Curso de Administración Linux

El curso se ha dividido en dos grandes bloques

- Gestión del sistema
  - Unidades 1 a 6
    - Control de servicios
    - Sistemas de archivos locales
    - Gestión de usuarios, grupos, permisos
    - Instalación y mantenimiento del software
- Administración en redes TCP/IP
  - Unidades 7 a 10

Bloque de Administración en Redes TCP/IP

- U7: 7-A) Las Redes TCP/IP e Internet.
  7-B) Configuración de la red.
  7-C) Seguridad *netfilter*: firewall y NAT.
- U8: Correo Electrónico.
- U9: Servicios WEB, FTP y Proxy.
- U10: Sistemas de Ficheros en Red:
  NFS y samba.

Unidad 7-A:
Introducción a las Redes TCP/IP

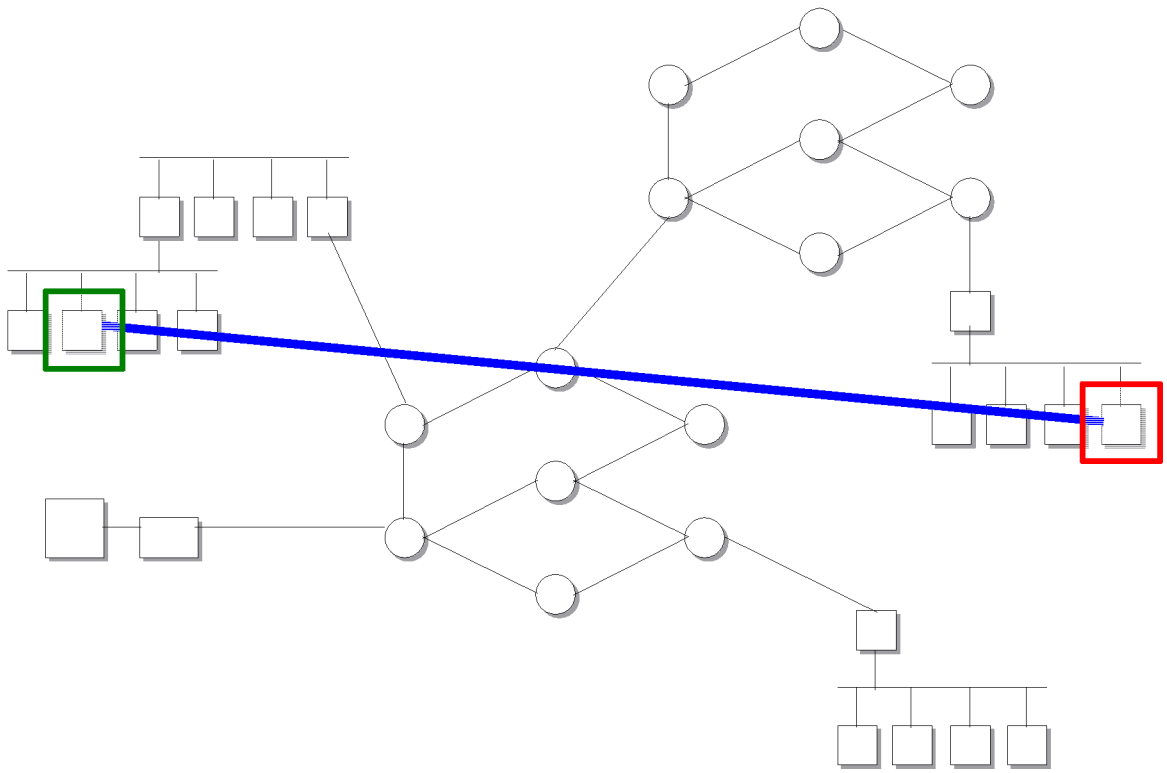
VII Curso de Introducción a la Administración de Servidores GNU/Linux
Extensión Universitaria. Universidad de Sevilla

Contenidos

1. Redes Informáticas

- Red: Conjunto de nodos conectados entre sí utilizando una serie de enlaces de datos.
- Nodos: ordenadores y otros dispositivos.
- Enlaces: cable, fibra óptica, radio, wifi...
- Tipos de Redes:
  - LAN: *Local Area Network* [**Red Local**]
    - Equipos conectados a corta distancia (~metros)
    - Alta velocidad y muy confiable
  - WAN: *Wide Area Network*
    - Equipos conectados a enormes distancias (~km)
    - Velocidades muchos menores

Redes Informáticas



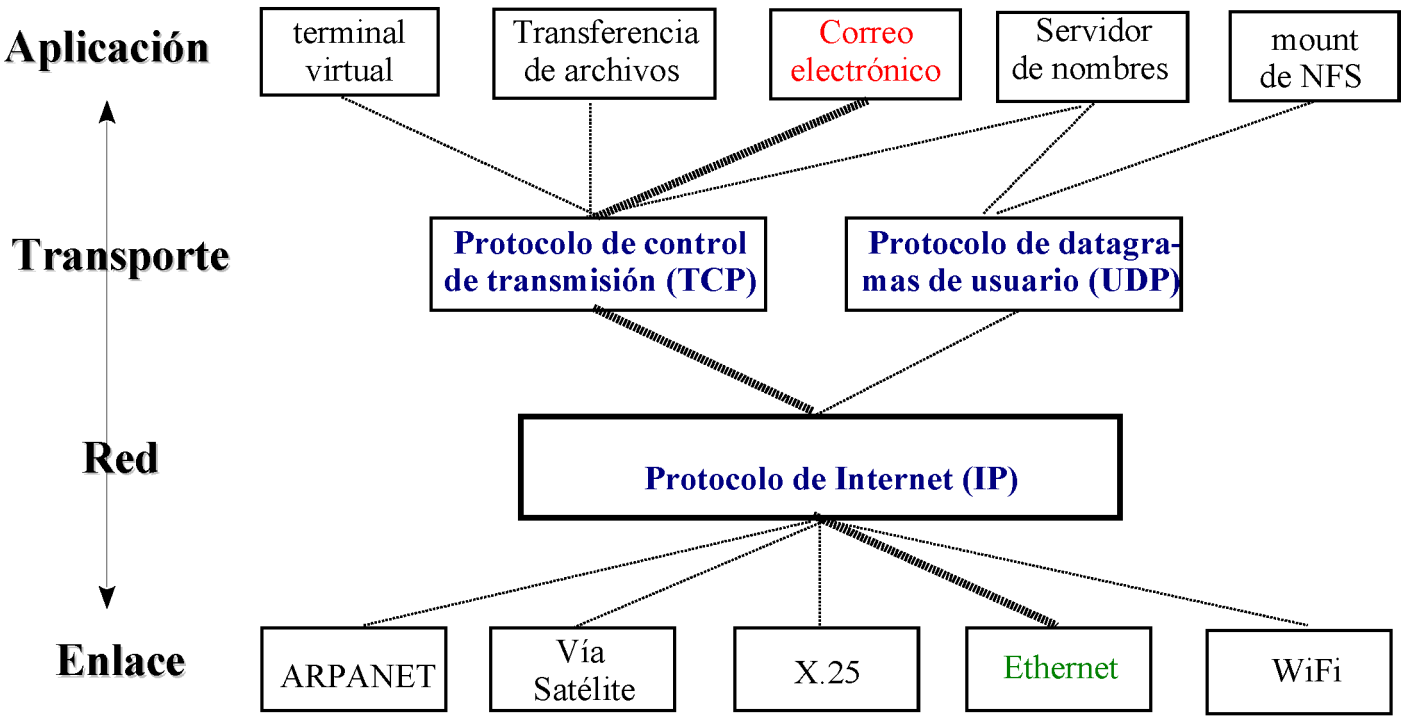
2. Redes TCP/IP

- Para que los nodos de una red se comuniquen, deben seguir una serie de protocolos (reglas de comunicación).
- **TCP/IP** es un conjunto de protocolos muy extendido, usado tanto en Internet como en muchas redes locales.
- Se implementa en todo tipo de ordenadores y otros dispositivos electrónicos.
- IP viene de “Internet Protocol”.

El TCP/IP

- TCP/IP quiere hacerle la vida más fácil al usuario o la aplicación
  - Divide la información en paquetes
  - Decide de qué forma enviar los paquetes (*routing*)
  - Asegura la entrega de los datos
  - Mantiene las conexiones establecidas
  - Detecta errores y si es necesario reintenta
  - Es independiente del tipo de red o fabricante
- Protocolos definidos: TCP, UDP, ICMP e IP.
- Versiones: IPv4 (actual), IPv6 (el futuro)

Arquitectura TCP/IP



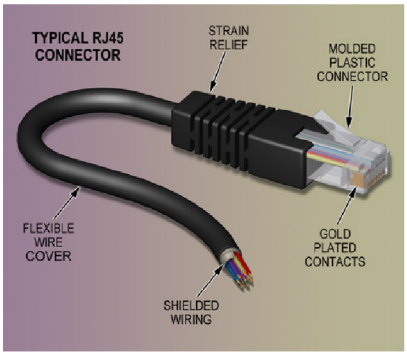
Internet

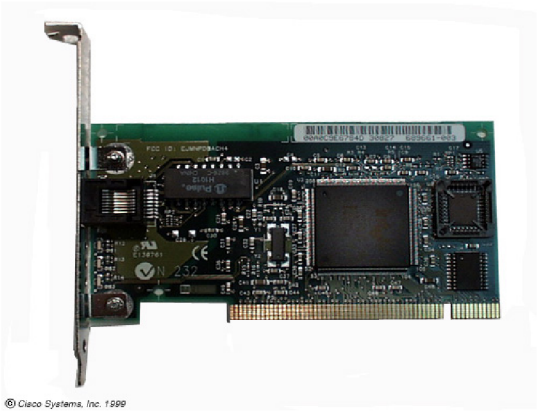
- Es una 'red de redes'
- Interconecta miles de redes heterogéneas distribuidas por todo el globo
- Gracias al protocolo TCP/IP todo parece una única red: *"La Red"*
- ¿El motivo del éxito? Aprovecha las redes existentes, muestra lo heterogéneo como homogéneo y permite comunicaciones entre todos los equipos conectados a Internet.

