Manual de Iptables.

Por Diego Lendoiro Rodríguez miembro del Inestable LUG
Este documento puede ser redistribuido y reeditado siempre que se mantenga al autor original.

Última revisión 21/04/05

Networking basics.

Antes de comenzar la inmersión en el mundo del filtrado de paquetes necesitamos definir una serie de conceptos previos que todo el mundo debe saber a la hora de comenzar a usar *iptables*. Dichos conceptos son, entre otros, el concepto de red, la topología de una red, etc.

Este será el coñazo que me toca explicaros antes de pasar a la chicha, así que sin más dilación ahí va.

¿Que es una red de ordenadores?

Consideramos una red de ordenadores como un conjunto de material que sirve para que uno o varios equipos (workstations) estén conectados entre si.

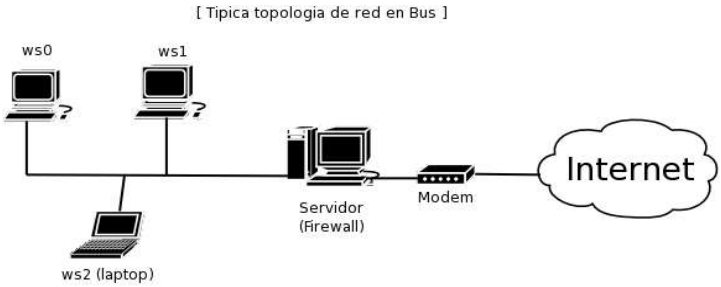
Este conjunto de material, hoy en día, está constituído frecuentemente por dos o más workstations cada uno con una tarjeta de red ethernet que ronda los 6€ y que varía entre los de 10/100 MB/s, 1000MB/s (Gigabit ethernet) o 10 000MB/s (menos comunes). Evidentemente además de las tarjetas ethernet necesitamos también cable, mucho cable.

NOTA: Generalmente cuando hablamos de red solemos hablar de más de dos ordenadores conectados entre sí.

Según el tamaño de las redes diferenciamos varios tipos como pueden ser LAN (Local Area Network), WAN (Wide Area Network), PAN (Personal Area Network) y muchas más, que con la proliferación de las redes sin cables (Wifi y bluetooth) han ido apareciendo.

Topología.

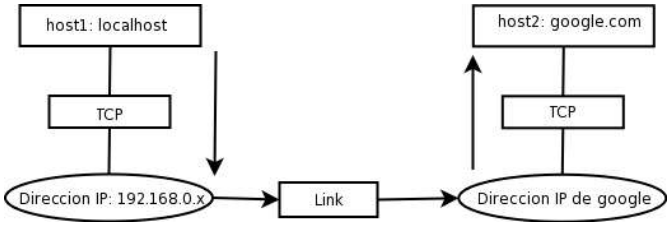
Las redes, según como estén conectadas físicamente obtienen distintas topologias (topología de estrella, token ring y un largo etc) sin embargo la que más prolifera a nivel de usuario y empresa pequeña/mediana es la topología en BUS dónde todas las estaciones de trabajo estan conectadas a una misma "tubería" por donde fluyen los datos.



En el caso de internet hablamos de la mayor WAN del mundo. En dicha WAN (en adelante internet) se necesitó estandarizar el protocolo utilizado para que los ordenadores se comunicasen entre ellos ya que a nivel de enlace existen muchos protocolos.

Para ello se utilizó un protocolo de red ahora conocido como IP (Internet Protocol) que hoy por hoy vive en su cuarta versión (ipv4) aunque existe ya su versión 6 (ipv6) que muchos esperan como agua de mayo.

Para que se aprecie un poco mejor como funciona la capa de aplicación y de enlace tenemos este diagrama:



Esto no es más que una petición común a lo que puede ser un host cualquiera (en este caso hemos puesto uno representativo como puede ser google.com) y los protocolos (a grosso modo) utilizados.

En la figura anterior aparece el protocolo TCP que es quien se encarga de iniciar la conexión (a muchos seguro que les suena el "TCP Handshake" que es como el "saludo" formal que se dan entre si los paquetes) y quien empaqueta los datos que se le pasan a la capa de enlace que mediante la IP enviará al host2 (en este caso google.com) hasta que se obtenga una respuesta.

Gracias a la IP es como podemos encontrar al destinatario/nodo/host al que entregarle el paquete TCP.

