Software Libre y Educación:

servicios de red, gestores de contenidos y seguridad

Redes TCP/IP

José Ángel Bernal, Fernando Gordillo, Hugo Santander y Paco Villegas

15 de febrero de 2005

Índice general

Capítulo 1

Introducción al Curso

> "Para Dan Halbert el camino hacia Tycho comenzó en la universidad, cuando Lissa Lenz le pidió prestado su ordenador. El suyo se había estropeado, y a menos que pudiese usar otro suspendería el proyecto de fin de trimestre. No había nadie a quien se atrevería a pedírselo, excepto Dan.
>
> Esto puso a Dan en un dilema. Tenía que ayudarla, pero si le prestaba su ordenador ella podría leer sus libros. Dejando de lado el riesgo de ir a la cárcel durante muchos años por dejar a otra persona leer sus libros, la simple idea le sorprendió al principio. Como todo el mundo, había aprendido desde la escuela que compartir libros era malo, algo que sólo un pirata haría."
>
> *El Derecho a Leer. Richard Stallman - GNU*

Este curso surge como una continuación lógica de los cursos sobre GNULinux convocados en años anteriores por la SAEM Thales y el CICA. En las encuestas realizadas al finalizar los mismos, siempre se ha puesto de manifiesto el interés de muchos de los participantes por ampliar los conocimientos adquiridos. El curso se dirige, por tanto, a aquellas personas que partiendo de unas nociones previas sobre Linux desean seguir profundizando en su conocimiento. Está enfocado sobre todo al profesorado y con él se pretende dar a conocer las herramientas necesarias para poner en funcionamiento un servidor de red centralizado (usando software libre), que permita compartir los recursos y la información en los centros educativos.

Antes de comenzar de lleno con los contenidos del curso debemos aclarar que las distintas entregas se enfocan bajo el uso de dos distribuciones[1]: Guadalinex 2004 (Debian) y Fedora Core 3, pudiendo los participantes optar por aquella que más les interese.

1.1. apt

> ¿Desea instalar o eliminar una aplicación? No hay problema. ¿Desea actualizar un programa que ya ha instalado? Muy fácil. Con un par de simples comandos o pulsando algunos botones, este proceso lo podrá realizar usted mismo. *Red Hat Linux 7.0: The Official Red Hat Linux Getting Started Guide*

Para garantizarnos que trabajamos siempre con la última versión disponible para los programas objeto de estudio, realizaremos la instalación de los paquetes bajo el supuesto de que estamos conectados a Internet. Si bien esto es "casi obligatorio" con Guadalinex, no lo es con Fedora ya que casi todos los paquetes comentados están en alguno de los CDs[2] que componen la distribución.

Aunque desde Fedora podemos usar la herramienta `yum` para instalar paquetes desde Internet, con el fin de homegeneizar el proceso de instalación, lo realizaremos usando los comandos en modo texto del paquete `apt`.

[1] Partimos de la base de que se dispone de un ordenador con alguna de estas distribuciones ya instalada.
[2] O en el DVD

Con los comandos del paquete `apt` instalaremos las últimas versiones de los programas, ya que siempre buscará la versión más reciente de los paquetes.

APT son las siglas de *Advanced Packaging Tool*, es decir, *herramienta avanzada de empaquetamiento*. El sistema APT es un sistema abierto, basado en la licencia GNU y desarrollado por el *APT Development Team*. Este paquete se instala por defecto en Debian (Guadalinex) y no así en Fedora. Así que antes de nada, si usamos Fedora o Red Hat debemos instalarlo.

Desde la página

`http://apt.freshrpms.net/`

podemos acceder a la última versión del programa para Fedora (o el resto de versiones de Red Hat). En la actualidad, y si trabajamos con Fedora Core 3, se trata de bajar:

```
$wget http://ftp.freshrpms.net/pub/freshrpms/fedora/linux/3/apt/apt-0.5.15cnc6-1.1.fc3.fr.i386.rpm
#rpm -ivh apt-0.5.15cnc6-1.1.fc3.i386.rpm
```

1.1.1. /etc/sources.list

Parte fundamental del funcionamiento de `apt` es el archivo en que están localizados los repositorios de paquetes. Este archivo es:

`/etc/apt/sources.list`

Su formato es similar en ambas distribuciones aunque no su contenido. En general las líneas de este fichero son del tipo:

Debian: `deb http://protocolo.site.org/debian distribución sección1 sección2 sección3`

Fedora: `rpm http://protocolo.site.org/ distribución sección1 sección2 sección3`

y para los comentarios se usa el carácter `#`.

GuadaLinex

El contenido de ese fichero en GuadaLinex es:

```
# Junta de Andalucía (Repositorio raiz)
# Método HTTP

deb http://http.guadalinex.org/debian sarge main contrib non-free
deb http://http.guadalinex.org/debian-non-US sarge/non-US main contrib non-
    free
deb http://http.guadalinex.org/debian-security sarge/updates main contrib
    non-free
deb http://http.guadalinex.org/repositorio muflon guada

# Fuentes
# deb-src http://http.guadalinex.org/debian sarge main contrib non-free
# deb-src http://http.guadalinex.org/repositorio muflon guada

# Método FTP
#deb ftp://ftp.guadalinex.org/repositorio muflon main contrib non-free guada
#deb ftp://ftp.guadalinex.org/repositorio muflon/non-US main contrib non-
    free

# Mirror Oficial de Guadalinex: Centro Informático Científico de Andalucía (
    CICA)
#deb ftp://ftp.cica.es/debian sarge main contrib non-free
#deb ftp://ftp.cica.es/guadalinex/repositorio muflon guada

# Mirror Oficial de Debian
# Sarge
#deb http://ftp.fi.debian.org/debian sarge main contrib non-free
```

```
#deb http://ftp.fi.debian.org/debian-security sarge/updates main contrib non
    -free
#deb http://non-us.debian.org/debian-non-US sarge/non-US main contrib non-
    free
```

Listado 1.1: Debian:/etc/apt/sources.list

Todas las líneas de este fichero están comentadas (están encabezadas por el símbolo "`#`") salvo dos. Las líneas comentadas encabezadas con `deb-scr` sólo se deben descomentar en el caso de que deseemos bajarnos los paquetes fuente.

Si bien los repositorios de Guadalinex están casi siempre al día, es preferible trabajar con las últimas versiones de los programas. Para conseguirlo debemos descomentar las líneas de los repositorios oficiales de Debian, se trata de descomentar las tres últimas líneas:

```
deb http://ftp.fi.debian.org/debian sarge main contrib non-free

deb http://ftp.fi.debian.org/debian-security sarge/updates main contrib non-
    free

deb http://non-us.debian.org/debian-non-US sarge/non-US main contrib non-
    free
```

Fedora

Con Fedora en general no tendremos que modificar nada, y con la configuración por defecto es suficiente para iniciar el curso.

En ambos

Si realizamos algún cambio en este fichero **siempre** deberemos ejecutar el comando:

```
# apt-get update
```

Los paquetes descargados son almacenados en el directorio `/var/cache/apt/archives` por si los necesitamos en algún otro momento o deseamos instalarlos en otro ordenador: ya no tenemos por qué descargarlos de nuevo.