3. Elementos de la red

TARJETA DE RED ETHERNET

- Conecta el equipo en la LAN **Ethernet** (cableada)
  - Conector RJ45 + cable par trenzado





Elementos de la red

TARJETA DE RED INALÁMBRICA

- Conecta el equipo en la LAN **WiFi** (*wireless*)
  - WiFi integrado o externo (usb o pcmcia)
- Tarjeta WiFi con antena (integrada o externa)

Elementos de la red

HUB y SWITCH

- Concentran las transmisiones, estructuran una red local (LAN)
  - Hub: lo que envía uno lo reciben todos (muchas colisiones, menos privacidad)
  - Switch: comunicación uno a uno (pocas colisiones, mayor fiabilidad)

Elementos de la red

MODEM y ROUTER

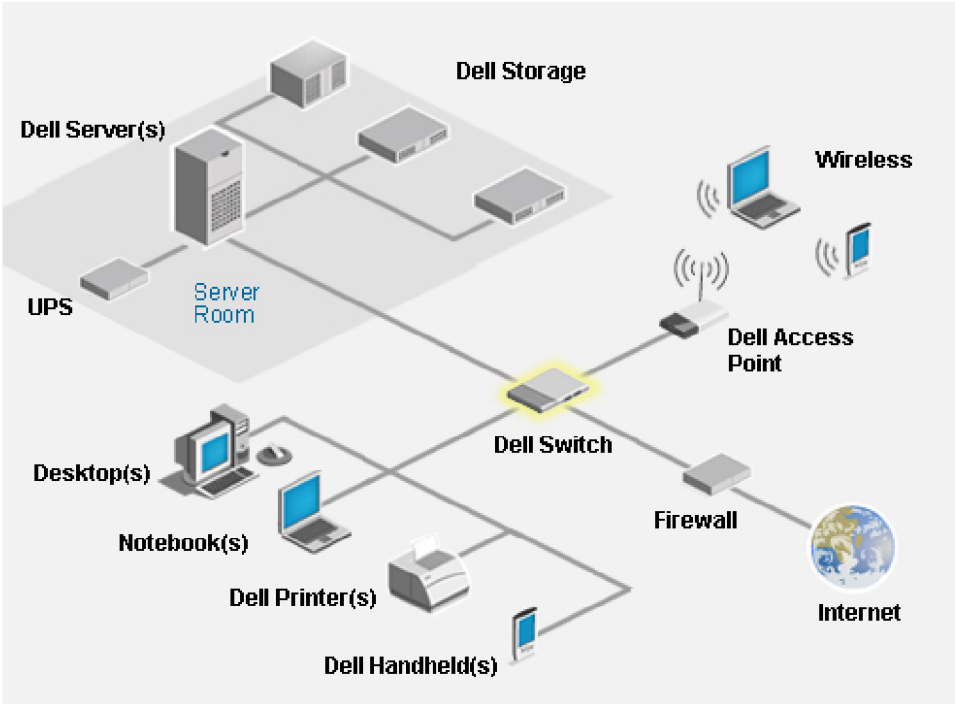
- Inter-Conexión de redes
- Módem: conecta un equipo a una red externa (da la dirección IP al equipo)
- Router: conecta redes completas entre sí (tiene una IP en cada red)

Elementos de la red

PUNTO DE ACCESO (A.P.)

- Conecta redes WiFi entre sí
- Generalmente también conecta las redes WiFi con una LAN cableada (Ethernet)
- Es un tipo particular de switch o router

Ejemplo de Red Local corporativa



4. Organización IP

4.1 Dirección IP

4.2 Subredes:

netmask, *broadcast,* dirección de red y *gateway*

4.3 Direcciones reservadas

4.4 Nombres y Dominios

4.5 Puertos y Servicios

4.1 Dirección IP

- Cada equipo conectado a una red TCP/IP tiene su dirección IP
- La dirección es única y le identifica en la red
- **IPv4**: Compuesta por 32 bits, representado como 4 números de 0-255 (~4 mil millones).
- Ejemplo:
  - Un dirección IP es 150.214.141.122
  - Bit a bit sería: 10010110.11010110.10001101.1111010
- Nota IPv6: (*¿muy pronto?*) cada IP tiene 128 bits!

4.2 Subredes: *netmask*

- Se forman **subredes** agrupando direcciones IP contiguas
- La máscara de red (***netmask***), de 32 bits, designa todos los bits que son de red [1] y los que se utilizan para cada equipo [0]

  Ejemplo:

  - IP 150.214.141.122 y Netmask 255.255.255.0
- Otra notación sería poner el "*nº de bits a uno*" de la netmask, separado con /, de esta forma:

  - 150.214.141.122/24

Dirección de red y *broadcast*

- La primera IP de la subred es la **dirección de red** y, junto con la máscara, sirve para definir la subred.
- La última IP de la subred es la dirección de **broadcast**, se usa para mandar un mensaje a todos los equipos.
- Ejemplo:

  IP 150.214.141.122 y Netmask 255.255.255.0 :

  - 150.214.141.0 es la dirección de red
  - 150.214.141.0/24 define la subred
  - 150.214.141.255 es el broadcast de la subred

Operaciones Bit a Bit

- Ejemplo: **150.214.144.12/24**

IP	10010110	11010110	10010000	00001100	150.214.144.12
Netmask	11111111	11111111	11111111	00000000	255.255.255.0
Red	10010110	11010110	10010000	00000000	150.214.144.0
Broadcast	10010110	11010110	10010000	11111111	150.214.144.255

Clases de red

- Las subredes se clasifican en tres tipos:
  - Clase C: netmask /24, red de 256 IP's (254 útiles)
  - Clase B: netmask /16, red de 256^2 IP's (~65 mil)
  - Clase A: netmask /8, red de 256^3 IP's (~16 Mill.)
    - Ejemplo: ***"La red de clase C 150.214.141.0"*** equivale a decir *"150.214.141.0/24"* y también es lo mismo que *"150.241.141.0 con netmask 255.255.255.0"*
- Hoy día se puede hacer *subnetting* de cualquier tamaño.
  - Ej: 150.214.141.0/25 y 150.214.141.128/25
  - Ej: 10.1.7.140.0/22=10.1.7.140.0/255.255.252.0

Puerta de enlace

- Un ordenador de una subred puede enviar datos de forma directa al resto de equipos de la misma subred
- Para comunicarse con equipos de otras subredes existe siempre un **Gateway** (Puerta de enlace) que conoce la forma de llegar hasta otras subredes
- Por convenio se *suele* usar la primera dirección libre de la red para el Gateway
- Ejemplo:
  - 150.214.141.1 es Gateway de 150.214.141.0/24

IP's dinámicas: DHCP

- Las direcciones IP de una red se pueden asignar de dos formas:
  - una IP va configurada siempre en el mismo equipo (direccionamiento **estático**)
  - se van configurando a medida que se van necesitando en la red (direccionamiento **dinámico**)
- *Dynamic Host Configuration Protocol*
- El protocolo DHCP se utiliza para asignar dinámicamente direcciones IP (junto con máscaras, gateways, DNS, etc...) en una LAN.

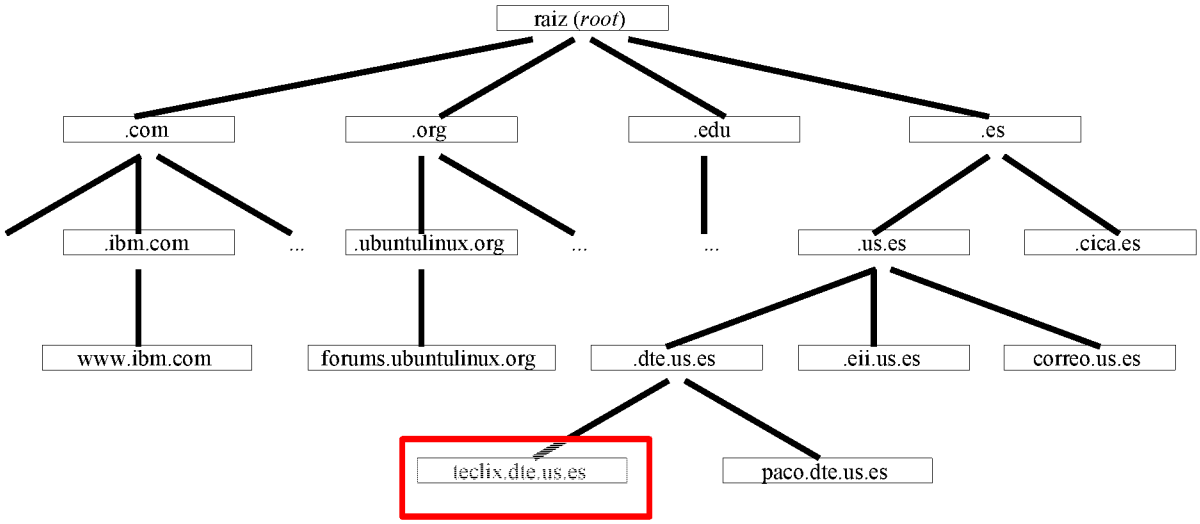
4.3 Direcciones reservadas

- Existe algunas direcciones reservadas para su uso en subredes 'privadas' (**intranets**).
- Están reservadas las siguientes subredes:
  - la “clase A” 10.0.0.0/8
  - desde 172.16.0.0/16 hasta 172.31.0.0/16
  - desde 192.168.0.0/24 hasta 192.168.255.0/24
- No son direccionables desde Internet.
- Desde la intranet se podría acceder a Internet usando un Proxy o con traducción de direcciones (NAT).