IP

Para que todo esto funciona el protocolo IP necesitará saber de alguna forma como "encaminar" los paquetes al host2 o de destino. Para ello cada host tonto en una red local como de cara a Internet deben tener una dirección IP, es decir como un "número de teléfono" identificativo que en este caso

consiste en 4 números separados por puntos.

Estos 4 números conforman una dirección de 32 bits como puede ser el clásico *192.168.0.1*

Estos numeros están siempre comprendidos entre **0** y **255**.

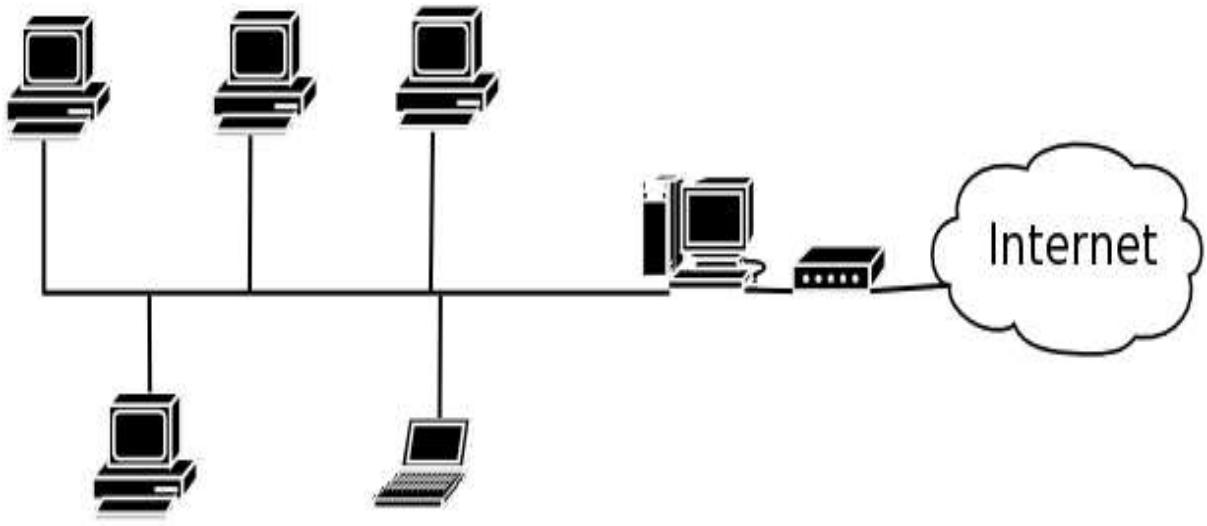
Evidentemente necesitamos de alguna manera distinguir entre lo que es nuestra red local de lo que es la "caótica y llena de peligros" a.k.a. Internet para ello entidades como la ICANN (Internet Corporation for Assigned Names and Numbers) se han encargado de hacer dicha distinción y existen una serie de direcciones IP que no es posible resolver en internet que son las consideradas como IP's privadas o rangos de IP's privados.
Un claro ejemplo de esto es el rango *192.168.0.0 (Clase C)* que es uno de los más utilizados a la hora de generar las direcciones de una red mediana pequeña

Vamos a diferenciar ahora un par de casos que suelen ser comunes en las redes domésticas.

Cuando tenemos un solo pc conectado a un módem (56kbps, ADSL, etc) la cosa es bastante fácil. El módem nos proporciona una ip a una de las interfaces de nuestro ordenador, interfaces que podemos fácilmente listar usando el comando *ifconfig* .

Dicha dirección será la que nos identifique en Internet y no tendremos mayor complicación a la hora de acceder a la misma, no tendremos que "encaminar" de ninguna manera los paquetes salvo usando las rutas por defecto (rutas que se pueden listar con el comando *route*)

Sin embargo este no es el caso que nos interesa, el esquema típico que nos interesa es el siguiente:



En este caso nuestro flamante PC con su maravilloso GNU/Linux instalado debe saber que si nosotros nos dirigimos a cualquier máquina dentro de nuestra LAN sin salir al exterior el paquete debe ser soltado sólamente en la LAN y no ser enviado a Internet y luego, cuando no se pueda resolver la dirección a la que es enviado enviarlo a la LAN (algo que sería bastante estúpido).

Sin embargo si nosotros queremos contactar con nuestro querido host2 (google.com) entonces nuestra maravillosa *tabla de enrutamiento* (route) debe decirle a nuestro paquete que debe salir a través de nuestra *puerta de enlace* o ***gateway*** que en este caso y en la mayoria de ellos será a todos los efectos nuestro *firewall* o el encargado de cuidarnos los paquetes :-)

Puertos.

