 20:21 -j ACCEPT
$IPT -A FORWARD -p tcp -d $LAN --dport $UP_PORTS --sport 20:21 -j ACCEPT

$IPT -A FORWARD -p tcp -s $LAN --sport $UP_PORTS  -dport $UP_PORTS -j ACCEPT
$IPT -A FORWARD -p tcp -d $LAN --dport $UP_PORTS  -sport $UP_PORTS -j ACCEPT
```

4.2. Confi guraciones medias

4.2.1. Ejemplo 3

Ahora separamos los elementos del ejemplo anterior, interponiendo entre el router y la LAN en firewall. El NAT lo realizará el router.

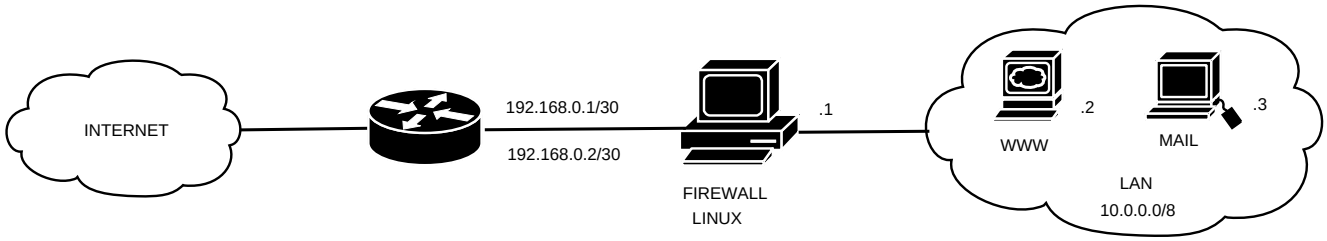



```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
LAN="10.0.0.0/8"
ADMIN="10.0.0.10"
IF_EXT="eth0"
IF_INT="eth1"
UP_PORTS="1024:65535"
DNS_SERVER="195.235.113.3"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

for TABLE in filter nat
do
$IPT -t $TABLE -F
 $IPT -t $TABLE -X
 $IPT -t $TABLE -Z
done

#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#ACTIVAMOS EL REENVIO DE PAQUETES

echo 1 > /proc/sys/net/ipv4/ip_forward

#ACTIVAMOS EL RETORNO DE LOS PAQUETES, CON LO QUE SÓLO ESPECIFICAREMOS UNA REGLA.

$IPT -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

```
#PERMITIREMOS QUE EL FIREWALL PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT

#LOGEAREMOS LOS ICMP QUE VENGAN DE LA LAN (SALVO DESDE EL HOST DEL ADMIN).

$IPT -N ICMP
$IPT -A ICMP -i $IF_INT -s $ADMIN -p icmp -j ACCEPT
$IPT -A ICMP -i $IF_INT -s ! $ADMIN -p icmp -j LOG \
--log-prefix "ICMP FILTRADOSDESDE LA LAN:   "


#EMPEZAMOS CON LAS REGLAS DE FILTRADO
------------------------------------------------------------------------

#PERMITIMOS QUE ÚNICAMENTE EL ADMINISTRADOR PUEDA CONECTAR POR SSH

$IPT -A INPUT -m state --state NEW -s $ADMIN -p tcp --dport 22 -j ACCEPT

#PERMITIMOS LAS CONSULTAS DNS

$IPT -A FORWARD -m state --state NEW -o $IF_EXT -p udp -s $LAN  -d $DNS_SERVER \
--dport 53 -j ACCEPT

#DAMOS SALIDA AL TRÁFICO WEB

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp -m multiport \
--destination-ports 80,443 -j ACCEPT

#PERMITIMOS EL INTERCAMBIO DE FICHEROS MEDIANTE FTP

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN --sport $UP_PORTS \
--dport 20:21 -j ACCEPT

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN --sport $UP_PORTS \
--dport $UP_PORTS -j ACCEPT

#DEJAMOS PASAR EL TRÁFICO DE CORREO

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp -m multiport \
--destination-ports 25,143,110 -j ACCEPT
```

4.2.2. Ejemplo 4

En esta arquitectura añadiremos dos servidores a la LAN que deberán ser accesible desde el exterior




```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
LAN="10.0.0.0/8"
ANY="0.0.0.0/0"
ADMIN="10.0.0.10"
IF_EXT="eth0"
IF_INT="eth1"
UP_PORTS="1024:65535"
WEB_SERVER="10.0.0.2"
MAIL_SERVER="10.0.0.3"
DNS_SERVER="195.235.113.3"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

for TABLE in filter nat
do
 $IPT -t $TABLE -F
 $IPT -t $TABLE -X
 $IPT -t $TABLE -Z
done


#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#ACTIVAMOS EL REENVIO DE PAQUETES

echo 1 > /proc/sys/net/ipv4/ip_forward

#ACTIVAMOS EL RETORNO DE LOS PAQUETES, CON LO QUE SÓLO ESPECIFICAREMOS UNA REGLA.

$IPT -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

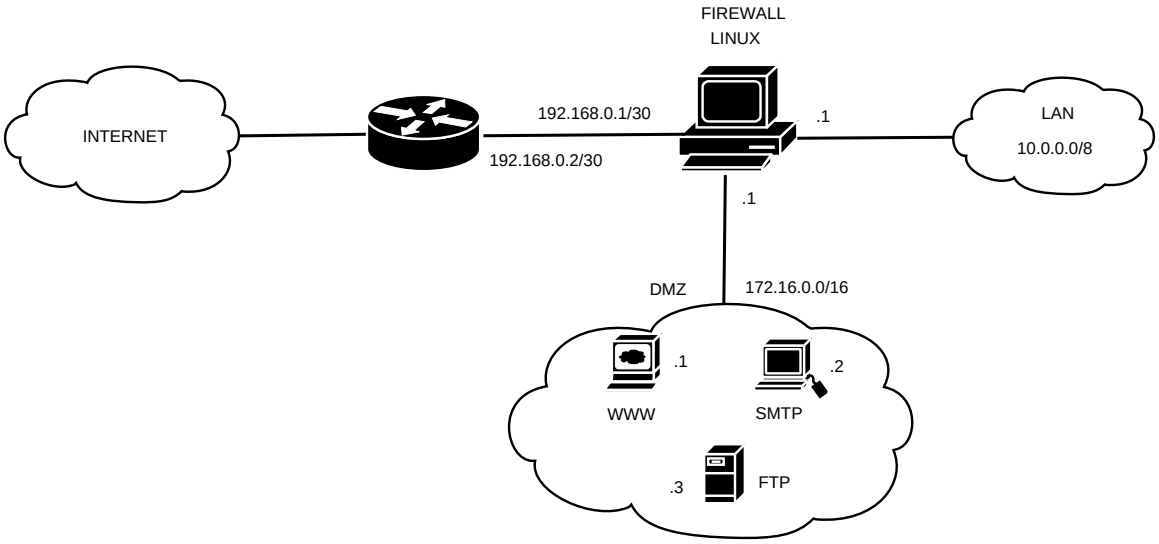
```
#PERMITIREMOS QUE EL FIREWALL PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT

#LOGEAREMOS LOS ICMP QUE VENGAN DE LA LAN  Y DE INTERNET
#(SALVO DESDE EL HOST DEL ADMIN)

$IPT -N ICMP
$IPT -A ICMP -s $ADMIN -p icmp -j ACCEPT
$IPT -A ICMP -i $IF_INT -s ! $ADMIN -p icmp -j LOG \
--log-prefix "ICMP FILTRADOS DESDE LA LAN:   "
$IPT -A ICMP -i $INT_EXT -s $ANY -p icmp -j LOG \
--log-prefix "ICMP FILTRADOS DESDE INTERNET:   "

#EMPEZAMOS CON LAS REGLAS DE FILTRADO
------------------------------------------------------------------------

#PERMITIMOS QUE ÚNICAMENTE EL ADMINISTRADOR PUEDA CONECTAR POR SSH

$IPT -A INPUT -m state --state NEW -s $ADMIN -p tcp --dport 22 -j ACCEPT

#PERMITIMOS LAS CONSULTAS DNS

$IPT -A FORWARD -m state --state NEW -o $IF_EXT -p udp -s $LAN  -d $DNS_SERVER \
--dport 53 -j ACCEPT

#DAMOS SALIDA AL TRÁFICO WEB

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp -m multiport \
--destination-ports 80,443 -j ACCEPT

#PERMITIMOS EL INTERCAMBIO DE FICHEROS MEDIANTE FTP

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN --sport $UP_PORTS \
--dport 20:21 -j ACCEPT

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN --sport $UP_PORTS  \
-dport $UP_PORTS -j ACCEPT

#DEJAMOS PASAR EL TRÁFICO DE CORREO

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp -m multiport \
--destination-ports 25,143,110 -j ACCEPT

#HABILITAMOS EL IRC

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp --dport 6667 -j ACCEPT

#REENVIAMOS LAS PETICIONES WEB, LAS POP3 Y LAS IMAP A LOS SERVIDORES DE LA LAN

$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_INT -s $ANY -p tcp \
--dport 80 -j DNAT --to-destination $WEB_SERVER:80
$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_INT -s $ANY -d $IP_FW_EX -p tcp \
--dport 443 -j DNAT --to-destination $WEB_SERVER:443
$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_INT -s $ANY -d $IP_FW_EX -p tcp \
--dport 110 -j DNAT --to-destination $MAIL_SERVER:110
```