4.4 Nombres y Dominios

- Cada equipo se identifica, además de por la dirección IP, por un nombre.
- Los nombres se agrupan en dominios y subdominios, en un orden jerárquico.
- Suele existir una relación unívoca entre la IP y el nombre y el dominio completo.
- Ejemplo:
  - Un PC 'saturno' tiene la IP 150.214.141.122,
  - los equipos de la red 150.214.141.0 pertenecen al subdominio dte.us.es (del dominio us.es),
  - 150.214.141.122 <-----> saturno.dte.us.es

Jerarquía de Dominios



Servidores de Nombres

- El servidor de nombres (DNS) es un servicio que sirve para la 'traducción' de un nombre completo a su dirección IP y viceversa.
- Es casi imprescindible para un cómodo manejo de Internet y las redes TCP/IP.
  - Ej: 150.214.141.170 es la IP del PC que tiene instalado el servidor de nombres de la subred 150.214.141.0/24
  - Ej: Para poder ver la web www.google.com el navegador debe primero preguntarle al DNS por la IP, para poder conectarse con él. El DNS le responderá que la IP es 209.85.135.104

4.5 Puertos y Servicios

- Con las direcciones IP puede comunicar equipos conectados a una red. Pero necesito saber a que **servicio** concreto voy a acceder.
- Existen multitud de servicios standard, como DNS, web, ftp, smtp, pop3, ssh, telnet, etc...
- Cada servicio lleva asociado un puerto, identificado por un nº de 16 bits, de 0 a 65535. Además puede ser TCP o UDP.
- Ejemplos:
  - el servicio de web (http), usa el puerto 80/TCP
  - el servicio de DNS usa el puerto 42/UDP

Unidad 7-B: Configuración de la Red

VII Curso de Introducción a la Administración de Servidores GNU/Linux

Centro de Formación Permanente
Universidad de Sevilla

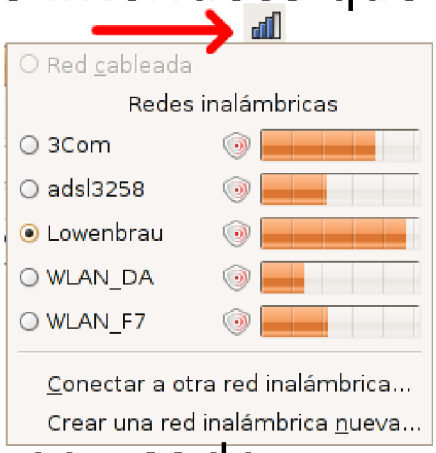
Contenidos

- Acerca del “Network Manager”
- Información para la configuración
- Detección del Hardware
- Configuración IP:
  - interfaz de red
  - nombre del host y anfitrioness
  - servicio de nombres
  - otros: dchp, wireless, modems, ...
- Comprobación de la red
- Introducción a los servicios de red
- Instalando un servidor de DHCP

1. Network Manager

Al instalar ubuntu viene un gestor de red llamado **network-manager**

Este intenta mantener al equipo siempre conectado por cualquiera de los interfaces que tenga a su alcance.



Se ejecuta un applet en el panel superior que nos permite conectar a la red que queramos y cambiar algunas configuraciones.

Pero en servidores recomiendo no usarlo:

$ sudo apt-get remove network-manager

2. Información para la configuración

Recopilamos esta información:

- dirección IP de la máquina
- máscara de la subred
- dirección IP del gateway (salida de la subred)
- dirección IP del/los servidor(es) de DNS
- nombre y dominio de la máquina

Ejemplo:

Nuestro PC va a tener la IP 150.214.141.196, en una subred de máscara 255.255.255.0, con la puerta de enlace 150.214.141.1. El nombre será 'teclix' en el dominio dte.us.es. Como DNS tenemos 150.214.186.69 y 8.8.8.8 (este último está fuera de la subred).

3. Detección del Hardware

Hardware de red: *tarjetas Ethernet* o *WiFi*

Para poder configurar la red, primero se deben cargar los drivers del kernel, generalmente en forma de módulos.

En kernels de Linux recientes es un proceso automático, se cargan en la detección durante el proceso de arranque.

Detección del hardware

Mensajes del kernel:

$ dmesg [“# dmesg -c” para borrarlo]

$ gnome-system-log (fichero kern.log)

Identificación del Hardware:

$ lspci [-v] # lshw

El administrador puede de forma manual cargar y descargar los módulos activos en el kernel:

$ lsmod

insmod, modprobe, rmmod

4.1 Configuración IP: Interfaces de red

Cada dispositivo de red se llama **interfaz**.

Cada *ethernet* se numera *eth0*, *eth1*, ...

Los *wireless* depende: *wlanX, ocX, raX, ...*

Siempre hay un *loopback*, con nombre “*lo*”

Las conexiones vía módem son *ppp0*, *ppp1*...

La información de configuración de los interfaces de red, en Linux basados en Debian (como Ubuntu), se guarda en el directorio **/etc/network**

El script de arranque es /etc/init.d/networking

/etc/network/interfaces (1)

```
$ cat /etc/network/interfaces
# Auto loading
auto lo eth0 eth2
# Loopback interface
iface lo inet loopback
# Red privada del laboratorio
iface eth0 inet static
        address 10.1.15.121
        netmask 255.255.252.0
        network 10.1.12.0
        broadcast 10.1.15.255
```

/etc/network/interfaces (2)

```
# Subred de profesores (acceso a internet)
iface eth1 inet static
        address 150.214.141.195
        netmask 255.255.255.0
        network 150.214.141.0
        broadcast 150.214.141.255
        gateway 150.214.141.1
        pre-up /usr/local/sbin/chequeoseguridad.sh

iface eth2 inet dhcp
```

Up y *down* de un interfaz

El proceso de activar o desactivar un interfaz en caliente se denomina *levantar* y *tirar* el interfaz (*up y down [o shutdown]*).

El administrador puede forzar unos determinados comandos a la hora de levantar o tirar un interfaz de red, colocando scripts en los subdirectorios de /etc/network:

if-pre-up.d/ *if-up.d/*

if-down.d/ *if-post-down.d/*

ifup / ifdown / ifconfig

Comandos:

ifup eth1

ifdown ppp0

* Activa (ifup) y desactiva (ifdown) la configuración del interfaz correspondiente, según el fichero */etc/network/interfaces* y los scripts en los subdirectorios *if-pre-up.d, if-up.d, if-down.d, if-post-down.d*

ifconfig [-a]

* Muestra valores de la interfaz de red o configura valores de red en una interfaz de forma manual.

Ej: ifconfig -a, ifconfig eth0, ifconfig eth0 10.1.4.5

4.2 Nombre de host

El nombre del propio equipo se guarda en /etc/hostname

El dominio se guarda en /etc/hosts

Comando **hostname**:

$ hostname -> saturno (nombre)

$ hostname -d -> dte.us.es (dominio)

$ hostname -f -> saturno.dte.us.es (completo)

hostname <nuevo_nombre>
-> cambia el nombre (se debe reiniciar)

Anfitriones (“hosts”)

En el fichero /etc/hosts se almacenan pares de direcciones IP y nombres de 'equipos locales' (también llamados “anfitriones”).

Se resuelven directamente, sin usar el DNS.

También se describe ahí el *localhost* (127.0.0.1, la IP del *loopback*) y también el dominio local del equipo.

Ejemplo: $ cat /etc/hosts

```
127.0.0.1       localhost   localhost.localdomain
150.214.141.122 saturno     saturno.dte.us.es
150.214.141.140 neptuno
```

4.3 Servicio de nombres (DNS)

DNS: Domain Name System

En /etc/resolv.conf se configuran los servidores de DNS

También se puede incluir un dominio local o un orden para la búsqueda de dominios.

Ejemplo-1: `$ cat /etc/resolv.conf`

```
domain dte.us.es
nameserver 150.214.186.69
```

Ejemplo-2: `$ cat /etc/resolv.conf`

```
search dte.us.es etsii.us.es
Nameserver 150.214.186.69
```

4.4 Otros: Tabla de Rutas

Con el comando **route** se puede consultar o modificar la tabla de rutas.

Ejemplo: $ route -n

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
150.214.141.0   0.0.0.0         255.255.255.0   U     0      0        0 eth1
10.1.12.0       0.0.0.0         255.255.252.0   U     0      0        0 eth0
default         150.214.141.1   0.0.0.0         UG    0      0        0 eth1
```

Salvo en redes complejas, los cambios se hacen de forma automática al ejecutar los comandos **ifup** e **ifdown**.

Protocolo DHCP

DHCP: Dynamic Host Configuration Protocol

Si hay un servidor DHCP en nuestra red bastará poner en /etc/network/interfaces:

```
iface eth0 inet dhcp
```

Al levantarse el interfaz se hace un broadcast pidiendo al servidor DHCP de la red una IP, el cual responderá con la que se le asigna, y entonces se configura el interfaz.

Ej: al levantar eth0 el servidor DHCP responde que debe configurarse la IP 150.214.141.44, netmask 255.255.255.0, gateway 150.214.141.1 y el DNS 150.214.141.2

"dhclient" solicita la IP al DHCP de la red.

Wireless

Lo fundamental es el paquete *wireless-tools*

Comandos específicos: **iwconfig** (como ifconfig), **iwlist** (lista redes, APs, ...)