Muchas veces queremos acceder a un servidor para obtener uno de sus servicios y solamente uno de ellos y no los demás. Para ello todos los servicios se organizan o mapean en todo ordenador de una forma y en esta forma de organizar los servicios entran en juego los puertos.

Pensemos en una máquina expendedora de... por ejemplo combinados con cierta graduación alcohólica (o no alcohólica para los abstemios) imaginemos que dicha máquina cuando pulsamos el 25 nos sirve un... Jack Daniel's con Cola y en el 110 nos sirve un Vodka con lima.

Suponemos que yo quiero el servicio que me da el número (puerto) 25 y no el del 110 entonces lo que hacemos es meter el dinero en la máquina (nos conectamos a ella) y le decimos que queremos lo que hay en el estante con el número 25 (nos conectamos al puerto 25). Pues esto mismo pasa con los servicios de un servidor o workstation.

Cuando hacemos un request a google.com para buscar algo estamos pidiéndole a la dirección IP de google (sea cual sea) en el puerto 80 (el standar del WWW) que nos busque fotos de Ana Obregón desnuda (Un ejemplo un poco macabro pero un ejemplo a fin de cuentas).

Es decir es como si en la expendedora pulsamos el 80 para obtener un licor de mora. Lo que pasa es que nuestro navegador nos hace transparente esa petición al puerto, porque por defecto él sabe que todo lo que escribamos en la barra de direcciones se refiere a servicios web que se hospedan en los puertos 80 de gran parte de los servidores del mundo y es por ello que si yo tecleo http://www.inestable.org me lleva a la página de nuestro LUG y si tecleo http://www.inestable.org:80 también lo que pasa es que sería una redundancia.

Lo mismo pasa con otros servicios que van a otros puertos como el SMTP que va al 25 el POP3 que suele estar asociado al puerto 110, etc.

El filtrado de paquetes: Iptables.

Despues de todo el rollo que hemos dicho en el capítulo anterior veremos ahora como se filtran los paquetes provinentes y con destino a internet y como, mediante algunas sencillas reglas podemos hacer que nuestra red esté un poco más segura.

Estudiaremos aquí el funcionamiento interno del sistema de filtrado de paquetes del kernel de linux (como los paquetes son procesados una vez que entran o cuando salen) y veremos también como es la sintaxis de la herramienta de filtrado que tenemos desde el kernel 2.4.x que no es otra que iptables. Para ello veremos una serie de reglas sencillas y alguna que otra no tan sencilla.

El procesado del paquete cuando llega a nuestra WS.

Bien, como ya deciamos antes un paquete tiene que pasar una serie de "pruebas" antes de llegar a su destino final. Por ejemplo un paquete provinente de Internet llega a través de nuestro cable y antes de que, finalmente, sea recibido por la aplicación que lo requería éste pasa por una serie de *tablas* en las que hai unas *reglas* o ***chains*** que indican que requisitos debe cumplir dicho paquete y en función de dichos requisitos el paquete tomará un camino u otro.

Cuando el destinatario del paquete es nuestro propio ordenador (en adelante *localhost*) nuestro pequeño visitante cruzará las tres tablas que existen por defecto que son las tablas *mangle, nat y filter.*

Brevemente explicaremos el cometido de dichas tablas:

mangle: El cometido de esta tabla no es mas que jugar un poco con los bits que posee el paquete y con ella podremos cambiar los campos de los paquetes. Como pueden ser el campo TOS (Tipe Of Service) TTL (Time To Live) y MARK que es usado para indicarle tareas avanzadas de enrutamiento.

nat: Esta tabla es usada para realizar el (Network Address Translation), es decir, esta tabla solo será usada cuando queramos cambiar el destinatario u origen del paquete.

filter: Esta es la tabla que se usa por defecto en el caso de no especificar ninguna de las anteriores (o cualquier otra que nosotros creasemos) y es donde normalmente realizaremos la mayor parte de las tareas con iptables, es decir, es con la tabla con la que mejor debemos llevarnos. Aquí es donde aceptaremos (***ACCEPT***) o denegaremos (***DROP***) paquetes .

Sintaxis.

Tras los funamentos teóricos (hay más) pero no los podemos explicar todos, no es el cometido de este manual hacer una inmersión profunda de todos los aspectos de funcionamiento de iptables, sino más bien dar una base al lector para que pueda generar sus propios firewalls a un nivel

aceptable.