```
$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_INT -s $ANY -d $IP_FW_EX -p tcp \
--dport 143 -j DNAT --to-destination $MAIL_SERVER:143
```

4.3. Confi guraciones avanzadas

4.3.1. Ejemplo 5

Aquí ya introducimos una DMZ en donde irán alojados todos los servicios accesibles desde el exterior, y dejando la LAN aparte.



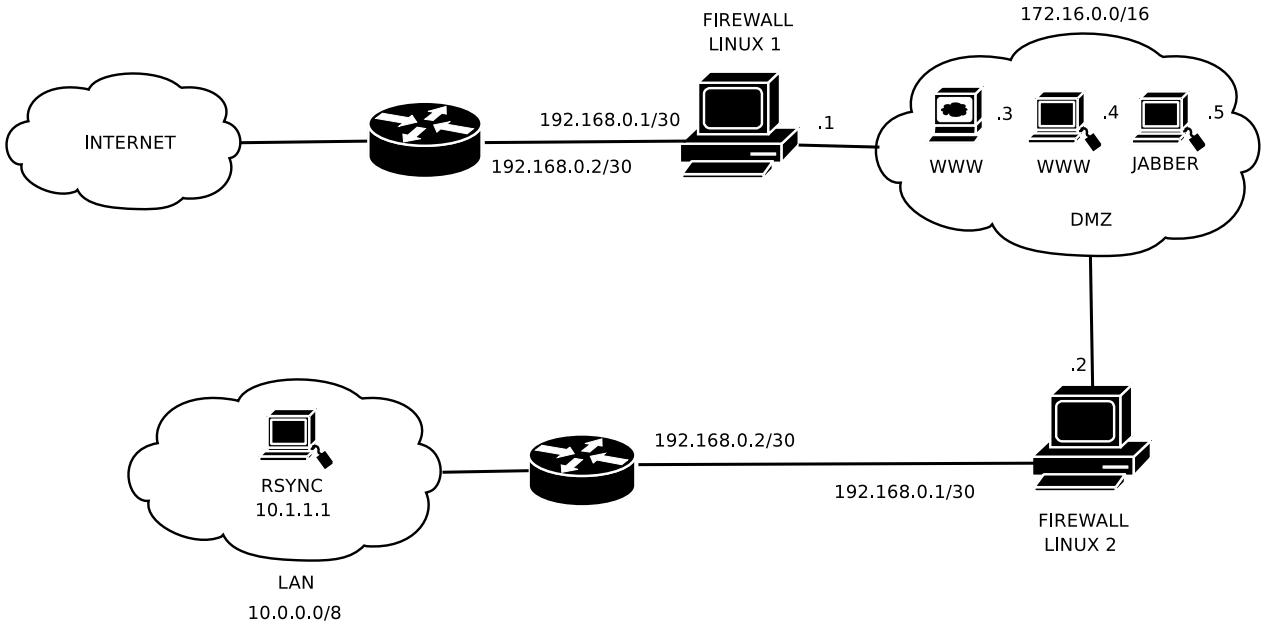

```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
LAN="10.0.0.0/8"
ANY="0.0.0.0/0"
DMZ="172.16.0.0/16"
ADMIN="10.0.0.10"
IF_EXT="eth0"
IF_INT="eth1"
IF_DMZ="eth2"
UP_PORTS="1024:65535"
WEB_SERVER="172.16.0.1"
MAIL_SERVER="172.16.0.2"
FTP_SERVER="172.16.0.3"
DNS_SERVER="195.235.113.3"
DOS="-m limit --limit-burst 10 --limit 5/second"
SYN="-m limit --limit-burst 20 --limit 10/second"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
```

```
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

for TABLE in filter nat
do
 $IPT -t $TABLE -F
 $IPT -t $TABLE -X
 $IPT -t $TABLE -Z
done

#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#ACTIVAMOS EL REENVIO DE PAQUETES

echo 1 > /proc/sys/net/ipv4/ip_forward

#ACTIVAMOS EL RETORNO DE LOS PAQUETES, CON LO QUE SÓLO ESPECIFICAREMOS UNA REGLA Y NO DOS.

$IPT -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#PERMITIREMOS QUE EL FIREWALL PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT

#LOGEAREMOS POSIBLES ATAQUES DoS Y SYN FLOODING

$IPT -N DOS_FLOOD
$IPT -N DOS_FOOD_DROP
$IPT -A DOS_FLOOD -s $ANY -p icmp $DOS -j RETURN
$IPT -A DOS_FOOD_DROP -j LOG --log-level debug  \
--log-prefix "EXCESO DE ICMP   "

$IPT -N SYN_FLOOD
$IPT -N SYN_FLOOD_DROP
$IPT -A SYN_FLOOD -s $ANY -p tcp --syn $SYN -j RETURN
$IPT -A SYN_FLOOD_DROP -j LOG --log-level debug \
--log-prefix "EXCESO DE CONEXIONES SYN   "

#EMPEZAMOS CON LAS REGLAS DE FILTRADO
-------------------------------------------------------------------------

#PERMITIMOS QUE ÚNICAMENTE EL ADMINISTRADOR PUEDA CONECTAR POR SSH

$IPT -A INPUT -m state --state NEW -s $ADMIN -i $IF_INT -p tcp --dport 22 -j ACCEPT

#PERMITIMOS LAS CONSULTAS DNS

$IPT -A FORWARD -m state --state NEW -o $IF_EXT -p udp -s $LAN  -d $DNS_SERVER \
--dport 53 -j ACCEPT
```

```
#DAMOS SALIDA AL TRÁFICO WEB

$IPT -A FORWARD -m state --state NEW -s $LAN -o $IF_EXT -p tcp -m multiport \
--destination-ports 80,443 -j ACCEPT

#PERMITIMOS EL INTERCAMBIO DE FICHEROS MEDIANTE FTP

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN -o $IF_EXT --sport $UP_PORTS \
--dport 20:21 -j ACCEPT

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN -o $IF_EXT --sport $UP_PORTS  \
--dport $UP_PORTS -j ACCEPT

#DEJAMOS PASAR EL TRÁFICO DE CORREO

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp -o $IF_EXT -m multiport \
--destination-ports 25,143,110 -j ACCEPT

#REENVIAMOS LAS PETICIONES WEB, Y LAS SMTP VENIDAS DE INTERNET A LA DMZ,
#AL FTP SÓLO SE PODRÁ ACCEDER DESDE LA LAN

$IPT -t nat -A PREROUTING -i $IF_INT -o $IF_DMZ -s $LAN -d $DMZ -p tcp \
--dport 20 -j DNAT --to-destination $FTP_SERVER:20
$IPT -t nat -A PREROUTING -i $IF_INT -o $IF_DMZ -s $LAN -d $DMZ -p tcp \
--dport 21 -j DNAT --to-destination $FTP_SERVER:21

$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_DMZ -s ! $LAN -d $IP_FW_EXT -p tcp \
--dport 443 -j DNAT --to-destination $WEB_SERVER:443
$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_DMZ -s ! $LAN -d $IP_FW_EXT -p tcp \
--dport 80 -j DNAT --to-destination $WEB_SERVER:80
$IPT -t nat -A PREROUTING -i $IF_EXT -o $IF_DMZ -s ! $LAN -d $IP_FW_EX -p tcp \
--dport 25 -j DNAT --to-destination $MAIL_SERVER:25
```

4.3.2. Ejemplo 6

Este será el ejemplo más complejo. Introduciremos la DMZ entre el exterior y la LAN, cada una de las cuales estará securizada por un firewall.



Este será el código para el firewall 1.

```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
LAN="10.0.0.0/8"
ANY="0.0.0.0/0"
DMZ="172.16.0.0/16"
IF_EXT="eth0"
IF_DMZ="eth1"
WEB_SERVER="172.16.0.3-172.16.0.4"
DNS_SERVER="195.235.113.3"
DOS="-m limit --limit-burst 10 --limit 5/second"
SYN="-m limit --limit-burst 20 --limit 10/second"
LOG_LIMIT="-m limit --limit-burst 20 --limit 5/second"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

for TABLE in filter nat
do
 $IPT -t $TABLE -F
 $IPT -t $TABLE -X
 $IPT -t $TABLE -Z
done
```

```
#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#ACTIVAMOS EL REENVIO DE PAQUETES

echo 1 > /proc/sys/net/ipv4/ip_forward

#ACTIVAMOS EL RETORNO DE LOS PAQUETES, CON LO QUE SÓLO ESPECIFICAREMOS UNA REGLA Y NO DOS.

$IPT -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#PERMITIREMOS QUE EL FIREWALL PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT

#LOGEAREMOS POSIBLES ATAQUES DoS Y SYN FLOODING

$IPT -N DOS_FLOOD
$IPT -N DOS_FOOD_DROP
$IPT -A DOS_FLOOD -s $ANY -p icmp $DOS -j RETURN
$IPT -A DOS_FOOD_DROP -j LOG $LOG_LIMIT --log-level debug  \
--log-prefix "EXCESO DE ICMP   " --log-ip-option

$IPT -N SYN_FLOOD
$IPT -N SYN_FLOOD_DROP
$IPT -A SYN_FLOOD -s $ANY -p tcp --syn $SYN -j RETURN
$IPT -A SYN_FLOOD_DROP -j LOG $LOG_LIMIT --log-level debug \
--log-prefix "EXCESO DE CONEXIONES SYN   " --log-ip-option

#EMPEZAMOS CON LAS REGLAS DE FILTRADO
------------------------------------------------------------------------

#DEJAMOS PASAR EL TRÁFICO PERMITIDO EN EL FW2 PARA SU SALIDA A INTERNET

$IPT-A FORWARD -m state --state NEW  $IF_DMZ -o $IF_EXT \
-s $LAN -d ! $DMZ -j ACCEPT

#EQUILIBRAMOS LA CARGA ENTRE LOS SERVIDORES WEB

$IPT -t nat -A PREROUTING -i $IF_EXT  -p tcp --dport 80  \
-o IF_DMZ -j DNAT --to-destination $WEB_SERVER
```

Este será el código para el firewall 2.

```
#!/bin/bash

#DEFINIMOS LAS VARIABLES CON LAS QUE TRABAJAREMOS

IPT=/sbin/iptables
LAN="10.0.0.0/8"
```

```
ANY="0.0.0.0/0"
ADMIN="10.0.0.10"
RSYNC_SERVER="10.1.1.1"
WEB_SERVER="172.16.0.3-172.16.0.4"
JABBER_SERVER="172.16.0.5"
IF_EXT="eth0"
IF_DMZ="eth1"
UP_PORTS="1024:65535"
DOS="-m limit --limit-burst 10 --limit 5/second"
SYN="-m limit --limit-burst 20 --limit 10/second"
LOG_LIMIT="-m limit --limit-burst 20 --limit 5/second"

#BORRAMOS CUALQUIER REGLA ANTERIOR,LAS CADENAS DE USUARIO EXISTENTES,
#Y PONEMOS A CERO LOS CONTADORES DE LAS CADENAS

for TABLE in filter nat
do
 $IPT -t $TABLE -F
 $IPT -t $TABLE -X
 $IPT -t $TABLE -Z
done


#DEFINIMOS LAS POLÍTICAS POR DEFECTO

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP

#ACTIVAMOS EL REENVIO DE PAQUETES

echo 1 > /proc/sys/net/ipv4/ip_forward

#ACTIVAMOS EL RETORNO DE LOS PAQUETES, CON LO QUE SÓLO ESPECIFICAREMOS UNA REGLA Y NO DOS.