1.1.2. apt-get

Para conocer más sobre el comando `apt` se puede consultar el documento *Apt HOWTO*: `http://www.debian.org/doc/manuals/apt-howto/index.es.html`

A partir de ahora usaremos sólo el comando `apt-get` (siempre que sea posible) y partiremos de la idea de que la lista de paquetes está siempre actualizada, es decir, que se ha ejecutado:[3]

```
# apt-get update
Get:1 http://ayo.freshrpms.net fedora/linux/3/i386 release [1990B]
Fetched 1990B in 1s (1146B/s)
Hit http://ayo.freshrpms.net fedora/linux/3/i386/core pkglist
Hit http://ayo.freshrpms.net fedora/linux/3/i386/core release
Get:1 http://ayo.freshrpms.net fedora/linux/3/i386/updates pk-
glist [287kB]
Hit http://ayo.freshrpms.net fedora/linux/3/i386/updates release
Get:2 http://ayo.freshrpms.net fedora/linux/3/i386/freshrpms pk-
glist [160kB]
Hit http://ayo.freshrpms.net fedora/linux/3/i386/freshrpms release
```

[3]En Debian los repositorios de paquetes son otros y no se corresponden con las líneas que se listan.

```
Fetched 447kB in 21s (21,2kB/s)
Leyendo listas de paquetes... Done
Construyendo árbol de dependencias... Done
```

Para instalar un paquete escribiremos:

```
# apt-get install paquete
```

y en el caso de que surja algún problema de dependencias

```
# apt-get -f install
```

Si se daña un paquete instalado o deseamos reinstalar una nueva versión disponible del mismo, deberemos añadir la opción `--reinstall`

```
# apt-get --reinstall install paquete
```

1.1.3. Desinstalando paquetes

Para eliminar un paquete del sistema escribiremos:
`#apt-get remove nombre_paquete`
o equivalentemente
`#apt-get install nombre_paquete-`
Como ya hemos comentado, los paquetes se bajan a `/var/cache/apt/archives`, si deseamos volver a instalarlos ya los tendremos a mano. Ejecutando `apt-get` como en el ejemplo eliminaremos los paquetes, pero no así sus archivos de configuración, si es que existían. Para una eliminación completa del paquete deberíamos ejecutar:

```
# apt-get --purge remove paquete
```

1.1.4. Eliminando archivos de paquete no utilizados

Los paquetes que se instalan en nuestro sistema se bajan previamente a un repositorio de paquetes desde el que son instalados automáticamente por APT. Con el paso del tiempo, este proceso hace que el repositorio empiece a crecer y vaya ocupando mucho espacio en nuestro disco duro.

Para borrar los paquetes después de haber actualizado por completo nuestro sistema podemos ejecutar:

```
# apt-get clean
```

De esta forma se elimina la totalidad de paquetes de la caché.

Pero si no tenemos problemas de espacio, mejor optar por:

```
# apt-get autoclean
```

De este modo, sólo se eliminan de `/var/cache/apt/archives` los paquetes "inútiles", es decir, los que ya no sirven porque existe una nueva versión de los mismos.

➥ Para practicar

Si está usando un servidor proxy, primero se debe dar valor a la variable de entorno `http_proxy`. Si trabajamos en modo consola (se puede definir en modo gráfico desde Gnome o KDE) y deseamos definir la variable de entorno `http_proxy`, podemos usar:

```
export http_proxy="http://ip_proxy:puerto"
```

Por ejemplo

```
export http_proxy="http://192.168.0.1:3128"
```

Para ver que todo está bien:

```
lynx http://www.iesmurgi.org
```

1. Actualicemos el sistema, para eso, con conexión a internet ejecutamos[4]

   ```
   #apt-get update
   #apt-get upgrade
   ```

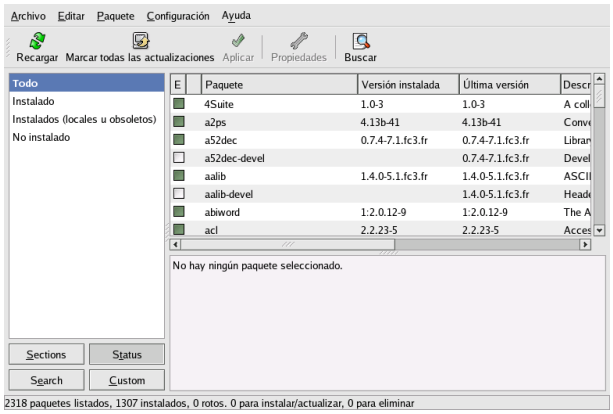
2. Veamos un par de pinceladas más sobre su uso con Fedora ya que si se viene del mundo Debian su manejo no debe presentar ningún problema.

   a) Instalación de **synaptic**: se trata de una herramienta gráfica para gestionar los paquetes instalados en nuestra máquina, disponible tanto para Guadalinex (se instala por defecto) como para Fedora. Para instalarla en Fedora sólo hay que escribir:

   ```
   # apt-get install synaptic
   Leyendo listas de paquetes... Done
   Construyendo árbol de dependencias... Done
   Se instalarán los paquetes NUEVOS siguientes:
     synaptic
   0 upgraded, 1 newly installed, 0 removed and 86 not upgraded.
   Need to get 1497kB of archives.
   After unpacking 5321kB of additional disk space will be used.
   ```

   Una vez instalado, para ejecutarlo desde un xterm, escribimos

   ```
   # synaptic &
   ```



   El manejo de este *front-end* gráfico para mantener el sistema de paquetes es inmediato.

   b) Como ejercicio: instalar el programa[5] `mc` usando el comando `apt-get`. ■

[4] ¡Cuidado: si no tenemos claro qué hacemos, mejor dejarlo como está!
Podemos usar el parámetro `dist-upgrade` en lugar de `upgrade`, la diferencia entre ambos es que con `upgrade` se actualizará el sistema pero no se instalará un paquete nuevo, ni se eliminará uno ya instalado, ni se actualizará un paquete que presente conflictos con otro ya instalado. Sin embargo, si usamos `dist-upgrade` realizamos una actualización completa, es decir, una vez determinado el mejor conjunto de paquetes para actualizar el sistema lo máximo posible, se instalan, actualizan y eliminan todos los que sean necesarios.

[5] En Fedora 3 no se instala por defecto

Capítulo 2

Redes de Ordenadores

> El concepto de trabajo en redes es probablemente tan antiguo como lo es el de las telecomunicaciones. Imagínese por un momento, gente viviendo en la Edad de Piedra, en donde los individuos usen tambores para transmitirse mensajes. Supóngase que un hombre de las cavernas A quiere invitar a otro hombre B a una partida de choques de piedra. Lamentablemente viven tan distantes, que a B le sería imposible escuchar el tambor de A cuando éste lo llame. ¿Qué puede hacer A para remediar esto? Él podría 1) ir caminando al sitio de B, 2) conseguir un tambor más grande, ó 3) pedirle a C, quien vive a mitad de camino, que reenvíe el mensaje. La tercera elección es denominada Trabajo en Redes. (*Guía de Administración de Redes Segunda Edición*, OLAF KIRCH)

La cita anterior nos expresa claramente que la necesidad de comunicación y, más aún, de la comunicación a distancia, ha existido entre los seres humanos desde la noche de los tiempos y ante esta necesidad, se han ido planteado diferentes alternativas "tecnológicas".

Más moderna es la necesidad de comunicar ordenadores, que en un principio fueron islas no conectadas entre sí. Rápidamente, se hizo necesario desarrollar sistemas que permitieran la comunicación entre diferentes ordenadores y la correcta transferencia de datos entre ellos, surgiendo de esta forma el concepto de "redes de ordenadores" y de "trabajo en red"[1].

La parte que viene a continuación puede "pecar" de teórica, pero preferimos presentarla por motivos de completitud y referencia. No es necesario que comprendáis completamente todos los conceptos y si alguno se cansa, puede pasar al capítulo 4.

2.1. Modelo de Referencia OSI

A través de una red se pueden ejecutar procesos en otro ordenador, acceder a sus ficheros, enviar mensajes, compartir programas, etc. Esta comunicación de datos se realiza mediante el envío de unidades de información, lógicamente agrupadas, denominadas *paquetes de datos* .

Los paquetes de datos incluyen la información que intercambian las aplicaciones, junto con otros elementos necesarios para hacer que la comunicación sea factible y confiable en la relación con los dispositivos de destino, como por ejemplo, las direcciones de origen y destino. La dirección de origen de un paquete especifica la identidad del ordenador que envía el paquete. La dirección de destino especifica la identidad del ordenador que recibe el paquete. Esta identificación es necesaria, de la misma forma que lo es la dirección del destinatario en una carta postal.