Comandos:

-A cadena
Añade una o más reglas al final de la cadena especificada

-I cadena
Inserta una o más reglas al comienzo de la cadena especificada

-D cadena
Elimina una o más reglas de la cadena especificada que coincidan con la especificación de la instrucción.

-D cadena número_de_regla
Esta es mucho más comoda que la anterior, nos permite eliminar una regla de la cadena especificada simplemente indicando el número de la regla.

-R cadena número_de_regla
Lo mismo que la anterior pero en vez de eliminar reemplaza la regla.

-L cadena
Nos permite ver las reglas de la cadena especificada. Si no indicamos ninguna cadena mostrara las reglas de todas las cadenas.

-F cadena
Borra todas las reglas de la cadena especificada o todas las reglas si no se especifica ninguna cadena.

-Z cadena
Pone a cero todos los contadores de la cadena o de todas si no se especifica nada.

-N cadena
Permite crear una cadena creada por el usuario.

-X cadena
Elimina la cadena especificada creada por el usuario o todas las cadenas creadas por el usuario si no se especifica.

-P cadena política
Aplica una determinada politica por defecto a la cadena especificada como primer argumento.

Posibles políticas son **ACCEPT, DROP, QUEUE y RETURN. ACCEPT** permite pasar a los paquetes, **DROP** descarta los paquetes. Las otras dos (QUEUE y RETURN) no vamos a utilizarlas.

Opciones:

La mayor parte de las opciones que comentaremos aquí pueden ir precedidas de [!] (sin los corchetes) que sirve para indicarle al programa que busque coindicencias contrarias a lo especificado en la regla (negación).

Todo lo contenido entre los corchetes son posibles valores asociados a la opción que se comenta.

-p protocolo {tcp,udp,icmp}

Especifica el protocolo del paquete del que buscamos la coincidencia. De no especificar ningún protocolo se buscarán coincidencias con todos los protocolos.

-s {dirección/máscara}

Indica la dirección de origen del paquete la dirección puede ser indicada como un host tipo "inestable.org" o como una dirección ip en formato numérico. La máscara se puede indicar de la manera tradicional (p.ej 192.168.0.1/255.255.255.0) o empleando la notación moderna (p. ej. 192.168.0.1/24) el 24 representa los 24 primeros bits de la máscara cerrados si pusiesemos 32 sería una máscara totalmente cerrada, con 16 obtenemos los 16 primeros cerrados, etc.

-d {dirección/máscara}

Especifica la dirección de destino del paquete así como su máscara. La máscara se indica exactamente igual que en la opción -s de igual modo ocurre con la dirección ip

-j {ACCEPT,DROP,cadena_de_usuario}

Indica que acción se tomará cuando se encuentre una coincidencia con dicha regla. También se puede indicar una cadena creada por el usuaro por donde continuaría el proceso de filtrado.

-i {interfaz}

Indica la interfaz por donde se recibirá el paquete.

-o {interfaz}

Indica la interfaz por donde saldrá el paquete.

-f

Indica que la regla se aplicará a la segunda y sucesivas partes de un paquete fragmentado.

Extensiones del filtrado de paquetes.

Aquí trataremos una serie de opciones extra que iptables nos confiere, para ello debemos de invocar dichas extensiones.

Tenemos dos maneras de cargar las extensiones:

La explícita: **-m {tcp,udp,icmp,mac,mark,multiport,owner,limit,tos,ttl,...}**

O directamente usando la opción implicita y que el propio iptables se encargue de cargar las extensiones, como por ejemplo especificar directamente: "*-p tcp"* que provocará que se cargue la extensión tcp o lo que sería equivalente: "*-m tcp -p tcp*"

Ejemplos.

Antes de comenzar con el tema de la manipulación del TOS y del NAT creo que ya tenemos material suficiente como para empezar a ver como serían las reglas básicas de un firewall sencillo. Así que allá vamos.

```
root@badmotherfucker:~# iptables -L
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination
```

Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Primero listamos las reglas que podamos tener y de paso vemos que políticas siguen las cadenas típicas que son las que se ven arriba (INPUT, OUTPUT y FORWARD)

Supongamos por ejemplo estos puertos abiertos en nuestra máquina local:

root@badmotherfucker:~# nmap localhost

Starting nmap 3.81 (http://www.insecure.org/nmap/) at 2005-03-08 19:16 CET
Interesting ports on localhost (127.0.0.1):
(The 1660 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
21/tcp open ftp
22/tcp open ssh
6000/tcp open X11

Nmap finished: 1 IP address (1 host up) scanned in 0.194 seconds

Vamos a ver como por ejemplo filtramos el puerto 21:

iptables -A INPUT -m tcp -p tcp -s 0/0 --dport 21 -j DROP

Si ahora hacemos un *nmap -p 21 localhost* obtendremos esto:

PORT STATE SERVICE
21/tcp filtered ftp

¿Como evitar filtrarnos a nosotros mismos?