$IPT -I INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPT -I FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT

#PERMITIREMOS QUE EL FIREWALL PUEDA TRABAJAR LOCALMENTE

$IPT -A INPUT -i lo -j ACCEPT

#EMPEZAMOS CON LAS REGLAS DE FILTRADO
------------------------------------------------------------------------

#PERMITIREMOS QUE EL ADMIN ACTUALICE LOS SERVIDORES DE LA DMZ MEDIANTE
#UN SERVIDOR RSYNC UBICADO EN LA LAN

$IPT -A FORWARD -i $IF_INT -s $RSYNC_SERVER -o $IF_EXT -d $DMZ \
-p tcp -sport 873 -j ACCEPT

#LOS USUARIOS DE LA LAN TENDRÁN ACCESO A LOS SERVIDORES DE LA DMZ

$IPT -A FORWARD -i $IF_EXT -s $LAN -o $IF_INT -d $DMZ -p tcp -m multiport \
--destination-ports 80,443 -j ACCEPT
```

```
#HABILITAMOS UN SERVIDOR JABBER PARA LA COMUNICACIÓN INTERNA
#DE LOS USUARIOS DE LA LAN

$IPT -A FORWARD -i $IF_EXT -s $LAN -o $IF_INT -d $JABBER_SERVER -p tcp \
--dport 5222:5223 -j ACCEPT

#PERMITIMOS LAS CONSULTAS DNS

$IPT -A FORWARD -m state --state NEW -o $IF_EXT -p udp -s $LAN  -d $DNS_SERVER \
--dport 53 -j ACCEPT

#DAMOS SALIDA AL TRÁFICO WEB

$IPT -A FORWARD -m state --state NEW -s $LAN -o $IF_EXT -p tcp -m multiport \
--destination-ports 80,443 -j ACCEPT

#PERMITIMOS EL INTERCAMBIO DE FICHEROS MEDIANTE FTP

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN -o $IF_EXT --sport $UP_PORTS \
--dport 20:21 -j ACCEPT

$IPT -A FORWARD -m state --state NEW -p tcp -s $LAN -o $IF_EXT --sport $UP_PORTS  \
--dport $UP_PORTS -j ACCEPT

#DEJAMOS PASAR EL TRÁFICO DE CORREO

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp -o $IF_EXT -m multiport \
--destination-ports 25,143,110 -j ACCEPT

#HABILITAMOS EL IRC

$IPT -A FORWARD -m state --state NEW -s $LAN -p tcp --dport 6667 -j ACCEPT
```

Capítulo 5. QoS

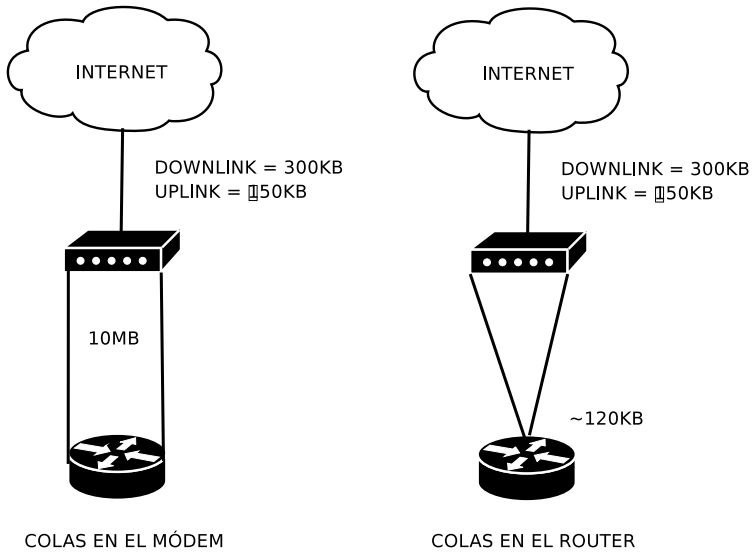
5.1. Introducción

El actual kernel de Linux, viene con unas serie de características que nos peritirá realizar un control avanzado del tráfico aplicando QoS (Calidad de Servicio). Podremos encontrar dos protocolos que realizan dicha tarea: *Diffserv* (RFC 2475) y *RSVP*, aunque nosotros nos basaremos en otras características de adaptación de tráfico del kernel. Se entiende por QoS el poder tratar a un paquete u otro dependiendo de sus características, con lo que conseguiremos una mejor gestión del ancho de banda. Para mí, el QoS es la alternativa a desconectar a mis compañeros de piso (grandes consumidores de redes P2P) del switch, para poder leer el correo o navegar un poco.

Para una buena gestión del ancho de banda, tan sólo necesitaremos el paquete `iproute2`, `iptables` y mucho café; pero antes deberemos comprender ciertos conceptos.

Cuando se envian datos por una interfaz, por defecto se encolan en una FIFO (el primero que entra es el primero que sale), y cuando tenemos una gran cantidad de tráfico, si sumamos el tiempo que tarda el paquete en entrar y salir de la FIFO y repetir el mismo proceso en nuestro ISP, tenemos como resultado una latencia bastante elevada y nada recomendable.

Para solucionarlo, lo que deberemos hacer es transladar esa cola (normalmente situada en el módem) donde se crean las congestiones, a nuestro router para darle forma a nuestro gusto. Pero, ¿y cómo consigo esto?, pues creando un **pequeño** cuello de botella entre nuestro router y el módem.



Con esto conseguimos un mejor rendimiento, puesto que ahora podemos configurar en el router ciertos tipos de disciplinas de cola para manejar el tráfico de una manera más eficiente. Las disciplinas de cola se encargan de decirle al kernel cómo debe elegir el siguiente paquete para transmitir (cada disciplina de cola se basa en unas características para realizar dicha labor).

5.1.1. Disciplinas de colas simple

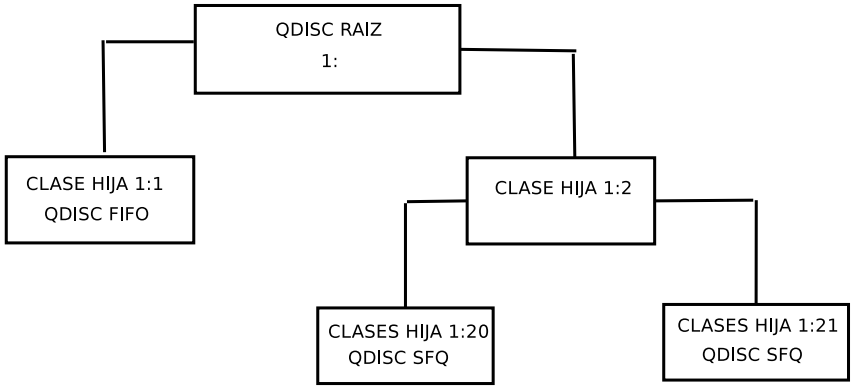
Simplemente se limitan a aceptar, reordenar, descartar o retrasar los datos. Podemos encontrar:

- **pfifo_fast**: First In Fisrst Out, con especial tratamiento al campo TOS.
- **TBF**: Deja pasar paquetes que no se excedan de una tasa impuesta por nosotros.
- **RED**: Únicamente indicado para routers de *backbone*, en donde el envio de tráfico debe ser lo más rápido posible.
- **SFQ**: Envia datos de forma equitativa, evitando que nadie se apodere del ancho de banda.

> **Nota:** Para una información más detallada sobre sus características y usos, se puede consultar Linux Advanced Routing & Traffic Control (http://lartc.org)

5.1.2. Disciplinas de colas con clase

Estas a diferencia de las simples, tienen nivel de profundidad. Es decir, se puede tener una qdisc raíz de la que cuelguen ciertas subdivisiones, en las cuales se podrá agrupar diferentes tipos de paquetes (clases) para luego darle prioridades. A su vez, las clases pueden tener otras qdisc con o sin clases. Por defecto, cuando se crea una clase, se le adjunta una qdisc FIFO (se puede cambiar por otra), salvo que de esa clase tenga clases hijas.



Para identificar las clases, utilizaremos una notación de dos números separados por dos puntos : *major*:*minor*. El major, se utiliza para referenciar la raíz de una clase, en cambio el minor se refiere a cualquiera de las clases que descienden de la raíz padre.

Por último, para tratar a los paquetes en las diferentes clases, deberemos de utilizar un filtro, que de la misma manera que hacían las reglas del firewall, en caso de cumplir unas determinadas características, irán a unas clases o a otras. Ahora ya podemos ver las principales qdisc con clases.

- **CBQ**: Algoritmo que basándose en prioridades, envia paquetes. Útil para enviar paquetes a clases hijas.
- **HTB**: Igual que CBQ pero un poco más eficiente.

5.1.3. Filtros

Los filtros serán los que nos permitirán clasificar los paquetes en diferentes clases.

- **fw**: Clasificará el trafico basándose en la marca del paquete que le colocaremos con el objetivo `MARK` de iptables.
- **u32**: Se basará en características de la cabecera IP para hacer las clasificaciones.

5.2. Reserva de ancho de banda según protocolo

```
#!/bin/bash

echo "Starting QoS"

# 10 -> ssh,syn,DNS,ipv6 (20kbits) --> PRIO 1
# 11 -> smtp,pop3,imap,news (10kbits) --> PRIO 2
# 12 -> http,https,ftp (40kbits) --> PRIO 3
# 13 -> emule... (50kbits) --> PRIO 4

MAX=120
IPT=/sbin/iptables
IP6T=/sbin/ip6tables
TC=/sbin/tc
IF_EXT="eth0"

$TC qdisc add dev $IF_EXT root handle 1: htb default 13

$TC class add dev $IF_EXT parent 1: classid 1:1 htb rate ${MAX}kbit ceil ${MAX}kbit

$TC class add dev $IF_EXT parent 1:1 classid 1:10 htb rate 20kbit ceil ${MAX}kbit prio 1
$TC class add dev $IF_EXT parent 1:1 classid 1:11 htb rate 10kbit ceil ${MAX}kbit prio 2
$TC class add dev $IF_EXT parent 1:1 classid 1:12 htb rate 40kbit ceil ${MAX}kbit prio 3
$TC class add dev $IF_EXT parent 1:1 classid 1:13 htb rate 50kbit ceil ${MAX}kbit prio 4

$TC qdisc add dev $IF_EXT parent 1:10 handle 100: sfq perturb 10 quantum 1500
$TC qdisc add dev $IF_EXT parent 1:11 handle 110: sfq perturb 10 quantum 1500
$TC qdisc add dev $IF_EXT parent 1:12 handle 120: sfq perturb 10 quantum 1500
$TC qdisc add dev $IF_EXT parent 1:13 handle 130: sfq perturb 10 quantum 1500

$TC filter add dev $IF_EXT parent 1:0 protocol ip prio 1 handle 1 fw classid 1:10
$TC filter add dev $IF_EXT parent 1:0 protocol ip prio 2 handle 2 fw classid 1:11
$TC filter add dev $IF_EXT parent 1:0 protocol ip prio 3 handle 3 fw classid 1:12
$TC filter add dev $IF_EXT parent 1:0 protocol ip prio 4 handle 4 fw classid 1:13

#Empezamos con el marcaje de paquetes!!!