Para que los paquetes de datos puedan viajar desde el origen hasta su destino a través de una red, es importante que todos los dispositivos de la red hablen el mismo lenguaje, lo que denominamos protocolo. Un **protocolo** es una descripción formal de un conjunto de normas y convenciones

[1] Hasta el punto de que hoy día, con una conexión a la red, es como podemos obtener el máximo aprovechamiento de un ordenador. Sin ella, un ordenador es como un naúfrago en una isla.

que determinan el formato y la transmisión de los datos entre los diferentes dispositivos de una red. Podemos entenderlo como la gramática que rige una lengua, aunque los ordenadores son menos "inteligentes" que nosotros y se bloquearían con nuestras faltas de ortografía. Por ello, esas reglas son más estrictas en la comunicación entre ordenadores.

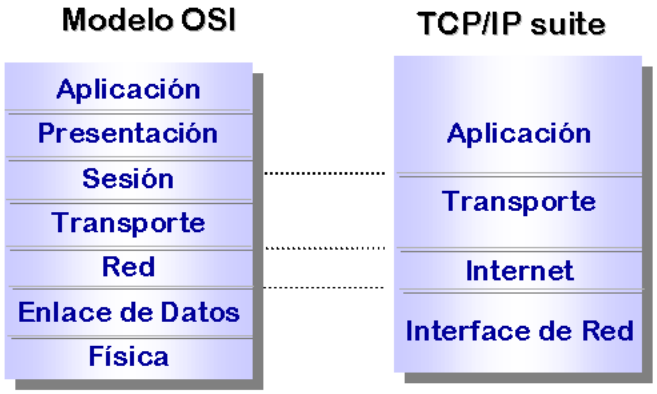
A mediados de los años 70, diversos fabricantes desarrollaron sus propios sistemas de redes locales y protocolos. En 1980, la empresa Xerox, en cooperación con DEC e Intel, desarrolló las especificaciones del primer sistema de red, denominado EtherNet. En 1982 aparecen los ordenadores personales, y en 1986, IBM introdujo la red TokenRing.

El principal inconveniente de los primeros sistemas de comunicación en red fue que cada uno de ellos era propiedad de una empresa, siendo desarrollados con hardware y software propietarios, con elementos protegidos y cerrados, que usaban protocolos y arquitecturas diferentes. Como consecuencia de ello, la comunicación entre ordenadores pertenecientes a distintas redes era muy difícil, por no decir imposible.

Cuando las empresas intentaron comunicar sus redes, cada una con su implementación particular, se dieron cuenta de que necesitaban salir de los sistemas de redes propietarios, optando por una arquitectura de red con un modelo común que hiciera posible interconectar distintas redes sin problemas.

Para solucionar este problema, la Organización Internacional para la Normalización (ISO[2]) reconoció que era necesario crear un modelo común que pudiera ayudar a desarrollar redes que pudieran comunicarse y trabajar conjuntamente[3]. Como consecuencia de ello, elaboraron el modelo de referencia OSI en 1984.

El modelo OSI (*Open Systems Interconnection*) define la forma en que se comunican los sistemas abiertos[4] de telecomunicaciones. El modelo de referencia está compuesto de 7 capas y la forma en que deben comunicarse entre ellas. Estas capas se presentan normalmente como una pila, que se conoce como la Pila de Protocolos OSI (*OSI Protocol Stack*).



Los niveles o capas del modelo OSI son desde el inferior hasta el superior: **Nivel Físico, Nivel de Enlace de Datos, Nivel de Red, Nivel de Transporte, Nivel de Sesión, Nivel de Presentación y Nivel de Aplicación**.

Aunque nuestro objetivo final sea el estudio de las redes TCP/IP, base de la Internet actual, por sus bondades didácticas, comentaremos antes el modelo OSI. Conoceremos el funcionamiento de los modelos de capas y lo trasladaremos posteriormente al estudio de TCP/IP.

↬ El concepto que subyace en el Modelo de Referencia OSI, es similar a cuando escribimos una carta[5]. Veamos la figura que viene a continuacion, que nos presenta dos ordenadores (equipo

[2] *International Standards Organization*

[3] denominado *interoperabilidad*.

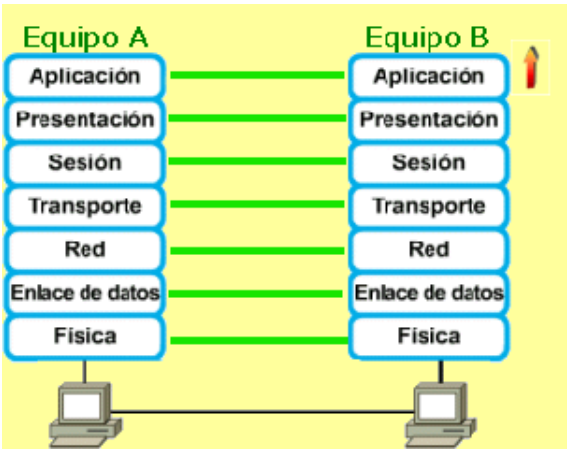
[4] Los SISTEMAS ABIERTOS son aquellos que se basan en especificaciones estándares, definidas por organismos internacionales independientes.

[5] La analogía solamente pretende que comprendamos cómo funciona la comunicación entre las capas del modelo OSI.

A y equipo B) e imaginémonos que usted (usuario del equipo A en Almería) desea inscribirse en los cursos que organiza la Sociedad Andaluza de Educación Matemática Thales y el CICA. Para ello, debe mandar una carta de solicitud a la Secretaría de Thales que se encuentra en Cádiz. Primero, vamos a descender por la pila de protocolos del Equipo A de la izquierda, desde la capa de Aplicación hasta la capa Física. Mediante el cable que une ambos equipos, pasamos al equipo B de la derecha y ascendemos por la pila de protocolos de este equipo, desde su capa Física hasta llegar a la capa de Aplicación.

Realizando una analogía entre ambos modelos, podemos imaginar que en la capa de Aplicación del Equipo A, decidimos el mensaje que queremos enviar: la petición de un curso de Thales. En la capa de Presentación lo plasmamos sobre el papel. Siguiendo el símil, en la capa de Sesión metemos el papel en un sobre con la dirección del destinatario[6]. Lo bajamos al buzón de Correos más cercano, que constituiría la capa de Transporte. Dentro de Correos, no sabemos qué ruta seguirá la carta[7], constituyendo una auténtica capa de Red: la Red de Correos. La capa de Enlace sería la Oficina de Correos que dirige la carta hacia el tren, que llegará "físicamente" a la localidad de destino.

Ya ha llegado nuestra carta a la estación de ferrocarril de Cádiz, con lo que ha llegado al nivel Físico del Equipo B. Se envía a la Oficina de Correos correspondiente a la zona de destino, pasando por los niveles de Enlace y de Red. El cartero, mirando la dirección de entrega, lo deposita en el buzón de la Secretaría de Thales, cumpliendo con el nivel de Transporte. Allí, el personal de la Secretaría recoge el sobre y lo abre, finalizando la capa de Sesión. La solicitud es leída como correspondería al nivel de Presentación y en la capa de Aplicación se procede a inscribir al alumno en el curso solicitado.



Podemos observar que la comunicación final "virtual" se ha producido entre el alumno y la Secretaría de Thales, solicitando nuestra inscripción en los cursos. Sin embargo, la comunicación "real" ha ido viajando desde nosotros, hasta el buzón de correos, los distintos medios de transporte, la oficina de correos de destino, el cartero y la Secretaría de Thales.

Volvamos al Modelo de Referencia OSI. En éste, el protocolo de cada capa sólo se interesa por la información que le corresponde a su capa, y no por la información que necesitan o procesan las demás capas.