```
# iwlist wlan0 scan
# iwconfig wlan0 essid Oficina
# dhclient wlan0
```

Una vez probado, se debe añadir en /etc/network/interfaces la configuración:

```
iface wlan0 inet dhcp
    wireless-essid Oficina
    wireless-mode ad-hoc
```

$ man wireless

Módem RTC

El protocolo es PPP (*Point to Point Protocol*)

Paquete **pppconfig** / **gnome-ppp** (gráfica)

Comandos: $ **pon** $ **poff** # **pppconfig**

Al conectar se configura la IP por DHCP.

Módem ADSL/Cable

El protocolo suele ser PPPoE (ó PPPoA) (*PPP Over Ethernet / ATM*)

Paquete **ppoeconf**

Básicamente funciona como un módem RTC, necesita validar usuario y contraseña (propio del PPP) y se asigna IP, DNS, *gateway*...

Comandos: **pppoe-discovery** y **pppoeconfig**

Router ADSL/Cable

El router es el dispositivo de red que se conecta a Internet, vía *PPPoE/PPPoA*, y el resto de la LAN le usa a él como *gateway*.

Así, en nuestro equipo sólo debemos configurar la red correctamente, poniendo el router como *gateway por defecto*.

Generalmente disponen de servidor DHCP por lo que ni siquiera tenemos que preocuparnos de elegir las direcciones de la subred privada.

GUI para Configuración de Red

En *Sistema* → *Administración* → *Red*

(paquete “gnome-network-admin”)

5. Comprobación de la red

Una vez todo configurado, vamos a comprobar que estamos en la red y tenemos conectividad con otros equipos e internet.

Varias comprobaciones:

- estado de las interfaces
- conectividad y rutado
- servicios de nombres

También disponemos de una herramienta gráfica para estas tareas (aunque no viene instalada por defecto)

Estado de las interfaces

```
$ ifconfig -a
```

eth0

Link encap:Ethernet HWaddr 00:4F:4E:05:FA:35

inet addr:150.214.141.122 Bcast:150.214.141.255 Mask:255.255.255.0

UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1

RX packets:9373307 errors:1801 dropped:993 overruns:80 frame:0

TX packets:8026804 errors:3583 dropped:0 overruns:0 carrier:7166

collisions:224583 txqueuelen:100

RX bytes:1764525259 (1682.7 Mb) TX bytes:3841778389 (3663.8 Mb)

Interrupt:9 Base address:0x4000

lo

Link encap:Local Loopback

inet addr:127.0.0.1 Mask:255.0.0.0

UP LOOPBACK RUNNING MTU:16436 Metric:1 ...

Conectividad entre equipos

La forma más sencilla, enviando un *ping* a otro equipo y esperando la respuesta.

Comprobaremos el *gateway*, *broadcast* y otros equipos de la red y de fuera.

$ **ping** <dirección_IP>|<nombre_maq>

Envía paquetes de forma sucesiva a una IP/nombre y mide el tiempo que tarda esta en responder.

Ejemplo:

```
$ ping 150.214.141.1
PING 150.214.141.1 (150.214.141.1): 56 data bytes
64 bytes from 150.214.141.1: icmp_seq=0 ttl=255 time=2.3 ms
64 bytes from 150.214.141.1: icmp_seq=1 ttl=255 time=3.9 ms
64 bytes from 150.214.141.1: icmp_seq=2 ttl=255 time=1.9 ms
....
```

"ping al broadcast"

$ ping *<ip_del_broadcast_de_mi_red>*

El kernel de Linux viene preconfigurado habitualmente para no responder a un ping al *broadcast*. Este comportamiento se puede desactivar

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

Este cambio se hará "en caliente", pero se pierde al reiniciar.

Para dejarlo **permanente** el cambio, se edita el fichero: /etc/sysctl y se añade/modifica la línea "net.ipv4.icmp_echo_ignore_broadcasts = 0"

Comprobando el Rutado

Aunque con **ping** podemos comprobar si llegamos a otras redes, **traceroute** nos da mucha información adicional sobre las rutas, sobre todo en redes WAN (o internet).

Ejemplo: $ **traceroute www.microsoft.com**

```
 1  150.214.141.2 (gw-141.us.es)  1.051 ms  0.568 ms  0.292 ms
 2  193.147.173.174 (193.147.173.174)  0.389 ms  0.304 ms  0.389 ms
 3  GE0-1-0.EB-Sevilla0.red.rediris.es (130.206.194.1)  0.359 ms  0.527 ms  0.391 ms
 4  * AND.SO4-1-0.EB-IRIS2.red.rediris.es (130.206.240.17)  11.586 ms  11.564 ms
 5  213.242.71.145 (213.242.71.145)  11.765 ms  11.895 ms  11.752 ms
 6  ae-0-51.mpls1.Madrid1.Level3.net (213.242.70.1)  11.990 ms  12.091 ms  12.265 ms
 7  ae-1-0.bbr2.London1.Level3.net (212.187.128.57)  90.061 ms  39.008 ms  39.070 ms
 8  ae-0-0.mp1.Seattle1.Level3.net (209.247.9.121)  174.553 ms 173.833 ms 202.173 ms
 9  ge-2-0-0-56.gar1.Seattle1.Level3.net (4.68.105.169)  173.826 ms 173.734 ms 173.736 ms
10  65.59.235.6 (65.59.235.6)  194.535 ms  194.134 ms  195.023 ms
11  gig3-1.tuk-76cb-1a.ntwk.msn.net (207.46.42.1)  194.232 ms  194.554 ms  194.230 ms
12  pos1-0.iuskixcpxc1202.ntwk.msn.net (207.46.36.146)  194.353 ms * 195.174 ms
13  pos1-0.tke-12ix-1b.ntwk.msn.net (207.46.155.5)  194.234 ms  194.026 ms  194.881 ms
14  po13.tuk-65ns-mcs-1b.ntwk.msn.net (207.46.224.217)  194.103 ms  * 194.731 ms
15  * * *
...
```

Comprobando el DNS

$ **host** <nombre|nombre.completo>

$ **host** <nombre.completo> <IP.del.servidor.DNS>

Realiza una consulta al servidor de DNS para resolver el nombre de host facilitado.

Ejemplo:

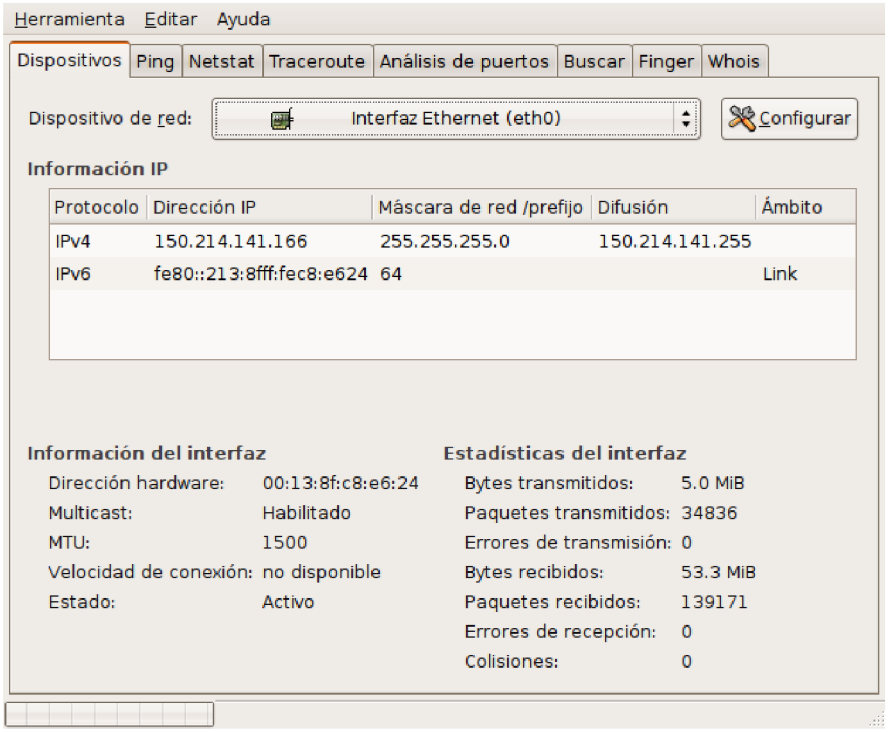
```
$ host www.ubuntulinux.com
www.ubuntulinux.com has address 82.211.81.166
```

$ **host** <direccion.IP>

Resolución inversa de la dirección IP.

Herramienta Gráfica

En *Sistema → Admón → Herramientas de Red*



Paquete “gnome-nettool”

6. Introducción a los servicios de red

Cada equipo de la red TCP/IP puede ofrecer el acceso a unos servicios al resto de equipos. Cada servicio lleva asociado un puerto.

El servicio se presta a través de un programa servidor específico (denominado de forma genérica “demonio” o *daemon*) o a través de *inetd*, un servidor genérico para lanzar servicios bajo demanda.

Generalmente los distintos *daemons* (incluido *inetd*) se controlan de forma similar.

Control de los servidores (I)

Las configuraciones se guardan en /etc/<servidor>

Ejemplo: El servidor web Apache2 guarda su configuración en /etc/apache2

El script de control es /etc/init.d/<servidor>

start: lanza el servicio

stop: detiene el servicio

restart: lo detiene y vuelve a lanzarlo

reload: recarga la configuración

Ejemplo: # /etc/init.d/apache2 stop

Control de los servidores (y II)

Recientemente se ha unificado el gestor de servicios:

service <servidor> <comando>

Ejemplos:

service cron restart

service apache2 status

service --status-all

El servidor INETD

Se controla como un *daemon* más, pero el propio inetd facilita el acceso a múltiples servicios, configurados en /etc/inetd.conf

Ejemplo: $ cat /etc/inetd.conf