# ICMP

$IPT -t mangle -A PREROUTING -p icmp -j MARK --set-mark 1
$IPT -t mangle -A PREROUTING -p icmp -j RETURN

$IPT -t mangle -A OUTPUT -p icmp -j MARK --set-mark 1
$IPT -t mangle -A OUTPUT -p icmp -j RETURN
```

```
# SSH

$IPT -t mangle -A PREROUTING -p tcp  --dport 22 -j MARK --set-mark 1
$IPT -t mangle -A PREROUTING -p tcp  --dport 22 -j RETURN

$IPT -t mangle -A OUTPUT -p tcp  --dport 22 -j MARK --set-mark 1
$IPT -t mangle -A OUTPUT -p tcp  --dport 22 -j RETURN

#HTTP & HTTPS

$IPT -t mangle -A PREROUTING -p tcp  --dport 80 -j MARK --set-mark 3
$IPT -t mangle -A PREROUTING -p tcp  --dport 80 -j RETURN

$IPT -t mangle -A PREROUTING -p tcp  --dport 443 -j MARK --set-mark 3
$IPT -t mangle -A PREROUTING -p tcp  --dport 443 -j RETURN

#NEWS (NNTP)

$IPT -t mangle -A PREROUTING -p tcp  --dport 119 -j MARK --set-mark 2
$IPT -t mangle -A PREROUTING -p tcp  --dport 119 -j RETURN

#FTP

$IPT -t mangle -A PREROUTING -p tcp  --dport 20:21 -j MARK --set-mark 3
$IPT -t mangle -A PREROUTING -p tcp  --dport 20:21 -j RETURN

$IPT -t mangle -A OUTPUT -p tcp  --dport 20:21 -j MARK --set-mark 3
$IPT -t mangle -A OUTPUT -p tcp  --dport 20:21 -j RETURN

#POP3

$IPT -t mangle -A PREROUTING -p tcp  --dport 110 -j MARK --set-mark 2
$IPT -t mangle -A PREROUTING -p tcp  --dport 110 -j RETURN

#IMAP

$IPT -t mangle -A PREROUTING -p tcp  --dport 143 -j MARK --set-mark 2
$IPT -t mangle -A PREROUTING -p tcp  --dport 143 -j RETURN

#SMTP

$IPT -t mangle -A PREROUTING -p tcp  --dport 25 -j MARK --set-mark 2
$IPT -t mangle -A PREROUTING -p tcp  --dport 25 -j RETURN

#DNS

$IPT -t mangle -A PREROUTING -p udp  --dport 53 -j MARK --set-mark 1
$IPT -t mangle -A PREROUTING -p udp  --dport 53 -j RETURN

$IPT -t mangle -A OUTPUT -p udp  --dport 53 -j MARK --set-mark 1
$IPT -t mangle -A OUTPUT -p udp  --dport 53 -j RETURN

# TCP con el bit SYN activado (principio de conexión)

$IPT -t mangle -I PREROUTING -p tcp  --syn -j MARK --set-mark 1
$IPT -t mangle -I PREROUTING -p tcp  --syn -j RETURN
```

```
#Tráfico en general (emule....)

$IPT -t mangle -A PREROUTING -j MARK --set-mark 4

#Tráfico IPv6

$IP6T -t mangle -A PREROUTING -j MARK --set-mark 1
$IP6T -t mangle -A OUTPUT -j MARK --set-mark 1
```

> **Nota:** Si se desea, se puede afinar la clasificación del tráfico P2P, con el módulo IPP2P de iptables, disponible en esta web. (http://rnvs.informatik.uni-leipzig.de/ipp2p/index_en.html)

Ahora podemos comprobar que las colas se han creado y que hay tráfico en ellas.

```
zen:~# tc -s class show dev eth0
```

```
class htb 1:11 parent 1:1 leaf 110: prio 1 rate 10Kbit ceil 120Kbit burst 1611b cburst 1752b
 Sent 22950 bytes 281 pkts (dropped 0, overlimits 0)
 lended: 281 borrowed: 0 giants: 0
 tokens: 921984 ctokens: 107143

class htb 1:1 root rate 120Kbit ceil 120Kbit burst 1752b cburst 1752b
 Sent 83502 bytes 849 pkts (dropped 0, overlimits 0)
 rate 16bps
 lended: 0 borrowed: 0 giants: 0
 tokens: 109934 ctokens: 109934

class htb 1:10 parent 1:1 leaf 100: prio 0 rate 20Kbit ceil 120Kbit burst 1624b cburst 1752b
 Sent 19969 bytes 222 pkts (dropped 0, overlimits 0)
 rate 11bps
 lended: 222 borrowed: 0 giants: 0
 tokens: 608399 ctokens: 109934

class htb 1:13 parent 1:1 leaf 130: prio 3 rate 50Kbit ceil 120Kbit burst 1663b cburst 1752b
 Sent 7623 bytes 85 pkts (dropped 0, overlimits 0)
 lended: 85 borrowed: 0 giants: 0
 tokens: 258558 ctokens: 113666

class htb 1:12 parent 1:1 leaf 120: prio 2 rate 40Kbit ceil 120Kbit burst 1650b cburst 1752b
 Sent 32960 bytes 261 pkts (dropped 0, overlimits 0)
 rate 4bps
 lended: 261 borrowed: 0 giants: 0
 tokens: 302423 ctokens: 108290
```

5.3. Reserva de ancho de banda por IP

Puede ser, que prefiramos distribuir el ancho de banda equitativamente entre nuestros usuarios, o asignar a cada usuario cierta parte. Para el primer caso, bastaría una simple qdisc SFQ, con lo que conseguiríamos que todos los usuarios pudiesen enviar paquetes de forma equitativa. Esta opción la conseguiríamos con una simple línea:

```
zen:~# tc qdisc add dev eth0 root handle 1: sfq perturb 10 quantum 1500
```

Tal vez sea una qdisc más apropiada que la FIFO por defecto, aunque seguramente nuestro usuario agradecerá el tener asignado una parte del ancho de banda. Un simple filtro U32 nos puede ayudar en esta tarea:

```
#!/bin/bash

MAX=120
IPT=/sbin/iptables
IP6T=/sbin/ip6tables
TC=/sbin/tc
IF_EXT="eth0"

$TC qdisc add dev $IF_EXT root handle 1: htb

$TC class add dev $IF_EXT parent 1: classid 1:1 htb rate ${MAX}kbit ceil ${MAX}kbit

$TC class add dev $IF_EXT parent 1:1 classid 1:10 htb rate 30kbit ceil ${MAX}kbit
$TC class add dev $IF_EXT parent 1:1 classid 1:11 htb rate 30kbit ceil ${MAX}kbit
$TC class add dev $IF_EXT parent 1:1 classid 1:12 htb rate 30kbit ceil ${MAX}kbit
$TC class add dev $IF_EXT parent 1:1 classid 1:13 htb rate 30kbit ceil ${MAX}kbit

$TC filter add dev $IF_EXT parent 1: protocol ip u32 match ip src 192.168.0.3  flowid 1:10
$TC filter add dev $IF_EXT parent 1: protocol ip u32 match ip src 192.168.0.4  flowid 1:11
$TC filter add dev $IF_EXT parent 1: protocol ip u32 match ip src 192.168.0.5  flowid 1:12
$TC filter add dev $IF_EXT parent 1: protocol ip u32 match ip src 192.168.0.6  flowid 1:13
```

Otra solución alternativa a U32, hubiera sido iptables + filtro fw:

```
#!/bin/bash

MAX=120
IPT=/sbin/iptables
IP6T=/sbin/ip6tables
TC=/sbin/tc
IF_EXT="eth0"

$TC qdisc add dev $IF_EXT root handle 1: htb

$TC class add dev $IF_EXT parent 1: classid 1:1 htb rate ${MAX}kbit ceil ${MAX}kbit

$TC class add dev $IF_EXT parent 1:1 classid 1:10 htb rate 30kbit
$TC class add dev $IF_EXT parent 1:1 classid 1:11 htb rate 30kbit
$TC class add dev $IF_EXT parent 1:1 classid 1:12 htb rate 30kbit
$TC class add dev $IF_EXT parent 1:1 classid 1:13 htb rate 30kbit

$IPT -A FORWARD -t mangle -s 192.168.0.3 -j MARK --set-mark 1
```

```
$IPT -A FORWARD -t mangle -s 192.168.0.4 -j MARK --set-mark 2
$IPT -A FORWARD -t mangle -s 192.168.0.5 -j MARK --set-mark 3
$IPT -A FORWARD -t mangle -s 192.168.0.6 -j MARK --set-mark 4