↬ Por ejemplo: El e-mail es un protocolo de aplicación que se comunica sólo con otras aplicaciones que hablan el mismo protocolo (SMTP, POP, IMAP). Por lo tanto, la aplicación de e-mail no se interesa de si la capa física es una red ethernet, una línea ADSL o un módem.

[6] Ponemos también el remite, por si nos han de enviar una carta de confirmación de la inscripción.

[7] Puede que en algunos casos, para ir de Almería a Cádiz, pase por Barcelona, pero a nosotros nos dará igual, excepto en el tiempo que tarde.

La información se pasa a las capas de abajo hasta que la información llega a la red. En el nodo remoto, la información es entonces pasada a las capas superiores hasta que llega a la aplicación correspondiente.

Pasamos a detallar las capas del modelo OSI, desde las capas del nivel superior a las inferiores[8]. Aunque las capas TCP/IP, como veremos más adelante, no son exactamente iguales, tomaremos como ejemplo el funcionamiento de un cliente y un servidor web.

2.1.1. Capa de Aplicación

La capa de aplicación permite al usuario acceder al servicio final deseado. Proporciona las interfaces de usuario y soporte a los servicios como correo electrónico, transferencia de archivos o bases de datos.

↬ En el caso de la comunicación entre un cliente y un servidor web, sería el protocolo HTTP que hace que el cliente solicite páginas al servidor y éste se las sirva.

2.1.2. Capa de Presentación

Este nivel elimina los problemas que puedan surgir al comunicar distintas arquitecturas, pues cada arquitectura estructura los datos de una forma específica[9], que no tienen por qué ser compatibles. Se traducen los datos a un formato común, independiente de la arquitectura de máquina.

En esta capa se define el formato de los datos que se van a intercambiar entre las aplicaciones y se ofrece a los programas de aplicación un conjunto de servicios de transformación de datos. En caso de ser necesario, también se encarga de la compresión y del cifrado

↬ Por ejemplo, si el cliente web se ejecuta en una máquina de arquitectura Intel y el servidor en un PowerPC, esta capa utilizaría un formato común de intercambio de la información, que luego cada arquitectura adapta al suyo en concreto.

2.1.3. Capa de Sesión

Se encarga de organizar y sincronizar el diálogo entre los dos extremos implicados. Ofrece mecanismos para gestionar el diálogo entre los sistemas como: quién debe emitir en cada instante, agrupamiento de datos en unidades lógicas, recuperación si se produce algún problema en la comunicación.

↬ Dentro de una sesión, el cliente y el servidor web se intercambian múltiples peticiones. Por ejemplo, dentro de una página web se pueden incorporar múltiples imágenes que el cliente tiene que solicitar al servidor. También, siguiendo los hiperenlaces, pedimos distintos contenidos del servidor. Esta conexión estable entre el cliente y el servidor se mantiene gracias a la capa de sesión. Si se produce un corte momentáneo en la línea de comunicaciones entre nuestro cliente web y el servidor web, esta capa se encarga de mantener la comunicación hasta que se restablezca, en el caso de que no sea demasiado tiempo, o la da por perdida definitivamente.

2.1.4. Capa de Transporte

Es la responsable del envío desde el origen al destino (es decir, de extremo a extremo) del mensaje completo.

La capa de red[10] supervisa el envío extremo a extremo de los paquetes de forma individual, pero no reconoce ninguna relación entre esos paquetes, tratando a cada uno de forma independiente. Sin embargo, la capa de transporte asegura que el mensaje completo llegue desde el origen al destino, intacto y en el orden correcto, supervisando el control de flujo y control de errores desde un extremo a otro de la comunicación. La capa de transporte asegura un servicio fiable.

[8] No os asustéis si hay términos que no entendéis o no habéis escuchado en la vida. Ah, y no es necesario aprendérselo de memoria. Lo importante es que captéis el modelo.

[9] Por ejemplo, en unas arquitecturas el octeto de información más significativo se sitúa primero y en otras, es justo al contrario.

[10] que se encuentra bajo la de transporte.

Si el fichero que el servidor web tiene que transmitirle al cliente es de un tamaño grande, tiene que partirse en trozos más pequeños durante el tránsito. La capa de transporte se ocupa de partirlo en trozos en el origen y de recomponer los trozos en el destino.

↬ Es como si un convoy de 200 vehículos debe atravesar un río y solamente existen barcas con capacidad para 50 vehículos. Deberán partirse en "trozos" de 50 vehículos para caber en las barcas y en la otra orilla, volver a recomponer el convoy.

2.1.5. Capa de Red

Es la responsable del envío desde la estación origen a la estación destino de los paquetes, es decir, se asegura de que cada paquete llegue desde su punto inicial hasta su punto final.

↬ Siguiendo el ejemplo anterior, sería el barquero, cuyo objetivo es pasar cada barca a la otra orilla, pero no tiene responsabilidad sobre el reagrupamiento del convoy una vez en la otra orilla.

Si dos sistemas están conectados en el mismo enlace, no existe la necesidad de la capa de red (por ejemplo, dentro de una misma LAN). Sin embargo, si dos sistemas están en diferentes redes físicas, será necesaria una capa de red para dirigir (o "enrutar") la entrega desde la red física de origen a la red física de destino del paquete.

Entre las responsabilidades de la capa de red se incluyen:

- *Direccionamiento lógico*: El direccionamiento físico implementado en la capa de enlace de datos[11] manipula el problema del direccionamiento localmente. Pero si un paquete pasa de la frontera de la red, se necesita otro sistema de direccionamiento para ayudar a distinguir los sistemas origen y destino. La capa de red añade un encabezamiento al paquete que llega de la capa superior, que entre otras cosas, incluye la dirección lógica del origen y del destino.

- *Enrutamiento*: Cuando redes independientes son conectadas para crear una inter-red (e.g. una red de redes, como Internet), los dispositivos (llamados routers o gateways) enrutan o encaminan los paquetes a su destino final.

↬ Si nuestro cliente web se encuentra en la red interna de nuestro centro y el servidor al que nos conectamos está en el Instituto Tecnológico de Massachusetts (MIT), la capa de red sabra cómo llegar hasta allí, pasando por los múltiples caminos de Internet.

2.1.6. Capa de Enlace de Datos

La capa de enlaces de datos ensambla los bits de la capa física en grupos de tramas (protocolos de red) y asegura su correcto envío.

También es la encargada de la verificación y corrección de errores de la capa física. En caso de que ocurra un error en los bits, se encarga de avisar al transmisor de que efectúe una retransmisión, y por lo tanto, la capa de enlace se encarga también del control de flujo físico de los datos.

La capa de enlace de datos se divide en dos subcapas:

1. LLC (*Logical Link Control*): define cómo se transfieren los datos sobre el cable y provee servicios de enlace de datos a las capas superiores.

2. MAC (*Medium Access Control*): define quién puede usar la red cuando múltiples dispositivos están intentando acceder simultáneamente (e.g. token passing, Ethernet CSMA/CD,..).

↬ Por ejemplo, comunica nuestro sistema linux a la red del centro y controla cuándo podemos emitir en un medio que es compartido entre todos los ordenadores del aula.

[11] Que se encuentra bajo la de red.