Fácil, iptables, busca en sus cadenas por coincidencias para cada paquete, en cuanto encuentra la primera se para y no busca más, entonces es obvio el método que debemos usar, primero debemos permitirnos a nosotros mismo el tráfico con nosotros y con internet y luego filtrar.

Entonces:

iptables -A INPUT -i lo -j ACCEPT
iptables -A INPUT -s 0/0 --dport 21-j DROP

y así obtenemos:

PORT STATE SERVICE
21/tcp open ftp
También podríamos permitirnos el paso vía nuestra ip local (127.0.0.1) sin embargo este método no es tan bueno como el anterior ya que si intentamos acceder via nuestra ip local (en el caso de que estuviesemos en una LAN y suponiendo que esta fuese, por ejemplo 192.168.0.1)

Supongamos el esquema típico ya antes comentado, un firewall basado en GNU/Linux y dos máquinas más en red con el mismo.

En el caso de que decidiesemos tener el firewall con una política estricta de denegación de paquetes usando:

iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

Deberiamos usar (siendo elegantes) una serie de filtros usando la maquina de estado con la extensión ***-m state*** que no veremos de momento.

Por ahora nosotros vamos a usar una política abierta en general de **ACCEPT** para todas las cadenas.

Siendo así y con el esquema anterior vamos a explicar (adelantando un poco de materia) como compartir conexión y como permitir el paso a los paquetes que vayan dirigidos a un servidor web (puerto 80) y ftp (puerto 21) que tendremos alojados en una de las workstations de nuestra red local.

Antes de nada si queremos activar el forwarding de paquetes necesitamos hacer un poco de setup en el directorio /proc

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

Y ahora ya si que podemos comenzar a poner en orden nuestro pequeño firewall

***iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ethx ***
-j MASQUERADE

Con esta línea le decimos a nuestro amigo iptables que use la tabla nat (network address translation) y que dentro de ella en la cadena POSTROUTING nos enmascare todas las direcciones de nuestra red local que salgan por la interfaz (ethx) que esté de cara a internet

Ahora para redirigir todas las peticiones que vayan al puerto a, por ejemplo la máquina de dirección 192.168.0.10 debemos hacer lo siguiente:

iptables -t nat -A PREROUTING -s 0/0 --dport 80 -j DNAT --to \ 192.168.0.10:80

Luego si quisieramos podríamos filtrar puerto a puerto, como por ejemplo filtrar el puerto 25

iptables -A INPUT -i lo --dport 25 -j ACCEPT
iptables -A INPUT -s 0/0 --dport 25 -j REJECT

La diferencia entre el objetivo DROP y el objetivo REJECT es que este último no procesa el paquete y luego lo descarta sino que lo ignora con lo cual el efecto que produce es el de un puerto cerrado y no el de un puerto filtrado, pero se pueden usar indistintamente.

Otra opción es filtrar todos, acompañaremos todo en un pequeño script esta vez:

#Comienzo del script

```
#!/bin/bash
#Sección de configuración
IPTABLES=`which iptables`

LAN_DEV="eth0"
INET_DEV="eth1"
LAN_RANG="192.168.0.0/24"
INTERNET="0/0"
#Fin de la sección de configuración
```

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

#Borramos todas las posibles reglas existentes
$IPTABLES -F
$IPTABLES -t nat -F

#Políticas
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT ACCEPT
$IPTABLES -P FORWARD ACCEPT

#Permitimos el tráfico exterior con la red local

$IPTABLES -A INPUT -i lo -j ACCEPT
$IPTABLES -A INPUT -i $LAN_DEV -j ACCEPT
$IPTABLES -A INPUT -s $LAN_RANG -d $INTERNET -j ACCEPT

#Permitimos la entrada a nuestro servidor web interno

$IPTABLES -t nat -p tcp -A PREROUTING -s $INTERNET --dport 80 -j\
DNAT --to 192.168.0.10:80