```
#:MAIL:
smtp stream  tcp   nowait   mail /usr/sbin/exim exim -bs
#:FTP:
ftp  stream  tcp   nowait   root /usr/sbin/tcpd /usr/sbin/in.ftpd
```

Los servicios estandard están en el fichero /etc/services

Hoy día no se usa mucho, generalmente sólo cuando no hay *daemon* propio.

Puertos en servicio

Se pueden listar los puertos TCP/IP dónde hay un *daemon* 'escuchando' (esperando servicio), con ayuda del comando **netstat**

Ejemplo: **# netstat --inet -l -p**

```
Proto RQ SQ Local          Foreign          State       PID/Program name
tcp   0  0  *:ftp           *:*              LISTEN      6864/pure-ftpd
tcp   0  0  localhost:smtp  *:*              LISTEN      6852/exim3
```

Para probar los puertos de otros equipos se usará un *análisis de puertos*, como **nmap**

Estas funciones también están en la utilidad gráfica ya vista, 'herramientas de red'

Análisis de puertos

$ nmap 150.214.141.122

Starting nmap 3.81 (http://www.insecure.org/nmap/) at 2006-01-31 14:49 CET
Interesting ports on saturno.dte.us.es (150.214.141.122):
(The 1650 ports scanned but not shown below are in state: closed)
PORT STATE SERVICE
22/tcp open ssh
25/tcp open smtp
80/tcp open http
81/tcp open hosts2-ns
111/tcp open rpcbind
139/tcp open netbios-ssn
143/tcp open imap
443/tcp open https
444/tcp open snpp
445/tcp open microsoft-ds
929/tcp open unknown
993/tcp open imaps
995/tcp open pop3s

Nmap finished: 1 IP address (1 host up) scanned in 0.205 seconds

netcat

netcat es una herramienta versátil para hacer pruebas con puertos

Permite 'escuchar' en un puerto determinado o bien conectarse a un servicio establecido, interactuando con el usuario.

```
$ netcat -l -p 5555          (escucha en TCP/5555)
$ netcat localhost 5555      (se conecta al puerto 5555)
$ netcat correo 25  (se conecta al servidor SMTP)
220 correo ESMTP Exim 4.50 Thu, 02 Feb 2006 13:38:30 +0100
help
214-Commands supported:
214 AUTH STARTTLS HELO EHLO MAIL RCPT DATA NOOP QUIT RSET HELP
quit
221 correo closing connection
```

6. Instalación Servidor DHCP

- Instalar “dhcp3-server” (y “gdhcpd”)
- Script de inicio: /etc/init.d/dhcp3-server
- Configuración: /etc/dhcp3/dhcpd.conf
- Los “scopes” son las subredes en las que está el DHCP, y dentro del scope hay que definir los rangos de IP's compartidas (“shared Ips”).
- También se puede añadir una configuración particular para algunos equipos, para que el servidor les asigne siempre la misma IP, por ejemplo a partir de la dirección MAC.

Configuración Servidor DHCP

 # cat /etc/dhcp3/dhcpd.conf

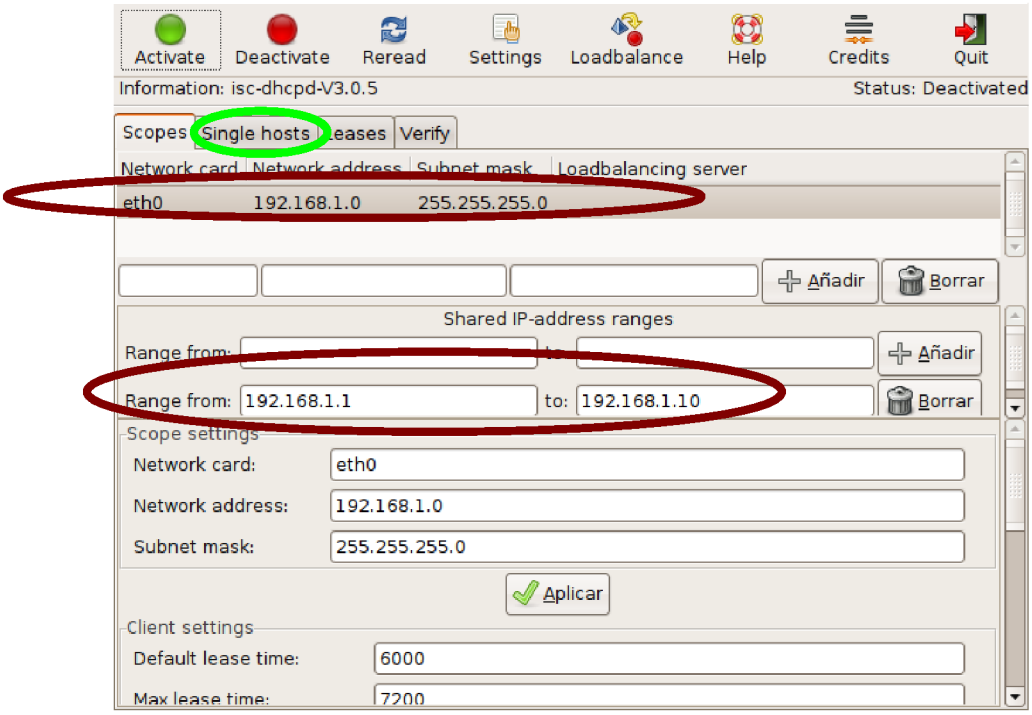
```
[...] option domain-name “dte.us.es”;
option domain-name-servers 10.0.1.70 212.59.120.33;
option routers 10.0.1.138;
subnet 10.0.1.0 netmask 255.255.255.0 {        # RED COMPLETA
   interface eth0;
   range 10.0.1.1 10.0.1.10;         # RANGO de IP's a ASIGNAR
}
host saturno {                # CONFIGURACIÓN DE UN EQUIPO CONCRETO
   hardware ethernet 00:9F:F5:0E:5A:30;   # DIRECCIÓN MAC
   fixed-address 10.0.1.50;                    # DIRECCIÓN IP FIJA
} [...]
```

 # cat /etc/default/dhcpd3-server