2.1.7. Capa Física

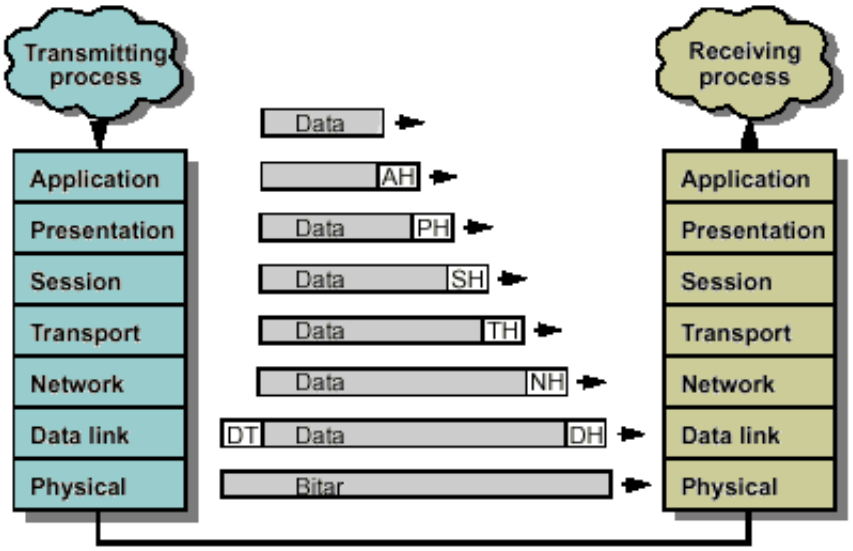
Se ocupa de la transmisión de bits a través de un canal de comunicación físico.

Regula aspectos de la comunicación tales como el tipo de señal (analógica, digital,..), el esquema de codificación, sincronización de los bits, tipo de modulación, tipo de enlace (punto a punto, punto-multipunto), el modo de comunicación (dúplex, half-dúplex o símplex), tasa de bits (número de bits por segundo), topología empleada, y, en general, todas las cuestiones eléctricas, mecánicas, señalización y de procedimiento en la interfaz física (cables, conectores,...) entre los dispositivos que se comunican.

Lo dicho, cables y señales eléctricas, ópticas o de cualquier otro tipo[12].

2.2. Comunicación entre capas

Tras ver las distintas capas del Modelo OSI[13], corresponde estudiar la forma en la que se lleva a cabo una comunicación en el Modelo de capas.



Supongamos que el proceso emisor (en la parte izquierda de la figura) tiene información que enviar, para ello entregará los datos[14] a la capa de Aplicación. Esta capa añade a la información una cabecera (AH[15]) que permite procesar el protocolo que tenga definido. El conjunto formado por los datos originales y la cabecera de aplicación es entregado a la capa de presentación.

Esta capa transforma el bloque recibido en función del servicio pedido, y añade también su nueva cabecera (PH[16]). Este nuevo conjunto de datos es entregado a la capa inmediatamente inferior: la capa de sesión.

Cada nivel de la torre OSI añade una cabecera a los datos a transmitir, a excepción del nivel 1 que no añade nada, y del nivel 2 que además añade una cola (DT[17]). Dichas cabeceras son datos de control para el nivel equivalente[18] en el otro extremo de la comunicación.

Es importante destacar que el conjunto de datos que pasa de la capa N a la capa (N -1) puede ser fragmentado en bloques más pequeños. En este caso, cada bloque llevará su propia cabecera y

[12] Señales cerebrales dentro de poco tiempo.
[13] Conocido coloquialmente como el modelo OSI de la ISO.
[14] En la figura, Data. Perdonad si incluimos la figura original en inglés, pero parecía adecuada y no hubiera salido tan bonita.
[15] Proviene de *Application Header*
[16] *Presentation Header*
[17] De *Data Link Trailer* (Cola de Enlace de Datos).
[18] El que se encuentra a la misma altura.

además, la capa que realiza la fragmentación deberá ser la encargada (en la máquina receptora) de recomponer los bloques hasta formar de nuevo el conjunto original de datos y entregarlos a la capa superior, así hasta llegar al proceso receptor.

↬ Aunque la idea puede parecer rebuscada, es un proceso parecido a la comunicación entre personas. Inicialmente tenemos una idea que deseamos comunicar a nuestro interlocutor. Esta idea es entregada a la zona del cerebro encargada del lenguaje, que generará los impulsos nerviosos necesarios para hacer vibrar nuestras cuerdas vocales y emitir unidades pequeñas (fonemas), produciéndose un sonido que será captado por el oído de nuestro interlocutor. Los impulsos nerviosos generados por el oído de nuestro interlocutor serán enviados a su cerebro, que reconstruye los fonemas, los agrupará en palabras, y de ellas se extraerá el significado de la información. Todo este proceso se puede abstraer y resumir como que la información, el mensaje, que deseábamos comunicar ha llegado a la otra persona.

2.2.1. Tipos de Servicios

Según las capacidades que aportan, para el modelo OSI existen varias clasificaciones de los servicios que puede proporcionar una red. Una clasificación permite dividirlos en *servicios orientados a la conexión* y en *servicios sin conexión*[19], y otra, los divide en *servicios confiables* y *servicios no confiables*.

1. *Servicios orientados a la conexión*: En ellos la conexión es como un tubo a través del cual se envía la información de forma continuada, por lo que los mensajes llegan en el orden en que fueron enviados y sin errores. Proporcionan un servicio confiable de comunicación de datos. Cada paquete se procesa en su secuencia correcta: después del anterior y antes del posterior. No pueden faltar paquetes, ni procesarse uno sin haber procesado los anteriores.

   ↬ Una analogía es el sistema telefónico. Si la comunicación se corta, hay que restablecerla de nuevo.

2. *Servicios no orientados a la conexión*: En los que cada mensaje lleva la dirección completa de su destino. La información no se garantiza que llegue de forma continua. La ruta que sigue cada mensaje es independiente. El servicio no es entonces confiable, pues no se garantiza el orden de llegada de los paquetes, ni se controla su flujo, porque para estos servicios no es necesario. Por ello, los paquetes deben llevar sus direcciones completas de destino.

   ↬ Una analogía sería el caso del sistema de correo convencional. Puede que una carta que habíamos enviado hace cinco días, llegue antes que una que enviamos hace dos semanas al mismo destinatario. Podemos procesar un paquete (leer la carta), aún cuando el anterior no haya llegado aún, o incluso puede que nunca lo haga.

Otra clasificación de los servicios es la que distingue entre confiables y no confiables:

- *Servicios confiables*: son aquellos en los que la transmisión de datos está controlada en cada momento, pudiéndose determinar el correcto envío y recepción de todos los datos transmitidos. Para ello la máquina receptora envía mensajes de acuse de recibo de las tramas recibidas a la máquina emisora.

  ↬ Es como si todas las cartas que mandáramos en correo postal lo hiciéramos con acuse de recibo. Es más lento y más caro, pero más seguro.

- *Servicios no confiables*: en éstos no existe un control de los datos transmitidos, por lo que no se puede garantizar que se hayan recibido todos los datos.

  ↬ Una analogía sería el caso de un naúfrago en una isla. Manda muchos mensajes dentro de botellas y se conformaría con que uno al menos sea encontrado por alguien.

[19] o no orientados a la conexión.

Capítulo 3

Conjunto de Protocolos TCP/IP.

> En vez de ser un ideal académico, el modelo de referencia TCP/IP lo elaboraron los fabricantes y los programadores que, por fin, llegaron a un acuerdo sobre las comunicaciones en Internet. Además, este modelo es más sencillo que el modelo ISO, porque desarrolla TCP/IP desde el punto de vista del programador, partiendo de que el mundo es real y práctico. (*Firewall Linux*, Robert L. Zieger)

TCP/IP ha llegado a convertirse en un estándar de facto para la comunicación de redes de ordenadores. De hecho, es la familia de protocolos[1] empleada por Internet.

TCP equivale a las siglas de *Transmission Control Protocol* e IP corresponde a las siglas de *Internet Protocol*. Estos protocolos se crearon y normalizaron mucho antes de que se definiera el Modelo de Referencia OSI de la ISO. Ya a finales de los 80, muchas empresas, administraciones y organismos usaban TCP/IP, cuando todavía OSI no estaba totalmente desarrollada. Aun así, el modelo OSI es una buena arquitectura de organización de protocolos, y es muy didáctico. Esa ha sido la razón por la que lo hemos visto anteriormente.

No existe un modelo oficial de protocolos TCP/IP, al contrario que en OSI. Los protocolos se han ido definiendo de una forma un tanto anárquica, y a posteriori han sido englobados en capas. La operatividad (que el sistema funcionara) primaba sobre el academicismo.