$TC filter add dev $IF_EXT parent 1: protocol ip handle 1 fw classid 1:10
$TC filter add dev $IF_EXT parent 1: protocol ip handle 2 fw classid 1:11
$TC filter add dev $IF_EXT parent 1: protocol ip handle 3 fw classid 1:12
$TC filter add dev $IF_EXT parent 1: protocol ip handle 4 fw classid 1:13
```

5.4. IMQ

IMQ es el acrónimo de *Intermediate Queuing Device* (Dispositivo Intermedio de Encolado), y no es más que un dispositivo virtual que nos ayudará a solucionar algunos problemas implícitos en el tratamiento de tráfico.

Una de las propiedades del control de tráfico que realizamos en nuestro router linux es que, SÓLO podemos manejar el tráfico de salida, puesto que el de entrada lo controla nuestro ISP. Así, con una política de control más o menos buena, y algún que otro arreglo mediante las respuestas a la congestión de TCP como el *comienzo lento* o el *decremento multiplicativo*, se puede conseguir una baja latencia para todos los usuarios de la LAN. Pero, algunas de las características que nos ofrece IMQ son:

1. Control del tráfico entrante.
2. Asignar políticas globales.

La segunda sería interesante en caso de tener conexión a Internet mediante varios ISP, ya que podríamos "fusionar" el tráfico de las dos salidas en un solo dispositivo imq y aplicarle a dicho dispositivo las políticas de QoS, en vez de a las dos `ethx`. La primera característica nos permitirá clasificar el tráfico de entrada (el que viene de Internet hacia nuestra LAN) en diversas qdisc o clases con diversas prioridades.

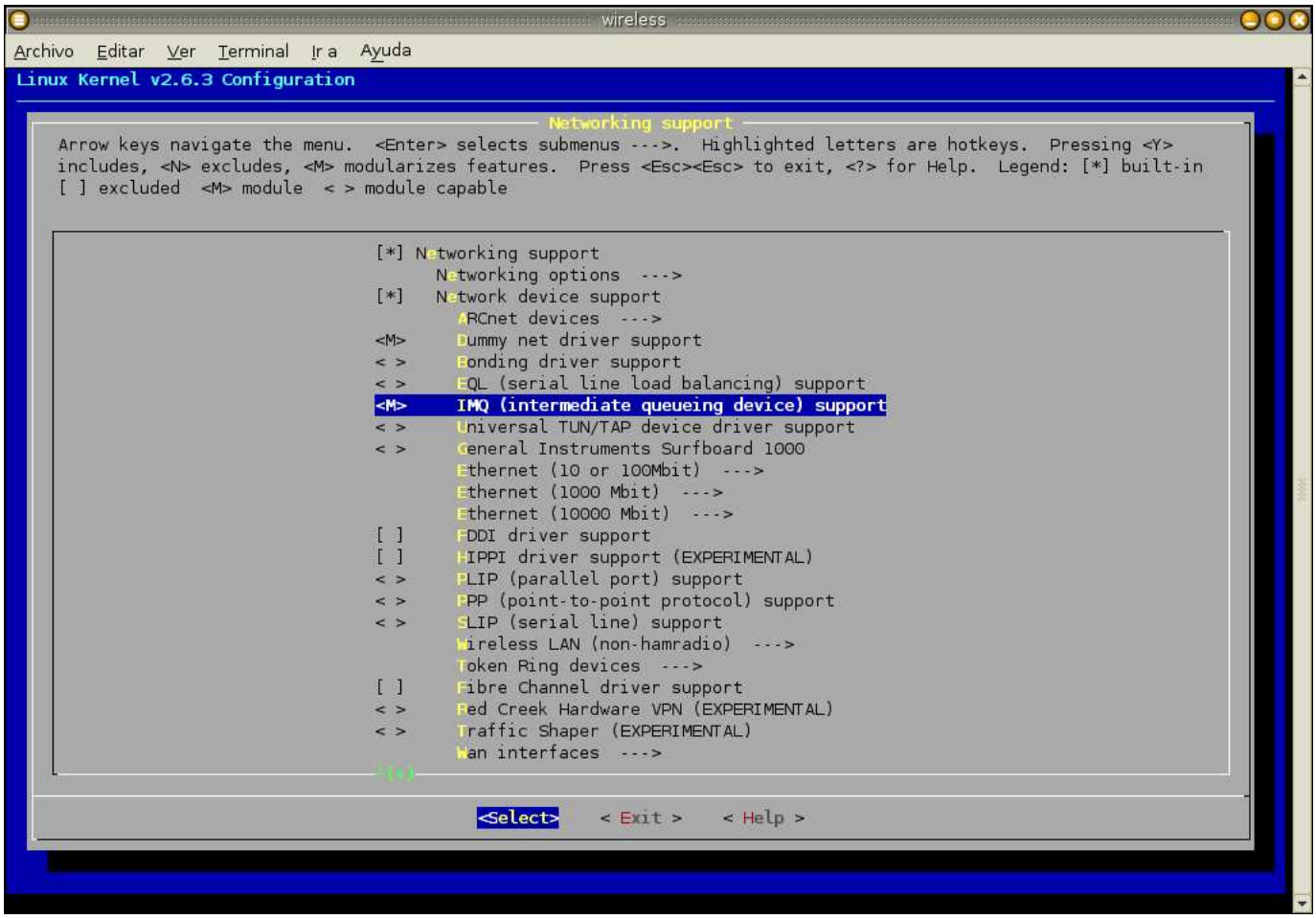
De momento en la versión del kernel que he utilizado para las pruebas, la 2.6.3, el soporte para IMQ no viene de serie, con lo que tendremos que parchear el kernel.

```
zen:/usr/src/linux# wget http://www.linuximq.net/patchs/linux-2.6.2-imq-4.diff
zen:/usr/src/linux# patch -p1 < linux-2.6.2-imq-4.diff

patching file drivers/net/Kconfig
patching file drivers/net/Makefile
Hunk #1 succeeded at 110 with fuzz 1.
patching file drivers/net/imq.c
patching file include/linux/imq.h
patching file include/linux/netfilter_ipv4/ipt_IMQ.h
patching file include/linux/netfilter_ipv6/ip6t_IMQ.h
patching file include/linux/skbuff.h
Hunk #1 succeeded at 251 (offset 5 lines).
patching file net/ipv4/netfilter/Kconfig
patching file net/ipv4/netfilter/Makefile
patching file net/ipv4/netfilter/ipt_IMQ.c
patching file net/ipv6/netfilter/Makefile
patching file net/ipv6/netfilter/ip6t_IMQ.c
patching file net/sched/sch_generic.c
```

```
zen:/usr/src/linux# make menuconfig
```

> **Nota:** Si se va a utilizar IMQ en una máquina que realiza NAT, también deberemos parchear el `../drivers/net/imq.c` , con el parche con soporte para NAT que encontraremos en la web principal de IMQ. (http://www.linuximq.net)

Con esto ya tendremos el kernel listo para darle soporte a IMQ.



Aviso

NO compilar IMQ como módulo `<M>`, sino como `<*>` , sino el kernel dará un error cuando llegue a IMQ. De momento el *bug* aún no se ha solucionado.

5.4.1. Configuración

Por el momento vamos a hacer un *mirror* de la configuración de salida para la de entrada.

```
#!/bin/bash

echo "Starting IMQ"

# 10 -> ssh,syn,DNS,ipv6 (60kbits) --> PRIO 1
# 11 -> smtp,pop3,imap,news (40kbits) --> PRIO 2
# 12 -> http,https,ftp (100kbits) --> PRIO 3
# 13 -> emule... (100kbits) --> PRIO 4

IPT=/sbin/iptables
IP6T=/sbin/ip6tables
IP= /sbin/ip
TC=/sbin/tc
MAX=300

$TC qdisc add dev imq0 root handle 1: htb default 13

$TC class add dev imq0 parent 1: classid 1:1 htb rate ${MAX}kbit ceil ${MAX}kbit

$TC class add dev imq0 parent 1:1 classid 1:10 htb rate 60kbit ceil ${MAX}kbit prio 1
$TC class add dev imq0 parent 1:1 classid 1:11 htb rate 40kbit ceil ${MAX}kbit prio 2
$TC class add dev imq0 parent 1:1 classid 1:12 htb rate 100kbit ceil ${MAX}kbit prio 3
$TC class add dev imq0 parent 1:1 classid 1:13 htb rate 100kbit ceil ${MAX}kbit prio 4

$TC qdisc add dev imq0 parent 1:10 handle 100: sfq perturb 10
$TC qdisc add dev imq0 parent 1:11 handle 110: sfq perturb 10
$TC qdisc add dev imq0 parent 1:12 handle 120: sfq perturb 10
$TC qdisc add dev imq0 parent 1:13 handle 130: sfq perturb 10
```

> **Nota:** Notar que ahora la variable **MAX** vale **300** en vez de 120, ya que estamos tratando el tráfico de bajada de nuestro ISP

Con esto ya tenemos las qdisc y las clases creadas para el tráfico de entrada, tan sólo falta clasificar el tráfico en un sitio u otro.

```
$TC filter add dev imq0 parent 1:0 protocol ip prio 1 handle 1 fw classid 1:10
$TC filter add dev imq0 parent 1:0 protocol ip prio 2 handle 2 fw classid 1:11
$TC filter add dev imq0 parent 1:0 protocol ip prio 3 handle 3 fw classid 1:12
$TC filter add dev imq0 parent 1:0 protocol ip prio 4 handle 4 fw classid 1:13

#Activamos la interfaz imq0

$IP link set imq0 up

#Transladamos todo el tráfico entrante al dispositivo IMQ

$IPT -t mangle -A POSTROUTING -o eth1 -j IMQ --todev 0

#Empezamos con el marcaje de paquetes!!!

# ICMP

$IPT -t mangle -A POSTROUTING  -o eth1 -p icmp -j MARK --set-mark 1
```

```
$IPT -t mangle -A POSTROUTING  -o eth1 -p icmp -j RETURN

# SSH

$IPT -t mangle -A POSTROUTING -o eth1 -p tcp  --sport 22 -j MARK --set-mark 1
$IPT -t mangle -A POSTROUTING -o eth1 -p tcp  --sport 22 -j RETURN

$IPT -t mangle -A INPUT -i $IF_EXT -p tcp  --sport 22 -j MARK --set-mark 1
$IPT -t mangle -A INPUT -i $IF_EXT -p tcp  --sport 22 -j RETURN

...
```

Una vez activadas las reglas, comprobaremos que hay tráfico en las clases.

```
      zen:~# tc -s class show dev imq0

class htb 1:11 parent 1:1 leaf 110: prio 1 rate 40Kbit ceil 300Kbit burst 1650b cburst 1983b
 Sent 5700 bytes 78 pkts (dropped 0, overlimits 0)
 rate 89bps 1pps
 lended: 78 borrowed: 0 giants: 0
 tokens: 320599 ctokens: 51625

class htb 1:1 root rate 300Kbit ceil 300Kbit burst 1983b cburst 1983b
 Sent 1471845 bytes 4186 pkts (dropped 0, overlimits 0)
 rate 18790bps 53pps
 lended: 1534 borrowed: 0 giants: 0
 tokens: 50560 ctokens: 50560

class htb 1:10 parent 1:1 leaf 100: prio 0 rate 60Kbit ceil 300Kbit burst 1675b cburst 1983b
 Sent 328 bytes 5 pkts (dropped 0, overlimits 0)
 rate 2bps
 lended: 5 borrowed: 0 giants: 0
 tokens: 216001 ctokens: 51413

class htb 1:13 parent 1:1 leaf 130: prio 3 rate 100Kbit ceil 300Kbit burst 1727b cburst 1983b
 Sent 1464109 bytes 4094 pkts (dropped 0, overlimits 0)
 rate 18338bps 52pps
 lended: 2560 borrowed: 1534 giants: 0
 tokens: -12540 ctokens: 50560

class htb 1:12 parent 1:1 leaf 120: prio 2 rate 100Kbit ceil 300Kbit burst 1727b cburst 1983b
 Sent 1708 bytes 9 pkts (dropped 0, overlimits 0)
 rate 6bps
 lended: 9 borrowed: 0 giants: 0
 tokens: 129922 ctokens: 50133
```

5.5. Automatización al inicio

Para que se apliquen todas las reglas anteriores en el inicio de la máquina, lo mejor será meterlas en un script de este tipo:

```
#! /bin/bash

IPT=/sbin/iptables
TC=/sbin/tc
IF_EXT="eth0"

case "$1" in

start)

echo  "Habilitando las reglas de iptables y de QoS"

echo 1 > /proc/sys/net/ipv4/ip_forward

#Aqui pondremos las reglas de iptables y de tc

   ;;

stop)

    echo  "Eliminando reglas de iptables y de QoS"

       echo 0 > /proc/sys/net/ipv4/ip_forward

       for TABLE in filter nat mangle
       do
           $IPT -t $TABLE -F
           $IPT -t $TABLE -X
           $IPT -t $TABLE -Z
       done

       $TC qdisc del dev $IF_EXT root

   ;;

restart)

   echo "Recargando las reglas de iptables y de QoS"

$0 stop
$0 start
   ;;

*)

echo "Utilización: $0 {start|stop|restart}"
    exit 1
   ;;

esac

exit 0
```

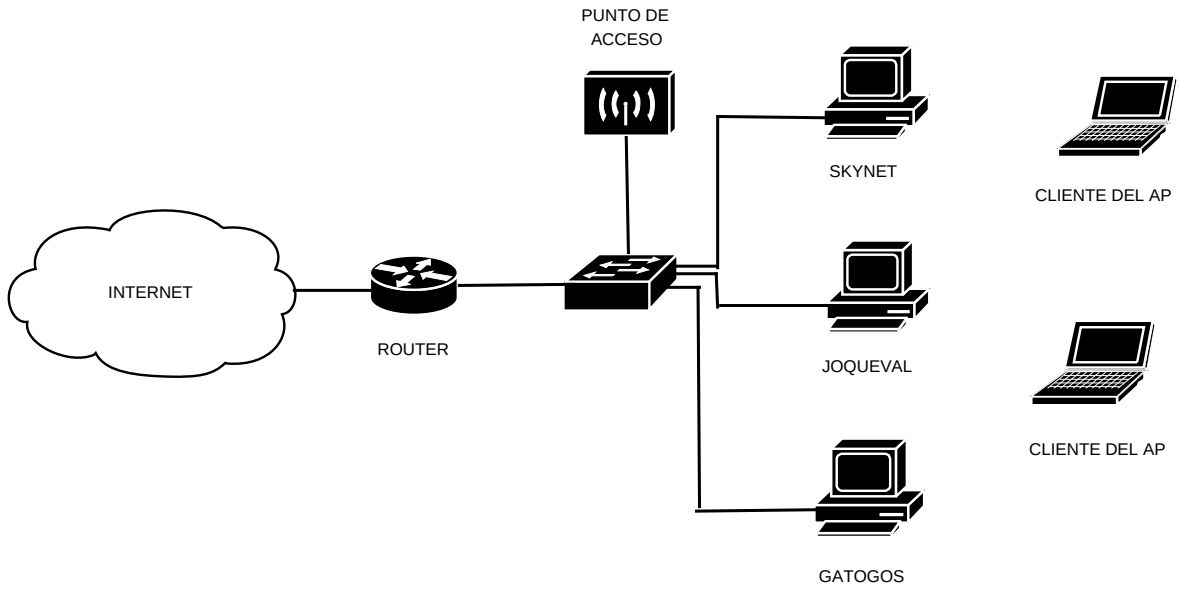
Una vez tengamos el script listo, le daremos permisos de ejecución, lo meteremos en `/etc/init.d/`, y crearemos el correspondiente enlace simbólico en el nivel de ejecución en el que arranca la máquina. Una vez arrancado, podremos parar las reglas, modificarlas y volverlas a arrancar sin necesidad de reiniciar.

> **Nota:** Resaltar que para conseguir una buena configuración, hay que jugar un poco con los valores y el tipo de clases utilizamos para cada caso. Cada usuario tiene diferente tipo de tráfico, diferente ISP, diferentes velocidades...

Apéndice A. Monitorización

No sólo de filtrado vive el home, así que podremos utilizar iptables junto con otras herramientas para otros menesteres que no sea el de filtrado puro y duro. Una de las utilidades bastante útiles es la de monitorización del tráfico en la LAN. Para esto, utilizaremos MRTG (http://www.ee.ethz.ch/~oetiker/webtools/mrtg/) y iptables.

En mi caso monitorizaré todo el tráfico que pase por el router, el individual de cada host, el tráfico wireless y el tráfico IPv6. Para empezar, implementaré NAT y las reglas de contabilidad IP para todo el tráfico que deseo monitorizar en el router.




```
#!/bin/sh

#ACTIVO EL FORWARDING EN EL KERNEL

echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 > /proc/sys/net/ipv6/conf/all/forwarding

IPT=/sbin/iptables
IP6T=/sbin/ip6tables

#ACTIVO EL NAT Y LAS REDIRECCIONES DE PUERTOS

$IPT -t nat -A POSTROUTING -o eth0 -j MASQUERADE

$IPT -t nat -A PREROUTING -i eth0 -p tcp --dport 4662 -j DNAT --to-destination
192.168.0.3:4662
$IPT -t nat -A PREROUTING -i eth0 -p tcp --dport 4663 -j DNAT --to-destination
192.168.0.4:4663
$IPT -t nat -A PREROUTING -i eth0 -p tcp --dport 4664 -j DNAT --to-destination
192.168.0.5:4664

#EMPIEZO CON LAS REGLAS DE CONTABILIDAD IP
```

```
$IPT -N SkynetOut
$IPT -N SkynetIn
$IPT -N GatogosOut
$IPT -N GatogosIn
$IPT -N JoquevalIn
$IPT -N JoquevalOut
$IPT -N WirelessIn
$IPT -N WirelessOut

$IP6T -N IPv6Out
$IP6T -N IPv6In

$IPT -A FORWARD -s 192.168.0.3 -j SkynetOut
$IPT -A FORWARD -d 192.168.0.3 -j SkynetIn
$IPT -A FORWARD -s 192.168.0.4 -j GatogosOut
$IPT -A FORWARD -d 192.168.0.4 -j GatogosIn
$IPT -A FORWARD -d 192.168.0.5 -j JoquevalIn
$IPT -A FORWARD -s 192.168.0.5 -j JoquevalOut
$IPT -A FORWARD -d 192.168.0.12 -j WirelessIn
$IPT -A FORWARD -s 192.168.0.12 -j WirelessOut

$IP6T -A FORWARD -i eth1 -j IPv6Out
$IP6T -A FORWARD -o eth1 -j IPv6In
```

Una vez implementadas, podremos comprobar que funcionan.

```
    zen:~# iptables -L -x -v -n

Chain INPUT (policy ACCEPT 534095 packets, 128105168 bytes)
    pkts      bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 40118717 packets, 14970956071 bytes)
    pkts      bytes target     prot opt in     out     source               destination
   31548  3821546 SkynetOut  all  --  *      *       192.168.0.3          0.0.0.0/0
   35895 39309847 SkynetIn   all  --  *      *       0.0.0.0/0            192.168.0.3
11921356 3093326878 GatogosOut  all  --  *      *       192.168.0.4          0.0.0.0/0
11311598 6067318847 GatogosIn  all  --  *      *       0.0.0.0/0            192.168.0.4
 7696020 2572191680 JoquevalIn  all  --  *      *       0.0.0.0/0            192.168.0.5
 7916195 2305685752 JoquevalOut  all  --  *      *       192.168.0.5          0.0.0.0/0
  653543 845175708 WirelessIn  all  --  *      *       0.0.0.0/0            192.168.0.12
  552368 44086447 WirelessOut  all  --  *      *       192.168.0.12         0.0.0.0/0

Chain OUTPUT (policy ACCEPT 663314 packets, 92129777 bytes)
    pkts      bytes target     prot opt in     out     source               destination



    zen:~# ip6tables -L -x -v -n

Chain INPUT (policy ACCEPT 60676 packets, 64048742 bytes)
    pkts      bytes target     prot opt in     out     source               destination

Chain FORWARD (policy ACCEPT 19984 packets, 14103717 bytes)
    pkts      bytes target     prot opt in     out     source               destination
```

```
    9138   691346 IPv6Out    all      eth1   *       ::/0                ::/0
   10846 13412371 IPv6In     all      *      eth1    ::/0                ::/0

Chain OUTPUT (policy ACCEPT 53813 packets, 4145909 bytes)
    pkts      bytes target     prot opt in     out     source               destination
```

Ahora tan sólo hace falta darle estos datos a MRTG de una forma que él lo comprenda para que pueda monitorizarlo. Utilizaremos unos scripts como estos:

```
    zen:~# cat skynet.data

#!/bin/sh

IPT=/sbin/iptables

statname="Tráfico de Skynet"
uptime=‘uptime‘
statin=‘/sbin/$IPT -L -n -x -v | /bin/grep -A 6 FORWARD | /bin/grep SkynetIn |
/usr/bin/awk ’{print $2}’‘
statout=‘/sbin/$IPT -L -n -x -v | /bin/grep -A 10 FORWARD | /bin/grep SkynetOut |
/usr/bin/awk ’{print $2}’‘

echo $statin
echo $statout
echo $uptime
echo $statname
```

Para el resto de hosts será igual pero cambiando los datos.