#Enmascaramos nuestras conexiones

$IPTABLES -t nat -A POSTROUTING -s $LAN_RANG -o INET_DEV\
 -j MASQUERADE

# Fin del script
```

Este es el clásico ejemplo de un firewall sencillo "non-statefull" es decir que no usa la extensión de estado que es la que nos permite saber en que estado se encuentra la conexión.

Últimamente con la proliferación de las redes wireless han, a su vez, aumentado/puesto de moda las ACL's (Access Control Lists o Listas de Control de Acceso) por MAC vamos a exponer tambien como iptables nos permite también hacer este tipo de curiosidades.

Para ello utilizaremos la extensión **-m mac** (Continuaremos usando los valores mencionados en el script anterior).

iptables -P INPUT DROP
iptables -A INPUT -i lo -j ACCEPT

Ahora en vez de permitir a todos los hosts de nuestra red el acceso al firewall solo se lo permitiremos a un host que tiene la siguiente dirección MAC:

00:08:02:E4:33:85

Entonces:

iptables -A INPUT -m mac --mac-source **00:08:02:E4:33:85 -d\ $INTERNET -j ACCEPT**

Si quisieramos negar dicha regla y que solo permitiese el paso a los que **no** tienen esa

dirección MAC no haría falta más que poner una exclamación [!] delante de la dirección MAC, dejando la regla tal que así:

iptables -A INPUT -m mac --mac-source !00:08:02:E4:33:85 -d
$INTERNET -j ACCEPT

Básicamente este es el funcionamiento de la ACL de un Access Point, filtra según una lista de MAC's que se le proporcionan a la hora de configurarlo, fácil, ¿no?

Vamos a dejar por ahora los ejemplos y seguir con un poco más de chicha, para hacer reglas un poco mas complicadas y eficientes.

Jugando con los bits de TOS.

Estos bits son un conjunto de 4 indicadores de la cabecera de cada paquete y según cual de ellos esté activado (Activado = 1) el firewall o router o encaminador puede tomar una acción distinta fiándose por cual de los indicadores sea el activo.

¿Y por qué se llaman indicadores de tipo de servicio? Pues porque indican a la red que tipo de servicio se requiere en cada momento, y los posibles tipos de servicio son:

Demora mínima: Da mayor importancia al tiempo que tarda el paquete en llegar desde su host origen a su host destino. Es decir, si el paquete tiene varias vías para llegar al host de destino esto provocará que se utilice la que menos demora produzca a la red.

Rendimiendo máximo: Cuando necesitamos transmitir datos y no queremos que se pierdan fragmentos a pesar del tiempo que requiera, ahí es cuando se usa este tipo de servicio. Servicios como el de FTP (File Transfer Protocol) requieren muchas veces de este tipo de servicio.

Fiabilidad máxima: Se usa cuando no queremos que los datos tengan que ser reenviados, que no se pierdan fragmentos.

Coste mínimo: Cuando resulta muy importante minimizar el coste de los datos transmitidos. El alquiler de ancho de banda de algún tipo de sistema de transmisión.

En iptables podemos buscar coincidencias y emprender acciones según el bit de TOS que esté activado con la extensión **-m tos** sería algo así:

-m tos --tos {valor} -j objetivo

Sin embargo no es quizás la característica mas interesante, sino que también podemos modificar los bits de TOS de los paquetes que entren/salgan de nuestra red y esto creedme es interesante (sobretodo si tienes un par leechers en casa y quieres usar ssh sin tener que contar los caracteres de cada línea).

Para este último cometido usaríamos algo como: **-j TOS --set-tos {valor}**

Dónde el parámetro entre corchetes tomaría uno de los siguientes valores:

Normal-service	**0x00** (valor de los bits de TOS por defecto)
Minimize-Cost	**0x02** (coste mínimo)
Maximize-Reliability	**0x04** (máxima fiabilidad)
Maximize-Throughput	**0x08** (máximo rendimiento)
Minimize-Delay	**0x10** (demora mínima)

Para activar ciertos bits de TOS necesitaremos usar la tabla mangle que se utiliza específicamente para "imprimir" en las cabeceras de los paquetes ciertos valores, como pueden ser en este caso los de TOS.

Esta línea por ejemplo es muy útil para hacer que el ssh no tenga "latencia", es decir, que cuando escribas el caracter no tarde 20 segundos en aparecer.

iptables -t mangle -A PREROUTING -p TCP --dport -j TOS --set-tos 0x10

Evidentemente si modificamos los bits de tos debe de ser antes de llegar a la decisión de enrutamiento, por ello usamos en la tabla mangle la cadena prerouting.

También podemos filtrar, buscando las coincidencias aka matches, de los paquetes según el bit de TOS que tengan activado. Esto sería así:

iptables -m tos --tos {valor} -j objetivo