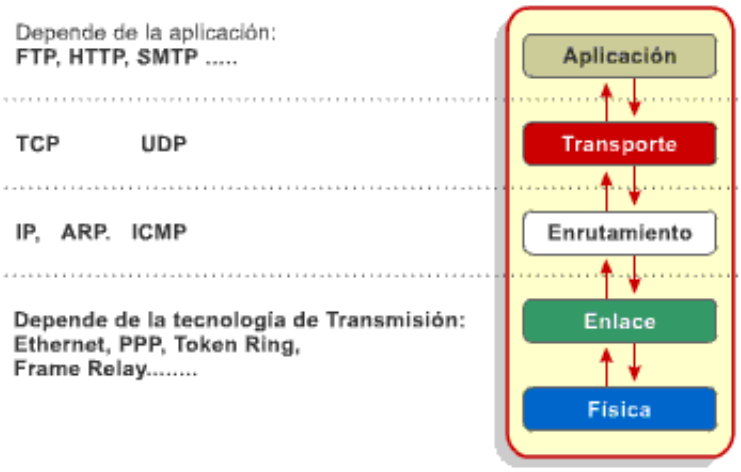
3.1. Niveles de la Arquitectura TCP/IP

En la imagen se pueden apreciar los niveles de la arquitectura TCP/IP junto a ejemplos de protocolos pertenecientes a cada una de las capas. La arquitectura TCP/IP podemos encontrarla, según las fuentes que consultemos, dividida en cuatro o en cinco niveles. Éstos son: Nivel Físico, Nivel de Enlace[2], Nivel de Red[3], Nivel de Transporte y Nivel de Aplicación.

[1] De todos los protocolos que la componen, dos de ellos, el protocolo TCP y el protocolo IP, dan nombre a toda la familia.

[2] Algunos autores unifican los niveles de Enlace y Físico en uno solo.

[3] o de Enrutamiento



La forma en que trabajan las capas de TCP/IP es similar a como hemos visto en el modelo OSI.

3.1.1. Nivel de Aplicación.

Proporciona una comunicación entre procesos o aplicaciones que pueden estar en sistemas distintos[4]. Además de las aplicaciones, este nivel se ocupa de las posibles necesidades de presentación y de sesión.

Los protocolos de aplicación más utilizados son: TELNET (terminal remoto), FTP (transferencia de ficheros), HTTP (el protocolo entre clientes y servidores web) o SMTP (correo electrónico).

3.1.2. Nivel de Transporte.

Proporciona transferencia de datos de extremo a extremo[5], asegurando que los datos llegan en el mismo orden en que han sido enviados, y sin errores. Esta capa puede incluir mecanismos de seguridad.

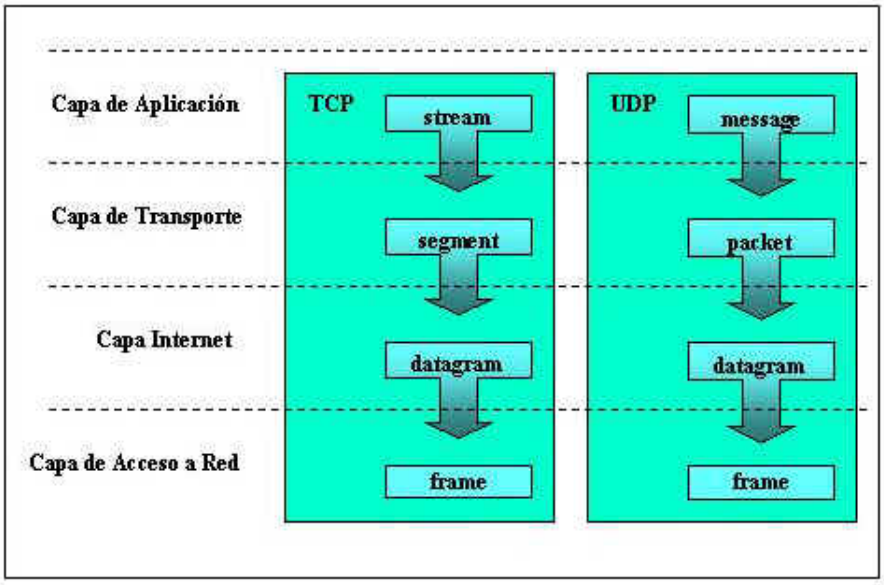
Consta de dos servicios diferenciados. Un servicio consiste en el envío y recepción de datos orientado a conexión (TCP) y el otro (UDP) consiste en el envío y recepción de datos no orientados a conexión.

↬ Para ver mejor la diferencia entre estos dos servicios, podemos poner dos ejemplos. En el primer caso, escuchamos una canción a través de Internet. Si en un momento determinado falta un paquete por congestión en la red, es preferible descartarlo y pasar al siguiente para seguir escuchando la música, a que paremos para recuperar el paquete perdido y no seguir escuchando nada. El servicio adecuado en este caso sería el no orientado a conexión (UDP). En el otro ejemplo, estamos descargando un fichero, por ejemplo una imagen del CD de instalación de Linux, desde un servidor de Internet. En este caso, si se pierde un paquete, debemos volver a pedirlo y esperar a que llegue, porque en caso contrario, todo lo recibido antes y después no nos serviría de nada.

El protocolo TCP (*Transmission Control Protocol*) de la capa de transporte es un servicio orientado a conexión. La unidad de datos que envía o recibe el protocolo TCP es conocida con el nombre de *segmento* TCP.

El protocolo UDP (*User Datagram Protocol*) de la capa de transporte es un servicio no orientado a conexión. La unidad de datos que envía o recibe el protocolo UDP es conocida con el nombre de *paquete* UDP. En la siguiente figura vemos los nombres que reciben las unidades de datos de las distintas capas, según pertenezcan al servicio TCP o al UDP.

[4]Para comunicar procesos en un único ordenador podemos también utilizar TCP/IP y, de hecho, es muy común.
[5]De ordenador origen a ordenador destino, aunque entre medio pase por otros sistemas como routers o gateways.



3.1.3. Nivel de Red.

Se encarga de conectar equipos que están en redes diferentes. Permite que los datos atraviesen distintas redes interconectadas desde un sistema origen hasta un sistema destino.

La capa de red es la responsable de proveer los siguientes servicios a la capa de transporte:

- Establecer el sistema de direccionamiento lógico de la red.
- Enrutamiento de paquetes.

Durante el proceso de enrutamiento se hace uso de un servicio no orientado a conexión para el envío y recepción de paquetes.

Si un paquete que va a ser enrutado excede la máxima unidad de transferencia *Maximum Transfer Unit* (MTU) de un medio físico[6], esta capa fragmenta el paquete con el fin de adaptarse al tamaño máximo de cada medio y el paquete es ensamblado en el sistema destino. Recordemos el ejemplo del convoy que debe fragmentarse para cruzar un río. Por ejemplo, en Ethernet este tamaño es de 1.500 bytes o de 4.450 bytes en un enlace de tecnología ATM.

Esta capa está compuesta por los protocolos: IP, ARP e ICMP principalmente.

- El protocolo IP (*Internet Protocol*) ofrece el servicio de direccionamiento lógico de la red TCP-IP y el de enrutamiento de paquetes.
- El protocolo ARP (*Address Resolution Protocol*) ofrece el servicio de resolución de direcciones IP con su respectiva dirección física.
- El protocolo ICMP (*Internet Control Message Protocol*) ofrece el servicio de informe de errores que pueden ocurrir durante el enrutamiento de paquetes.

La unidad de datos que envía o recibe el protocolo IP se conoce con el nombre de *datagrama* IP.

3.1.4. Nivel de Enlace o de Acceso a la red.

Es el nivel responsable del intercambio de datos entre dos sistemas conectados a una misma red[7]. Controla la interfaz entre un sistema final[8] y una red.

[6] Que puede ser una Ethernet, línea de módem...