```
[...] INTERFACES="eth0"        # Para que se lance el demonio escuchando en
   eth0
```

GUI de configuración

Instalar “gadmin-dhcpd”. Se ejecuta desde Aplicaciones → Herramientas de Sistema.



Unidad 7-C: Seguridad con netfilter, NAT y Firewall.

Curso de Introducción a la administración de servidores GNU/Linux

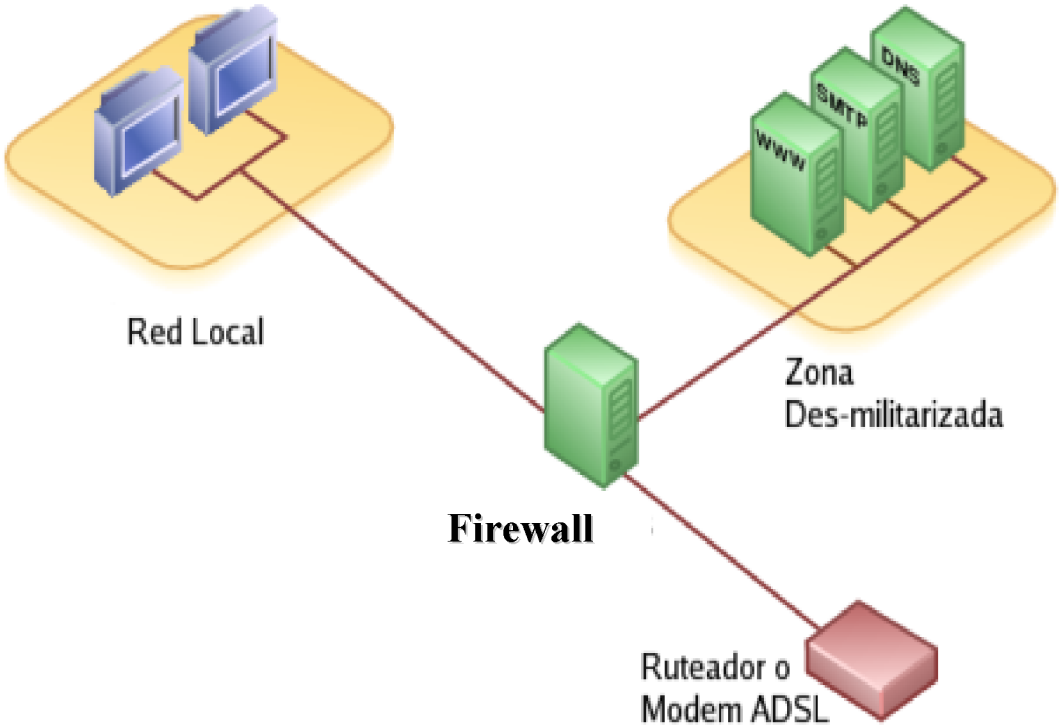
Centro de Formación Permanente
Universidad de Sevilla

Contenidos

1. Introducción

- *La red* no es segura, TCP/IP no lo es
- Conexiones cliente-servidor no encriptadas
- Servidores escuchando en todas las interfaces
- El usuario es anónimo, sólo se conoce la IP
- Un **firewall** nos ayuda a proteger nuestra LAN y a gestionar el acceso desde el exterior a los servicios 'abiertos al público'
- Pero con netfilter de Linux también se pueden hacer más cosas...

El firewall en la LAN



Filtrado de Paquetes

- Se trabaja con los paquetes IP directamente
- El filtrado de paquetes consiste en examinar la información del paquete y tomar una decisión de que haremos con él
- Podemos dejar que siga su camino, simplemente descartarlo, u otras cosas...
- Es la base de un firewall
  - controla el tipo de tráfico entre redes
  - asegura el acceso a equipos y servicios
  - fuente de información para el admin de la red

NAT

- *Network Address Translation*
- Es el proceso de cambiar la dirección IP (de origen o de destino) de los paquetes IP al pasar de una red a otra
- Se usa generalmente en *intranets*, dónde se usa un direccionamiento privado (con IPs reservadas)

2. Netfilter

- Funcionalidades de netfilter:
  - filtrado de paquetes, sin estados
  - filtrado de paquetes, con estados
  - cualquier tipo de traducción de direcciones IP y puertos (**NAT** y NAPT)
  - estructura fléxible y extensible por el usuario
  - varios niveles de API's para programación
  - muchos módulos y plug-ins mantenidos

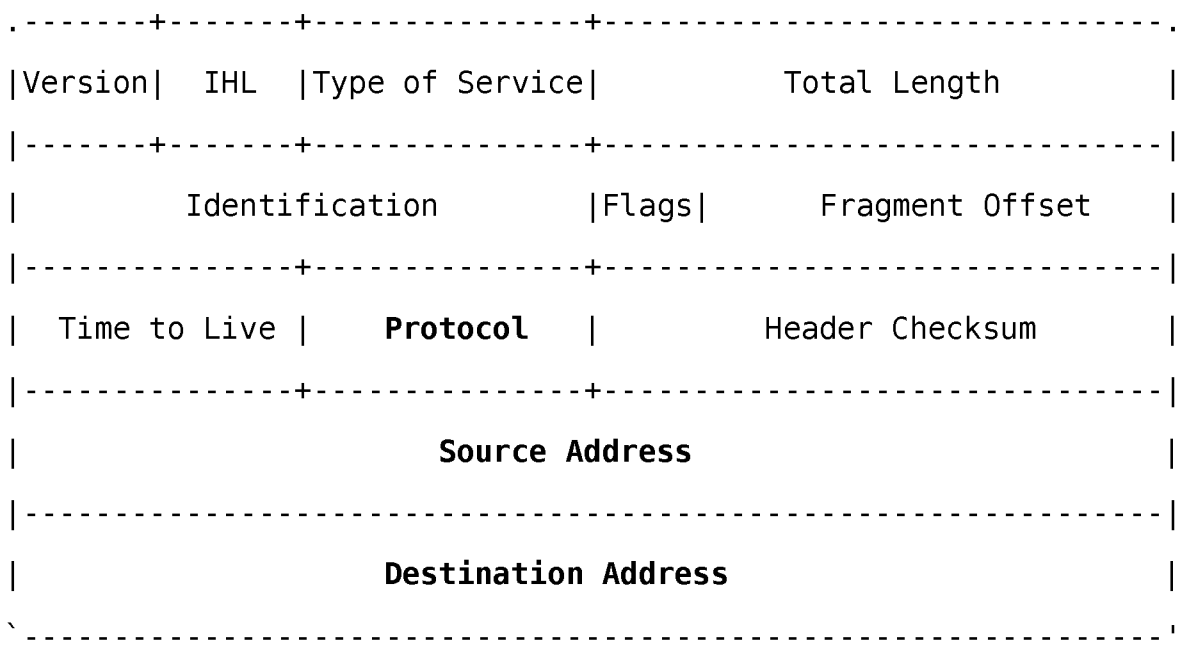
IPtables

- Para decidir que debe hacer con cada paquete, netfilter usa unas reglas definidas en una serie de tablas
- **IPtables** es el comando que el administrador usa para modificar esas tablas de filtrado y en consecuencia configurar netfilter
- En kernels 2.0 se usaba ipfwadm
- En kernels 2.2 se usaba ipchains
- En kernels 2.4+ se usa iptables/netfilter
- Ojo, son parecidos pero no funcionan igual

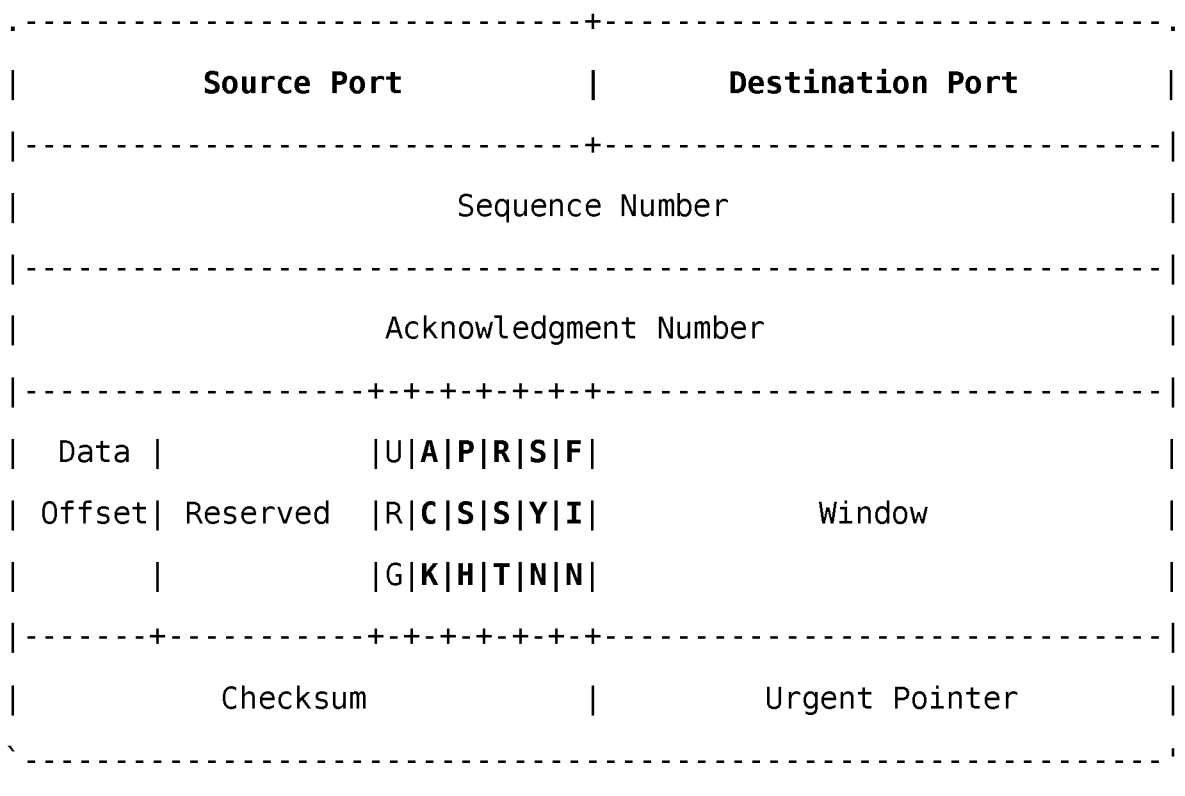
3. Filtrado de paquetes

- **Netfilter** es el software diseñado para hacer el filtrado de paquetes (y NAT) en Linux
- El filtrado de paquetes, al igual que el *routing*, se realiza desde el propio kernel
- Parte de netfilter se incluye ya en los kernels recientes, versiones 2.4 en adelante.
- netfilter examina paquete a paquete todo el tráfico que pasa por el equipo

3.1 Cabecera IP



Cabecera TCP

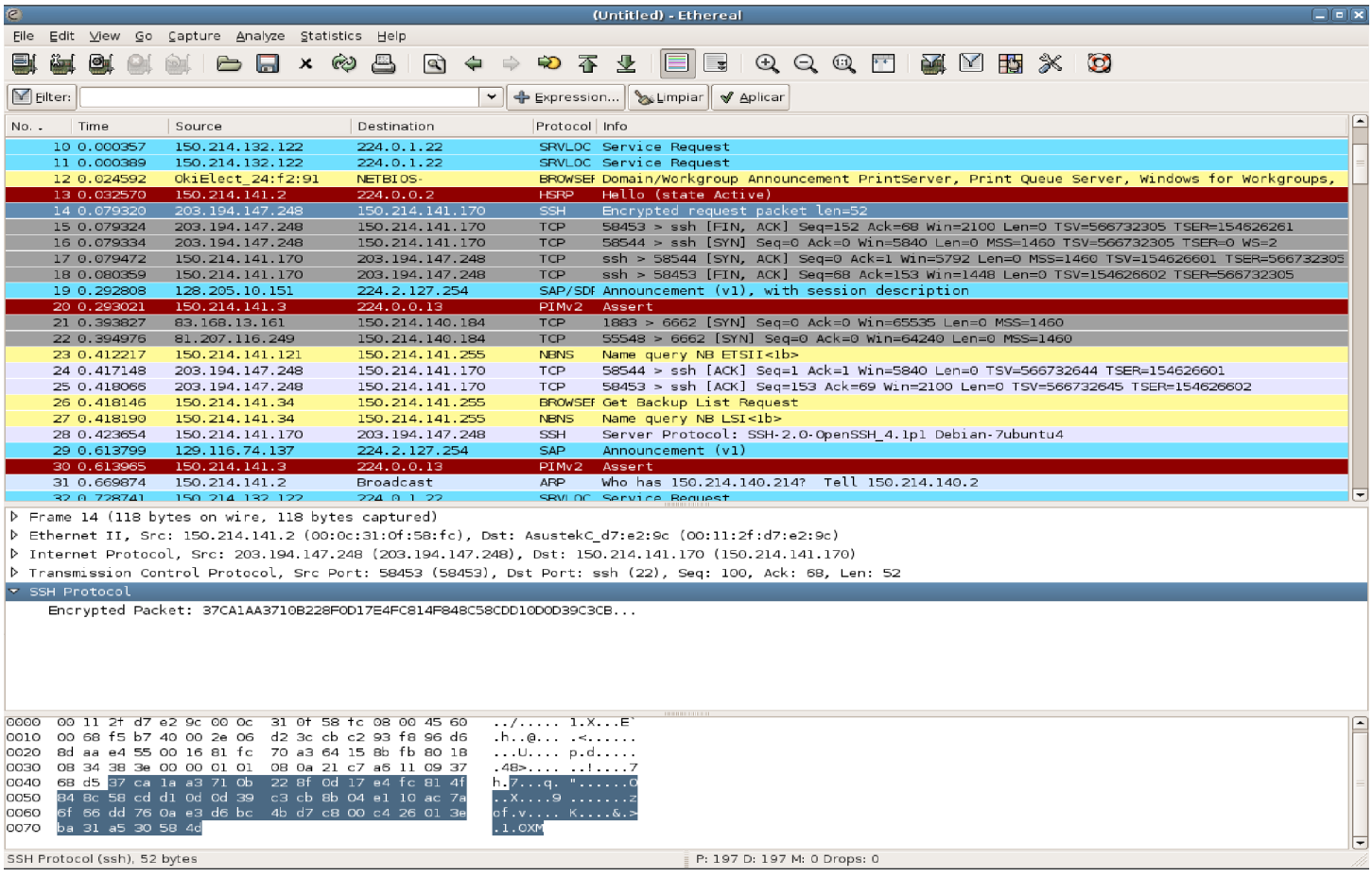


tcpdump

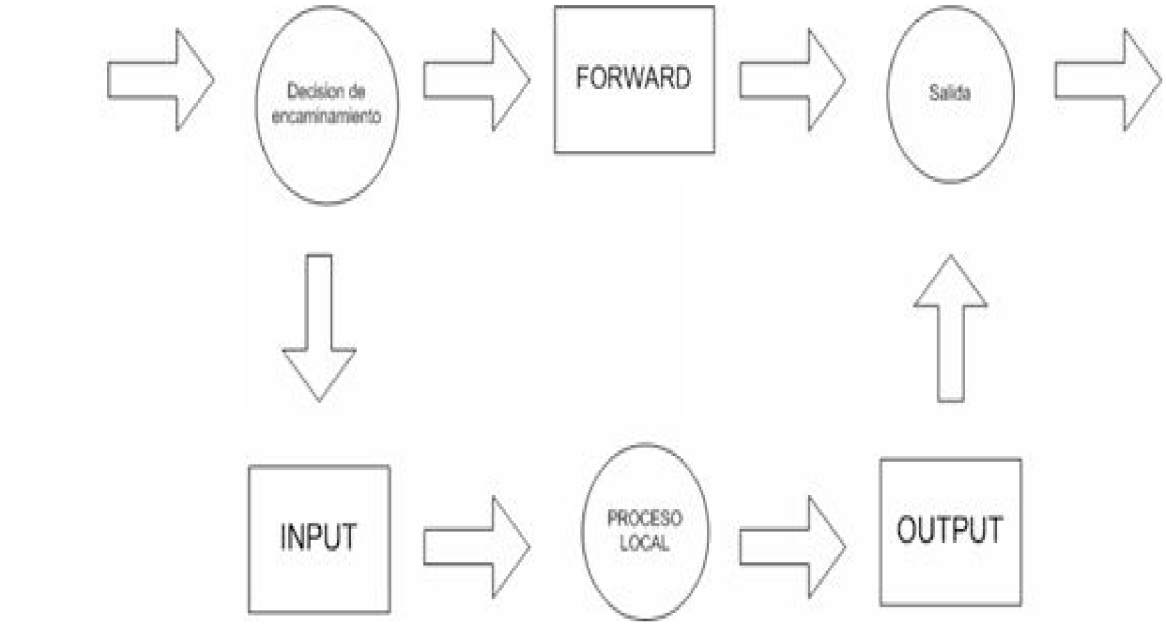
- La herramienta **tcpdump** nos permite poner un interfaz en *modo promiscuo* y mostrar 'en tiempo real' información de los paquetes que llegan y salen por esa interfaz.
- Formato: tcpdump [-n] [-i int] _reglas_, con _reglas_=_regla_ [and|or _reglas_] y _regla_= tcp, udp, icmp, dst host IP, src host IP, host IP, dst net IP/m, src net IP/m, net IP/m, dst port P, src port P, port P, portrange P1-P2...
- Cada regla puede llevar un “not” delante.

  # tcpdump -n -i eth0 dst port 80 and
  not src host 150.214.141.170

wireshark



3.2 Routing



El paso de los paquetes

- Si llega un paquete se mira destino: *routing*
- Si el destino es el propio equipo, se pasa a la *chain* de INPUT, y si continua su camino, llegará al proceso local que corresponda
- Si el destino es alcanzable por otra interfaz del equipo y el *forwarding* está activo, se pasa a la chain de FORWARD, si es aceptada saldrá por la interfaz correspondiente.
- Los paquetes generados localmente pasan directamente al chain OUTPUT y, si lo superan, finalmente al interfaz de salida.

3.3 Construyendo filtros con IPtables

- Contamos con 3 chains iniciales (INPUT, FORWARD y OUTPUT) y podemos añadir más
- Cada chain tendrá una serie de reglas (*rules*) y una política por defecto (*policy*)
- Los paquetes al entrar en una *chain* se irán evaluando por las reglas y si alguna se cumple se tomará la acción (*target*) que diga la regla.
- Si ninguna regla se aplica, se usará la política por defecto

Iptables: *targets* frecuentes

- **ACCEPT**: acepta el paquete, sale del *chain*
- **DROP**: elimina el paquete, sale del *chain*
- **LOG**: guarda un log, pero continua evaluando
- **REJECT**: rechaza el paquete, enviando un 'mensaje de error' al origen, sale del *chain*
- <*chain*>: se puede especificar otro *chain* y se evaluarán entonces las reglas de ese.

Iptables: manejando chains

- Comandos para operaciones con chains:

-P: fija política por defecto en la chain

-L: lista de chains y reglas

-F: elimina todas las reglas de una chain

-Z: reset a cero de contadores

-N: nueva *chain de usuario*

-X: elimina *chain de usuario* vacía

- Ejemplos:

iptables -P FORWARD DROP

iptables -F OUTPUT

Iptables: manejando reglas

- Comandos para manejar reglas de una chain:

-A: añade una regla al final de la chain

-I: inserta una regla en una posición determinada

-D: borra una regla, por posición o especificándola

-R: reemplaza una regla en una posición por otra

- Ejemplos:

iptables -A INPUT <regla> (añade al final)

iptables -I INPUT 2 <regla> (añade en 2ª pos)

iptables -D OUTPUT 3 (borra la 3ª regla)

Iptables: definiendo reglas

- La regla define las condiciones que debe cumplir un paquete y la acción a tomar. Se forman con los siguientes parámetros:

-p [!] protocolo: tcp, udp, icmp o all

-s [!] IP[/máscara]: dirección/red origen

-d [!] IP[/máscara]: dirección/red destino

-j target: marca la acción/target a ejecutar

-i [!] interfaz: interfaz de entrada del paquete

-o [!] interfaz: interfaz de salida del paquete

-m extensión: usa una extensión para la regla

Iptables: un primer ejemplo

- Para probar, queremos filtrar en nuestro equipo el tráfico ICMP (ping) a nuestra IP eth.
- La chain implicada es la de INPUT y la acción será DROP (o REJECT)
- El protocolo es ICMP. El origen cualquiera. El destino 10.1.15.120. El interfaz de entrada será “eth0”, si sólo hay ese.
- El comando quedaría:

```
# iptables -A INPUT -p icmp -d 10.1.15.120 -i eth0 -j DROP
```

Iptables: un primer ejemplo

```
# iptables -L -n (muestra IPs, no resuelve DNS)
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP       icmp --  0.0.0.0/0            10.1.15.120

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

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

- Comprobemos si funciona.
- ¿Desde nuestro equipo responde el ping?
- ¿Y desde otro equipo? ¿Y por otra interfaz?
- ¿Cómo filtramos el icmp completamente?

Iptables: segundo ejemplo

- Queremos filtrar ahora el ping al localhost (sólo se llega desde nuestro propio equipo). La regla sería la siguiente...

 # iptables -D INPUT 1

 # iptables -A INPUT -p icmp -s 127.0.0.1 -i lo -j DROP

- Comprobemos si filtra, ping al 127.0.0.1
- ¿Por qué funciona? ¿Los procesos locales no iban directos al OUTPUT? ¿Qué estoy filtrando?

Iptables: segundo ejemplo

```
    # iptables -F
    # tcpdump -n -i lo icmp
20:06:05.805823 IP 127.0.0.1 > 127.0.0.1: ICMP echo request, id 49698, seq 1, length 64
20:06:05.805878 IP 127.0.0.1 > 127.0.0.1: ICMP echo reply, id 49698, seq 1, length 64
```

- Quitemos el firewall y veamos que está pasando por la interfaz de “loopback”.
- **tcpdump** nos muestra el tráfico del loopback (-i lo), en este caso sólo el tráfico icmp
- Haciendo un ping vemos que realmente hay una salida (el “echo request”) y una entrada (el “echo reply”)
- Si filtramos en INPUT, eliminamos la respuesta! ¿Y en OUTPUT, como sería?

Iptables: consideraciones

- Lo más seguro es cerrar todo (policy DROP) e ir abriendo lo imprescindible, probando servicio a servicio
- Si es un firewall dedicado, en el gateway por ejemplo, mucho cuidado sobre todo con lo que se deja entrar en la red
- Si hay multiples interfaces, comprobar que el tráfico llega/sale por el interfaz correcto
- ¿Realmente necesito el FORWARD? Recuerda activarlo
- Script del firewall para arranque automático

3.4 Extensiones

- Hay multitud de extensiones mantenidas, se cargan de forma automática los módulos.
- Extensión TCP (-p tcp):

  --**sport** [!] <rango_puertos>: puerto/s de origen

  --**dport** [!] <rango_puertos>: puerto/s de destino

  --**tcp-flags** [!] <lista_flags> <flags>: comprueba que de la <lista_flags> están activos <flags>

  [!] --**syn**: una abreviatura de “--tcp-flags SYN,RST,ACK SYN”, paquetes de inicio de conexión del protocolo TCP.