```
    zen:~# cat wireless.data

#!/bin/sh

IPT=/sbin/iptables

statname="Tráfico Wireless"
uptime=‘uptime‘
statin=‘/sbin/$IPT -L -n -x -v | /bin/grep -A 10 FORWARD | /bin/grep WirelessIn |
/usr/bin/awk ’{print $2}’‘
statout=‘/sbin/$IPT -L -n -x -v | /bin/grep -A 10 FORWARD | /bin/grep WirelessOut |
/usr/bin/awk ’{print $2}’‘

echo $statin
echo $statout
echo $uptime
echo $statname

   zen:~# cat ipv6.data

#!/bin/sh

IP6T=/sbin/$IP6T

statname="Tráfico IPv6"
```

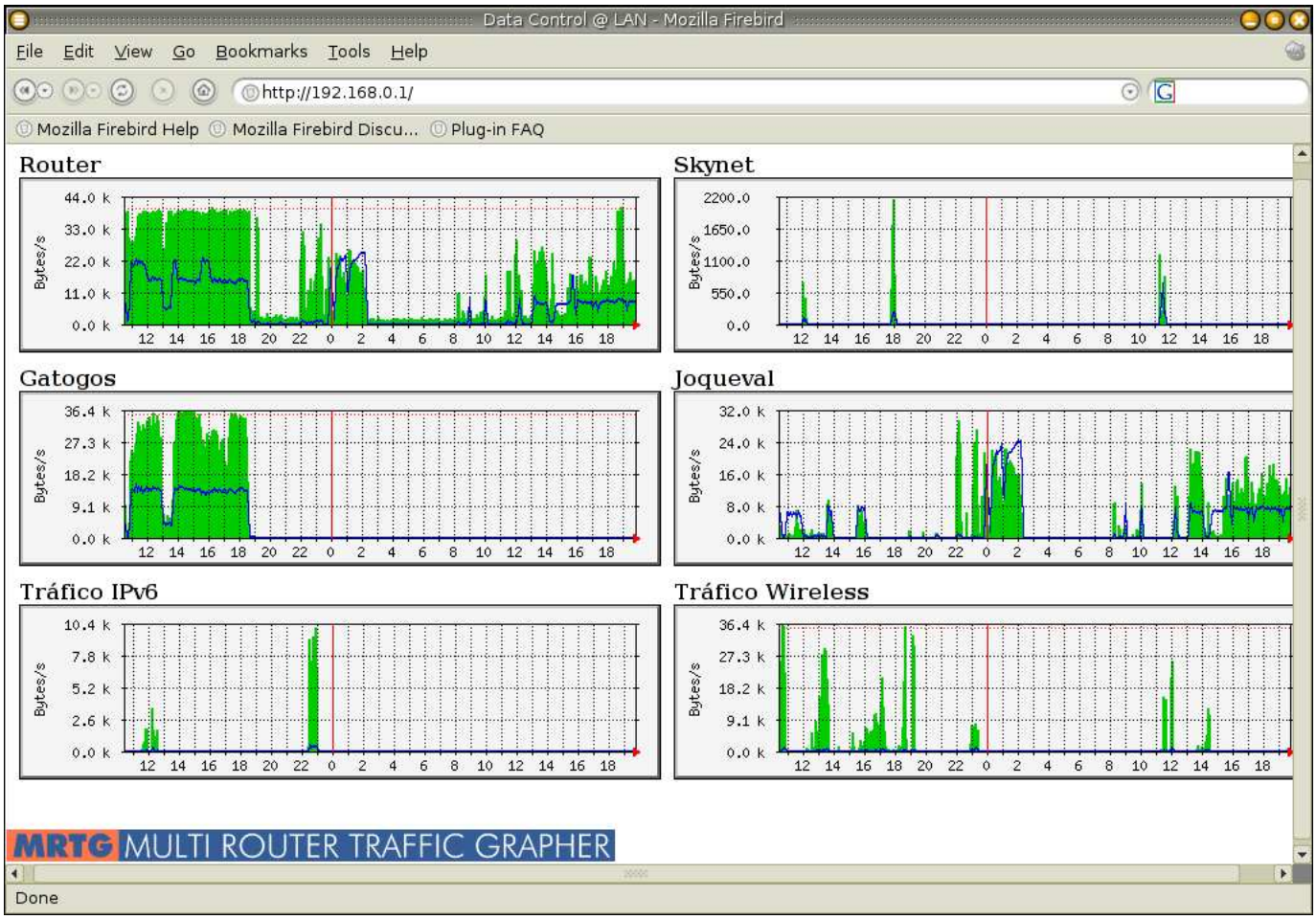
```
uptime=‘uptime‘
statin1=‘/sbin/$IP6T -L -n -x -v | /bin/grep  INPUT |  /usr/bin/awk ’{print $7}’‘
statin2=‘/sbin/$IP6T -L -n -x -v | /bin/grep -A 3 FORWARD | /bin/grep IPv6In |
/usr/bin/awk ’{print $2}’‘
statout1=‘/sbin/$IP6T -L -n -x -v | /bin/grep -A 3 FORWARD | /bin/grep IPv6Out |
/usr/bin/awk ’{print $2}’‘
statout2=‘/sbin/$IP6T -L -n -x -v | /bin/grep  OUTPUT | /usr/bin/awk ’{print $7}’‘

statout=$(($statout1+$statout2))
statin=$(($statin1+$statin2))

echo $statin
echo $statout
echo $uptime
echo $statname
```

Ya sólo nos queda configurar el archivo de configuración de MRTG, `mrtg.cfg`, recargarlo y juntar las gráficas.

```
    zen:~# cat /etc/mrtg.cfg

##################################################################
# This file is for use with mrtg-2.5.4c

# Global configuration


WorkDir: /var/www/mrtg
WriteExpires: Yes
Language: spanish
Title[^]: Control de Tráfico
Options[_]: growright

Target[eth0]: ‘/usr/bin/mrtg-ip-acct eth0‘
MaxBytes1[eth0]: 40000
MaxBytes2[eth0]: 50000
Title[eth0]: Análisis del tráfico total en eth0
YLegend[eth0]: Bytes/s
PageTop[eth0]: <H1>Router</H1>

Target[skynet]: ‘/etc/skynet.data‘
Title[skynet]: Zen
PageTop[skynet]: <H1>Skynet</H1>
MaxBytes1[skynet]: 35000
MaxBytes2[skynet]: 40000
YLegend[skynet]: Bytes/s
ShortLegend[skynet]: B/s

Target[gatogos]: ‘/etc/gatogos.data‘
Title[gatogos]: Zen
PageTop[gatogos]: <H1>Gatogos</H1>
MaxBytes1[gatogos]: 35000
MaxBytes2[gatogos]: 40000
YLegend[gatogos]: Bytes/s
ShortLegend[gatogos]: B/s

Target[joqueval]: ‘/etc/joqueval.data‘
```

```
Title[joqueval]: Zen
PageTop[joqueval]: <H1>Joqueval</H1>
YLegend[joqueval]: Bytes/s
MaxBytes1[joqueval]: 35000
MaxBytes2[joqueval]: 40000
ShortLegend[joqueval]: B/s

Target[ipv6]: ‘/etc/ipv6.data‘
Title[ipv6]: Zen
PageTop[ipv6]: <H1>Tráfico IPv6</H1>
MaxBytes1[ipv6]: 35000
MaxBytes2[ipv6]: 40000
YLegend[ipv6]: Bytes/s
ShortLegend[ipv6]: B/s

Target[wireless]: ‘/etc/wireless.data‘
Title[wireless]: Zen
PageTop[wireless]: <H1>Tráfico Wireless</H1>
MaxBytes1[wireless]: 35000
MaxBytes2[wireless]: 40000
YLegend[wireless]: Bytes/s
ShortLegend[wireless]: B/s
```