[7] Por ejemplo, en una Red de Área Local con tecnología Ethernet.

[8] La terminología para referirse en este ámbito a un ordenador conectado a una red es variada: sistema, host, estación, nodo...

3.1.5. Nivel Físico.

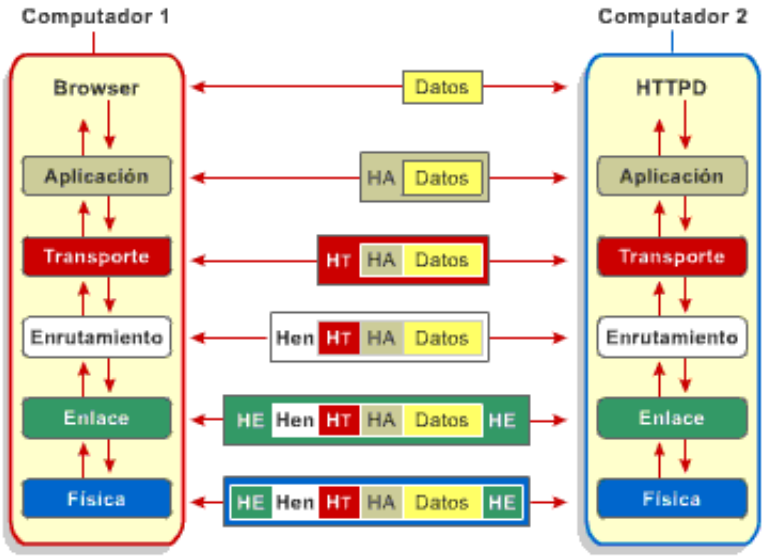
Define las características del medio, su naturaleza, el tipo de señales, la velocidad de transmisión, la codificación, etc.

La unidad de envío o recepción de datos de la capa física se conoce con el nombre de *trama* (frame). Entrarían aquí el cableado, la electrónica de red y parte de la tarjeta de red.

3.2. Añadiendo cabeceras y colas

En la siguiente figura, podemos observar que el envío y recepción de datos entre dos aplicaciones es un proceso de intercambio de datos entre capas del mismo nivel, basado en un modelo cliente-servidor.

La aplicación del computador 1 es una aplicación cliente (Browser o Navegador) que utiliza el protocolo "HTTP" de la capa de aplicación. La aplicación del computador 2 es un servidor de páginas web (proceso httpd, que puede ser Apache), que se intercambian los datos, en un mismo lenguaje. Los niveles correspondientes en cada una de las máquinas se comunican haciendo uso de la información que incorporan en las cabeceras y colas correspondientes.

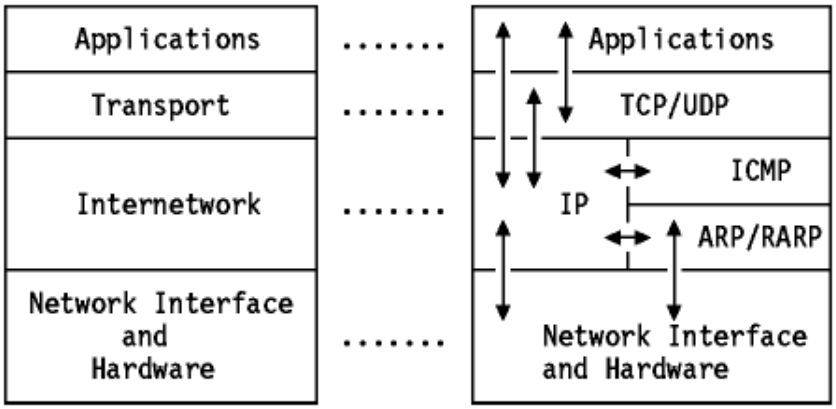


Capítulo 4

TCP/IP: desde los pulsos hasta los datos

> Crear y operar cortafuegos que protejan el perímetro de la red no es física cuántica. De todas formas, hacerlo exige un mejor conocimiento de las redes TCP/IP que el necesario para la mayoría del resto de las funciones de la administración de sistemas y redes. (*Firewalls*, Bill MacCarty)

Pasemos a introducir algunos protocolos importantes de las diferentes capas de TCP/IP, empezando desde el nivel inferior y continuando hasta los de nivel superior. En el nivel inferior, el físico, el protocolo más común en una red de área local es el Ethernet. En el nivel de red, la estrella es el protocolo IP junto a sus ayudantes[1], ARP, RARP e ICMP. En el nivel de transporte, se encuentran TCP y UDP, que dan servicio a los protocolos de aplicación como pueden ser HTTP, FTP o SMTP.



4.1. Nivel Físico

4.1.1. Ethernet e IEEE 802.3

Es el medio más común que nos encontramos en una Red de Área Local (LAN). Conviven dos implementaciones muy parecidas, la Ethernet y la norma de IEEE 802.3, que está basada en la anterior.

[1] IP es el protocolo que transfiere los datos hacia las aplicaciones y ARP, RARP e ICMP le sirven para el control de errores, búsqueda de direcciones...

Ethernet es una especificación LAN de "banda base" inventada por Bob Metcalfe (fundador de 3com) y David Boggs en 1973, mientras trabajaban en Xerox PARC (*Palo Alto Research Center*) que operaba a 10Mbps utilizando un protocolo de acceso múltiple al medio conocido como CSMA/CD (*Carrier Sense Multiple Access/Collision Detect*) sobre un cable coaxial. Ethernet fue creado en Xerox en los años 70, pero el término es usualmente referido para todas las LAN CSMA/CD.

La especificación IEEE 802.3 fue desarrollada en 1980 basada sobre la tecnología original Ethernet. La versión 2.0 de Ethernet fue desarrollada conjuntamente por DEC (*Digital Equipment Corporation*), Intel, y Xerox y es compatible con el estándar IEEE 802.3. El estándar IEEE 802.3 provee una gran variedad de opciones de cableado. Tanto Ethernet como IEEE 802.3 se implementan normalmente en la tarjeta de red.

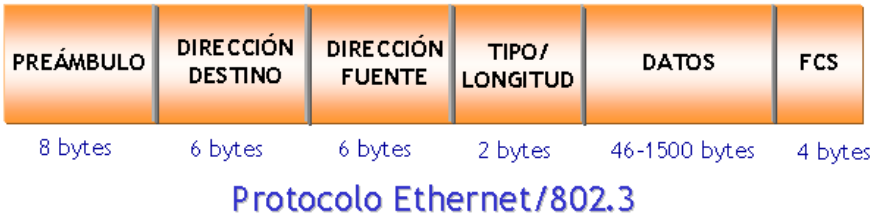
Direcciones MAC

Todos los interfaces (tarjetas) compatibles Ethernet/802.3 poseen una Dirección MAC o **Dirección Hardware** única en el mundo, de 48 bits (o lo que es lo mismo, 6 bytes) de longitud. Cada fabricante de equipos ethernet tiene asignado un rango de direcciones, y es responsabilidad de éste asignar una dirección distinta a cada tarjeta de red. Las direcciones MAC están almacenadas en una pequeña memoria que poseen las tarjetas de red.

Las direcciones MAC se representan en hexadecimal con el siguiente formato: XX:XX:XX:XX:XX:XX[2]. Por ejemplo, 00-40-05-7F-CE-85, sería una dirección MAC válida.

Conviene decir que las direcciones MAC solamente sirven para comunicarse en un mismo medio físico, como por ejemplo, una Red de Área Local (LAN). Cuando hay que comunicarse con un dispositivo que no pertenece a esa LAN, entran en juego el nivel de red y las direcciones lógicas[3].

Estructura de la trama Ethernet



Protocolo Ethernet/802.3

La estructura de la trama Ethernet/802.3 es como sigue:

Preámbulo: (64 bits) El paquete comienza con una secuencia de unos y ceros alternados (de 56 bits en 802.3 ó de 62 bits en Ethernet), que se completa hasta 64 bits en ambos casos. El preámbulo recibido en la red no es pasado por la tarjeta de red hasta el sistema.