  Ejemplo: -p tcp --syn --dport 20:25 -s 10.1.15.120

  [inicios de conexión desde 10.1.15.120 a tcp 20 al 25]

Extensiones

- Extensión UDP (-p udp):

  --**sport** [!] <rango_puertos>: puerto/s de origen

  --**dport** [!] <rango_puertos>: puerto/s de destino
- Extensión ICMP (-p icmp):

  --**icmp-type** [!] <tipo>: selecciona el tipo de icmp

  Ver # iptables -p icmp –help
- Extensión MAC (-m mac):

  --**mac-source** [!] <eth_addr>: comprueba la MAC address de origen (en INPUT)

Extensiones

- Extensión STATE (-m state)

  [!] --**state** <estado>: comprueba el estado de la conexión al que se refiere el paquete. Puede ser:

  NEW: creando una nueva conexión

  ESTABLISHED: paquete que pertenece a una conexión ya abierta

  RELATED: relacionado con otra conexión ya establecida, pero no es la misma conexión (ej: conexión de datos en FTP o respuesta al PING)

  INVALID: paquetes no identificados

  Ejemplo: # iptables -A FORWARD -i ppp+ -m state ! --state NEW -j DROP

Extensiones

- Extensión LIMIT (-m limit)

  --**limit** <numero>[/<tiempo>]: máx. nº de coincidencias por segundo, minuto, hora o día.

  Se usa para no guardan demasiados LOG o para evitar ciertos tipos de ataques o escaneos de puertos.

  - Ejemplo:

  # iptables -A FORWARD -j LOG -m limit --limit 10/m

  - Ejemplo:

  # iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT

4. NAT con netfilter

- Hay dos tipos fundamentales:
  - SNAT: Source NAT, cambiamos 'de dónde viene' el paquete. Se realiza después del *routing*. Masquerading es un tipo específico de SNAT.
  - DNAT: Destination NAT, cambiamos 'adonde va' el paquete. Se hace siempre antes del routing. Este tipo incluye el NAPT (forward de puertos), balanceo de carga y proxy transparente.
- Tabla aparte en IPtables, “iptables -t nat” y dos chains: PREROUTING, POSTROUTING.
- Necesita que el kernel haga FORWARD:

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

Reglas para hacer NAT

- Es imprescindible el '-t nat'
- Los parámetros fundamentales siguen siendo válidos:

  -p, -s, -d, --sport, --dport, etc...
- Nuevas chains

  -A PREROUTING, -A POSTROUTING
- Nuevos targets

  -j SNAT, -j DNAT, -j MASQUERADE

Haciendo SNAT

- El SNAT ocurre en el POSTROUTING, se evalúa justo antes de la salida del paquete, así que el routing y el resto de chains verán el paquete antes del cambio.

  -j SNAT [-o <int>] --to-source <nueva-IP>

- Ejemplo, NAT para una intranet, a través de un gateway conectado a internet:

```
# iptables -t nat -A POSTROUTING -s 10.1.15.120/22
  -o eth0 -j SNAT --to-source 150.214.141.120
```

Haciendo Masquerading

- Masquerade es un tipo específico de SNAT, dónde no hace falta especificar la <nueva-IP>, sino que usa la del interfaz por el que sale. Se usa con módems (direccionamiento dinámico), nunca con direcciones IP estáticas.

  -j MASQUERADE -o <int>

- Ejemplo, conectar la intranet a internet por un módem instalado en el Linux:

```
# iptables -t nat -A POSTROUTING -s 192.168.10.0/24
  -o ppp0 -j MASQUERADE
```

Haciendo DNAT

- El DNAT ocurre en el PREROUTING, se evalúa justo a la entrada del paquete, así que el routing y el resto de chains verán la dirección de destino ya cambiada.

  -j DNAT [-i <int>] --to-destination <nueva-IP[:puerto]>

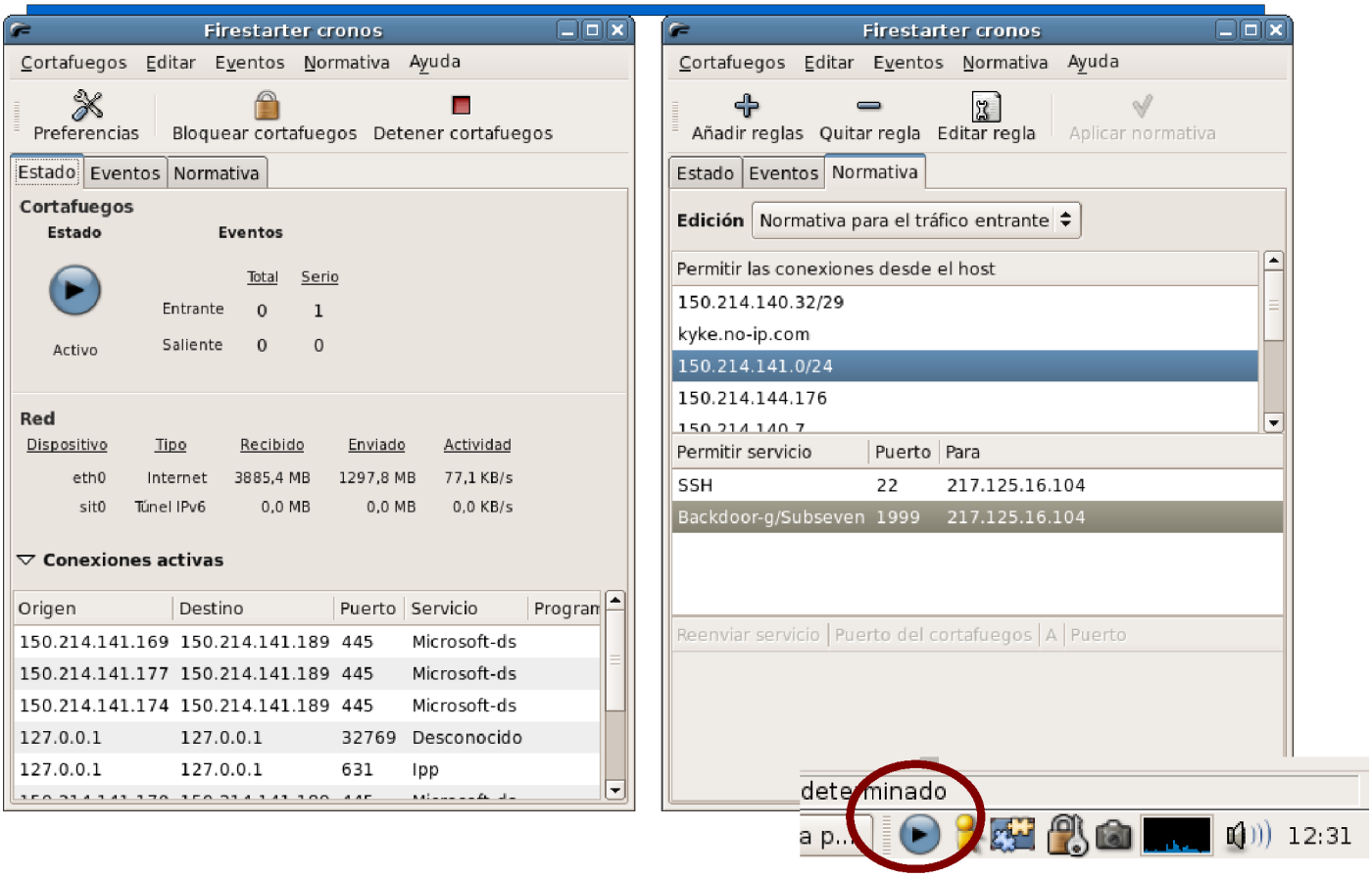
- Ejemplo, DNAT para acceder desde el exterior a un servidor web colocado en un equipo de la intranet en el puerto 8080:

```
# iptables -t nat -A PREROUTING -i ppp0 -p tcp
  --dport 80 -j DNAT --to-destination
  10.1.15.99:8080
```

5. Otras opciones: firestarter

- Firestarter es un software de firewall sencillo pero potente. Posee una interfaz gráfica y permite las restricciones más habituales, a nivel de IPs y puertos, para tráficos entrante y saliente.
- Todo esto lo implementa con netfilter.
- Resulta interesante en servidores pequeños.
- Además proporciona otros servicios: un asistente, la “conexión compartida a internet” (NAT), hace de servidor DHCP y funciones de priorización del tráfico (TdS)

firestarter



UFW (Uncomplicated FireWall)

- La herramienta *standard* de ubuntu. Es una solución intermedia, a partir de una descripción de las redes y de los servicios y restricciones que queremos implementar en el equipo, genera una serie de reglas para el netfilter del kernel.
- Se habilita y configura desde línea de comandos (comando “ufw”) y los cambios se guardan automáticamente entre sesiones... esto se procesa y se generan las reglas apropiadas de netfilter.
- Es muy potente y flexible.

UFW: ejemplos de uso

- # ufw enable|disable
- # ufw status [verbose] [numbered]
- # ufw allow|deny *<port#>[/proto]*

 # ufw allow port 80/tcp

 # ufw deny port 2050/udp

- # ufw allow|deny from <ip> to <ip> port <port#> proto <tcp|udp|icmp>

 # ufw deny from 10.1.12.0/22 to any port 22

 # ufw allow from 192.168.10.0/24 port 22

- # ufw delete <rule_string>
- # ufw delete <rule_number#>

GUFW (GUI de UFW)