```
zen:~# mrtg /etc/mrtg.cfg
zen:~#indexmaker /etc/mrtg.cfg > /var/www/mrtg/index.html
```


Data Control @ LAN - Mozilla Firebird
File Edit View Go Bookmarks Tools Help
http://192.168.0.1/
Mozilla Firebird Help
Mozilla Firebird Discu...
Plug-in FAQ
Router
Skynet
Gatogos
Joqueval
Tráfico IPv6
Tráfico Wireless
Bytes/s
MRTG MULTI ROUTER TRAFFIC GRAPHER
Done

Apéndice B. Patch-o-Matic

Al igual que ocurre con el kernel, que se puede parchear para obtener nuevas funcionalidades que no vienen de *serie*, podemos hacer lo mismo con Netfilter. Es decir, podemos parchear Netfilter para que nos ofrezca nuevas posibidades de uso. Una vez parcheado y recompilado, podremos utilizar los nuevos módulos con la opción `-m` de iptables seguida del nombre del módulo. Con esto quedará cargado y listo para utilizar.

Los parches en cuestión los podemos encontrar en la web de Netfilter (http://www.netfilter.org). La mayoría con el tiempo acaba incorporandose al propio código del kernel, pero mientras tanto aquellos que quieran/necesiten configuraciones avanzadas pueden empezar a utilizarlos, y con un poco de suerte reportar algún *bug*.

Cuando nos bajemos los archivo y los descomprimamos, nos encontraremos con una serie de carpetas en las cuales nos encontraremos los parches:

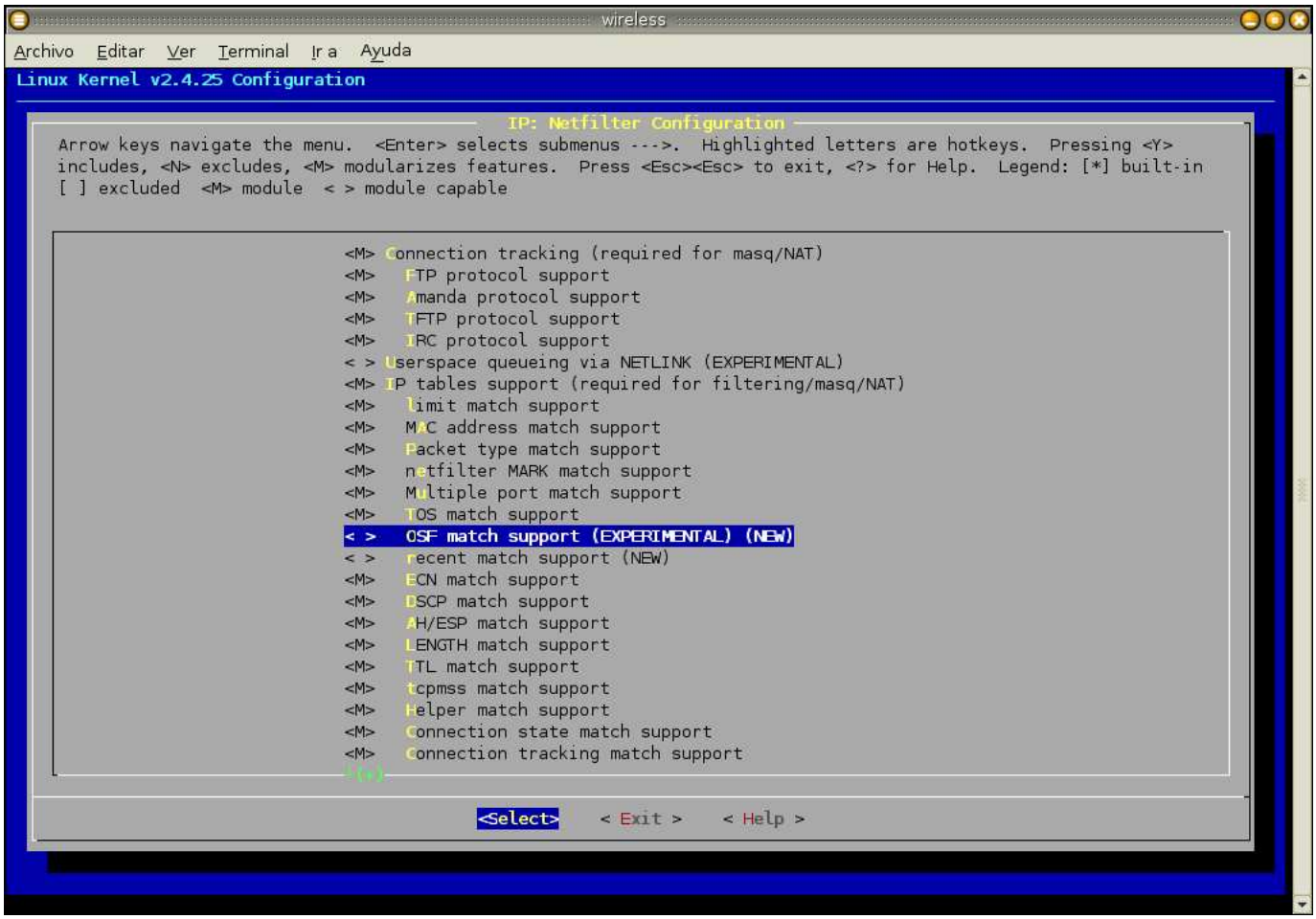
- Base
- Extra
- Obsolete
- Optimizations
- Broken
- Submitted
- Pending
- Userspace
- Oldnat

Los que más nos interesarán serán los de la carpeta `Base` y los de `Extra`. Los de la primera son los que fucionan bien de forma conjunta y los de la segunda son los que aparte de funcionar bien podrían causar algún conflicto. En `Submitted` encontraremos los parches que ya han sido enviados para formar parte del kernel, y en `Pending` los que lo harán en el momento en que se envien.

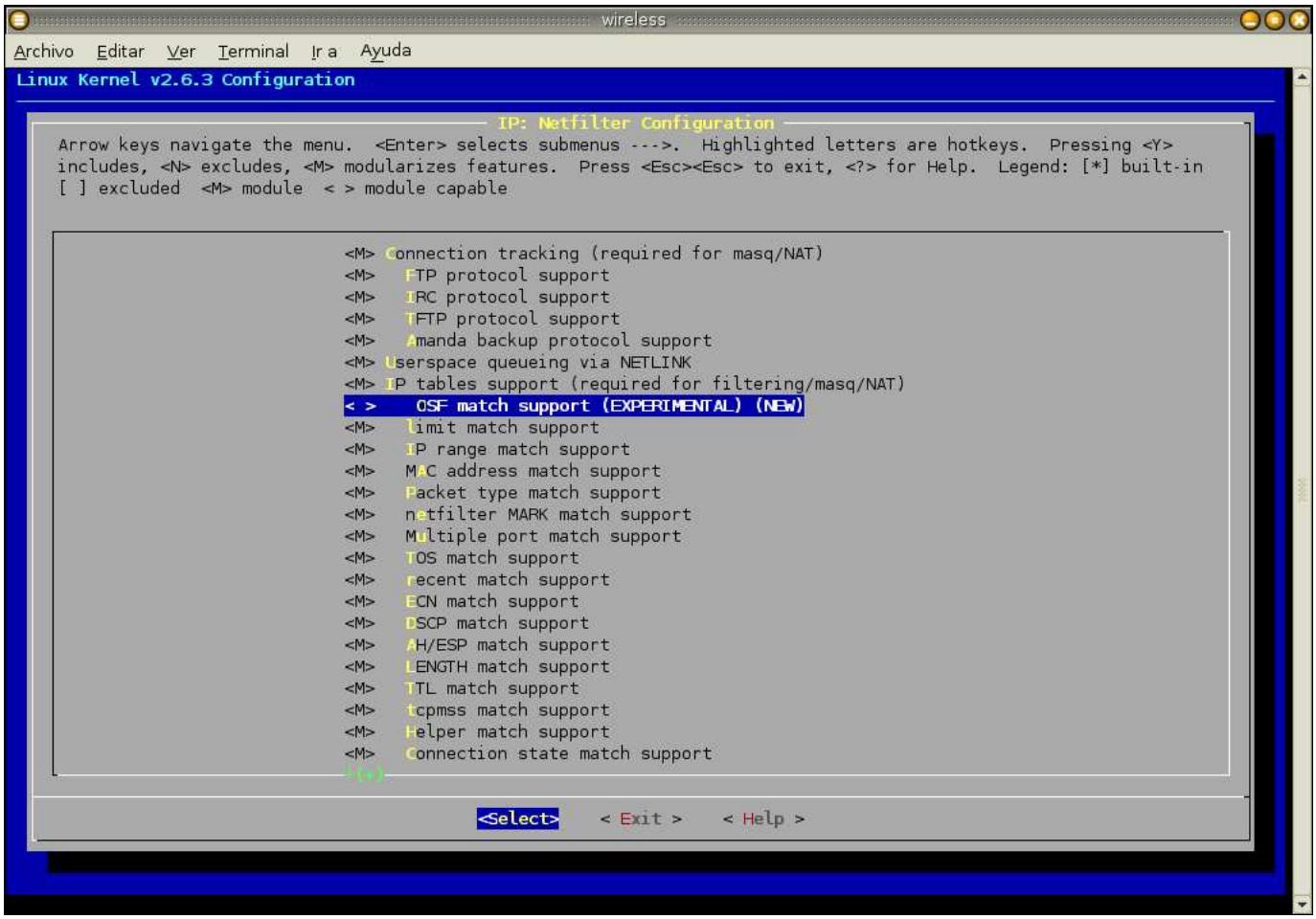
Para aplicar los parches, los *hackers* de Netfilter nos incluyen una herramienta, `runme`, con la que será muy fácil aplicar parches. Para utilizarlo tan sólo deberemos ejecutar el script e indicarle la carpeta donde se encuenta el parche.

```
snowball:/usr/src/linux/patch-o-matic# ./runme base/osf.patch
```

Podemos observar que tenemos disponibles nuevas funcionalidades que antes no teníamos, así que le daremos soporte y recompilaremos el kernel.



Nota: La actual versión de `runme` funciona con los kernels de la serie 2.4, aunque existe pom26convert (http://www.stearns.org/pom26convert/pom26convert), que lo soluciona mientras se reescribe `runme` para el soporte de las series 2.4 y 2.6.



Para saber cómo funcionan los módulos consultaremos el *.patch.help.

```
      snowball:/usr/src/linux/patch-o-matic# cat base/osf.patch.help

Author: Evgeniy Polyakov <johnpol@2ka.mipt.ru>

  The idea of passive OS fingerprint matching exists for quite a long time,
  but was created as extension fo OpenBSD pf only some weeks ago.
  Original idea was lurked in some OpenBSD mailing list (thanks
  grange@open...) and than adopted for Linux netfilter in form of this code.

  Original table was created by Michal Zalewski <lcamtuf@coredump.cx> for
  his excellent p0f and than changed a bit for more convenience.

  This module compares some data(WS, MSS, options and it's order, ttl,
  df and others) from first SYN packet (actually from packets with SYN
  bit set) with hardcoded in fingers[] table ones.

  Example: (Of course this only an example, do not get inspired by this)

  # iptables -N LINUX
  # iptables -A LINUX -j LOG --log-prefix "Linux"

  # iptables -A INPUT -p tcp -m osf --genre Linux -j LINUX
  # iptables -A INPUT -p tcp -m osf --genre FreeBSD -j REJECT
```

```
  NOTE: -p tcp is obviously required as it is a TCP match.

  OSF also has:
  --log 1/0.
If present, OSF will log determined genres even if they don't match
desired one.
0 - log all determined entries,
1 - only first one.

Example:
#iptables -I INPUT -j ACCEPT -p tcp -m osf --genre Linux --log 1 --smart

In syslog you find something like this:
ipt_osf: Windows [Windows XP Pro SP1, 2000 SP3]: 11.22.33.55:4024 -> 11.22.33.44:139
ipt_osf: Unknown: 16384:106:1:48:020405B401010402 44.33.22.11:1239 -> 11.22.33.44:80

  --smart
   if present, OSF will use some smartness to determine remote OS.
Now only not use TTL( with it far remote machines can be determined).

  If you say Y here, try iptables -m osf --help for more information.

  Fingerprints can be loaded through /proc/sys/net/ipv4/osf file.
  Only one fingerprint per open/close.

  Fingerprints can be downloaded from http://www.openbsd.org/cgi-bin/cvsweb/src/etc/pf.os

  Thanks to Maciej Soltysiak <solt@dns.toxicfilms.tv> for converting patches to patch-o-matic.
```

Apéndice C. GNU Free Documentation License

C.1. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

C.2. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the

above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

C.3. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute.

However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

C.4. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

C.5. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

C.6. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

C.7. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

C.8. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

C.9. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

C.10. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

C.11. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

C.12. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Bibliografía

[Eli00] *Building Internet Firewalls*, Elizabeth D. Zwicky, Simon Cooper, y D.Brent Chapman, O'Reilly & Associates, Inc., 1-56592-871-7, 2000.

[Ker24] *El Kernel 2.4 de Linux*, Fernando Sánchez y Rocio Arango, Prentice Hall, 84-205-3610-5, 2003.

[Rob01] *Linux Firewalls*, Robert L. Ziegler y Carl B. Constantine, New Riders, 0-7357-1099-6, 2001.

[Man] *Manual de Iptables*.

[Man6] *Manual de Ip6tables*.

[RH03] *Firewalls: El libro oficial de Red Hat Linux*, Bill McCarty, Anaya Multimedia, 84-415-1584-0, 2003.

[Netf] *www.netfilter.org (http://www.netfilter.org)*.

[MrtgD] *Monitorización del tráfico con MRTG : Apuntes sobre Debian GNU/Linux (http://www.neozero.net/linux/manuales/mrtg/)*.

[UnSec] *Seguridad en Unix y Redes (http://es.tldp.org/Manuales-LuCAS/doc-unixsec/unixsec-html)*, Antonio Villalón Huerta.

[Lartc] *Linux Advanced Routing & Traffic Control (http://lartc.org)*.

[ManTc] *Manual de tc**.

[Imq] *www.linuximq.net (http://www.linuximq.net/)*.

[Kei03] *Firewalls. Manual de referencia*, Keith Strassberg, Richard Gondek, y Gary Rollie, Mc Graw Hill, 84-481-3995-X, 2003.