Dirección de Destino: (6 bytes) La dirección de destino (DD) es de 48 bits (6 bytes) de tamaño[4], de la cual se transmite primero el bit menos significativo. La DD es utilizada por la tarjeta del sistema receptor, para determinar si el paquete entrante es para él. Si el sistema recepetor detecta una correspondencia entre su dirección MAC y la dirección que viene en el campo DD, recogerá el paquete. Los otros sistemas, a los que no se dirige la trama, ignorarán el resto del paquete.

Las direcciones de destino pueden ser:

1. **Individual** (física): El campo DD contiene una dirección única e individual asignada a un nodo en la red. Es decir, una dirección MAC de una tarjeta de la red.

2. **Broadcast** (difusión): El campo DD está formado todo por unos. Es una dirección especial y todos los dispositivos MAC de la red deberán recibir el mensaje de broadcast.

[2] El separador pueden ser los dos puntos (:) o un guión (-)
[3] Por ejemplo, las direcciones IP.
[4] Es una dirección MAC.

Dirección Fuente: (6 bytes) La dirección fuente (DF) es de 48 bits (6 bytes) de tamaño, transmitiéndose primero el bit menos significativo. El campo DF lo provee la MAC de la tarjeta emisora, la cual inserta su propia dirección física en este campo al transmitirse la trama, indicando que fue la estación origen. Los formatos de direcciones tipo broadcast son ilegales en el campo DF.

Longitud/Tipo: (2 bytes) El campo Longitud (en 802.3) o Tipo (en Ethernet) de 2 bytes va tras el campo DF. La elección de escoger Longitud o Tipo es dependiente de si la trama es 802.3 o Ethernet.

Datos: (46 - 1500 bytes) Este campo contiene los datos (la información útil que es transferida) cuyo tamaño varía de 56 a 1.500 bytes.

FCS (*Frame Check Sequence*): (4 bytes) Contiene el valor del algoritmo CRC (*Cyclic Redundancy Check*) de 32 bits de la trama completa. El CRC es calculado por la estación emisora sobre los campos DD, DF, Longitud/Tipo y Datos y es anexado en los últimos 4 bytes de la trama. El mismo algoritmo CRC es utilizado por la estación receptora para calcular el valor CRC en la trama recibida. El valor calculado por el receptor es comparado con el valor que fue puesto en el campo FCS por la estación emisora, obteniendo un mecanismo de detección de errores en caso de datos corruptos.

Protocolo CSMA/CD.

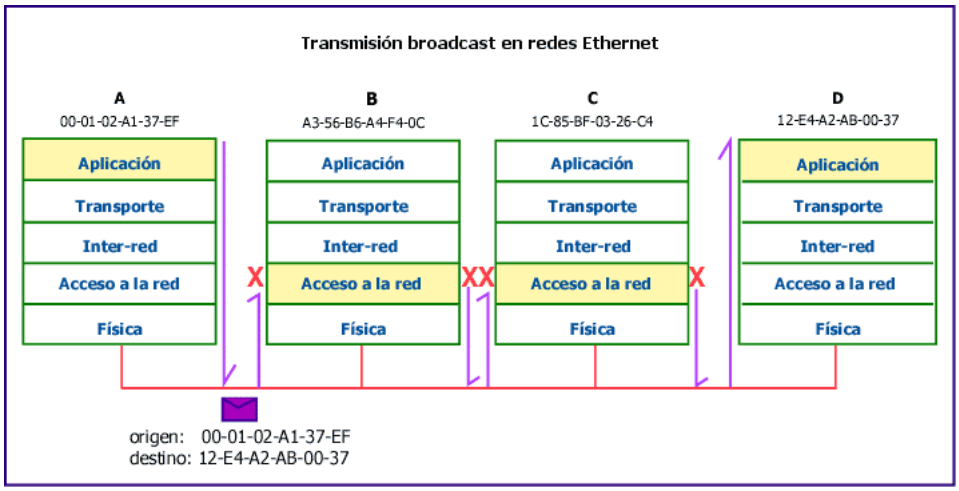
Expliquemos cómo funciona el protocolo de acceso al medio[5] CSMA/CD. Son las siglas de *Carrier Sense, Multiple Access with Collision Detect*[6].

El problema que pretende resolver es cómo acceden a un medio compartido (el cable de red) varios ordenadores. Otro protocolo de acceso al medio es el Paso de Testigo[7], en donde solamente la estación que disponga del token (testigo) puede transmitir en ese momento. Como en una carrera de relevos.

↬ El protocolo CSMA/CD funciona de la siguiente forma. Cuando un ordenador necesita transmitir información, la tarjeta de red (que es donde se implementa el protocolo, como dijimos antes) escucha en el cable, de forma parecida a como los indios del Oeste Americano escuchaban si venía un tren por la vía, pegando la oreja al raíl. Si está pasando información, el medio está ocupado y tiene que esperarse durante un tiempo aleatorio. Si no pasa información en ese momento, puede transmitir. Hasta aquí la parte de Acceso Múltiple con Detección de Portadora. Poner la oreja y ver que no pasa nadie.

Puede ocurrir que dos estaciones hayan hecho lo mismo: escuchar, ver que no emitía nadie, y emitir ellas, produciéndose lo que se conoce como **colisión**. Para resolver el conflicto, lo que hace cada estación es emitir y quedarse escuchando el momento posterior, por si se ha producido una colisión. Si se ha producido una colisión, cada estación da por no emitida la trama y espera un tiempo aleatorio[8] para volver a intentarlo.

[5] Estos protocolos se encargan de arbitrar quién puede acceder al medio físico en un momento determinado.
[6] En castellano: Acceso Múltiple con Detección de Portadora y Detección de Colisiones.
[7] Token. Utilizado en entornos específicos, con requerimientos muy exigentes de tiempo real, como los de producción industrial. Ejemplos: token ring, token bus.
[8] El tiempo de espera debe ser aleatorio para minimizar el que vuelvan a colisionar indefinidamente.



Como se observa en la figura, cuando la estación A ha emitido la trama, las demás estaciones de la red están escuchando si la comunicación va dirigida a ellas. En este caso, la dirección MAC de destino es la de la estación D, que será la única que reciba la trama. El resto de estaciones (B y C en este caso) descartarán la trama al nivel de la tarjeta de red[9].

4.2. Nivel de Red

4.2.1. Direccionamiento IP

Las direcciones TCP/IP (y aplicable a Internet) pueden ser simbólicas o numéricas. La forma simbólica es más fácil de leer y recordar por las personas, por ejemplo: *mileto.cica.es*. La forma numérica es un número binario sin signo de 32 bits, habitualmente expresado en forma de números decimales separados por puntos. Por ejemplo, *150.214.5.11* es una dirección numérica, que se refiere a la misma máquina que la dirección simbólica anterior. Ocurre lo mismo que con el nombre de una persona (Juan Pérez García) y su número de documento nacional de identidad (23.456.789-X). A las personas nos es más fácil recordar el nombre y sin embargo, las máquinas[10] trabajan mejor con el DNI.

La forma numérica es usada por las máquinas porque es más eficiente y es la que utilizan los protocolos de nivel de red. La función de correspondencia entre los dos tipos de direcciones, la simbólica y la numérica, la realiza el sistema de DNS (*Domain Name System*). A partir de ahora[11], nos referiremos a la forma numérica, cuando hablemos de dirección IP.

Las direcciones IP son números de 32 bits, habitualmente expresados como cuatro valores decimales[12] separados por puntos[13]. El formato binario para la dirección IP 128.2.7.9 es: 10000000 00000010 00000111 00001001[14].

Cada interfaz de una máquina conectada a una red TCP/IP tiene asignada una dirección IP. Por interfaz de red entendemos el dispositivo que nos une a la red, como una tarjeta de red o un módem.

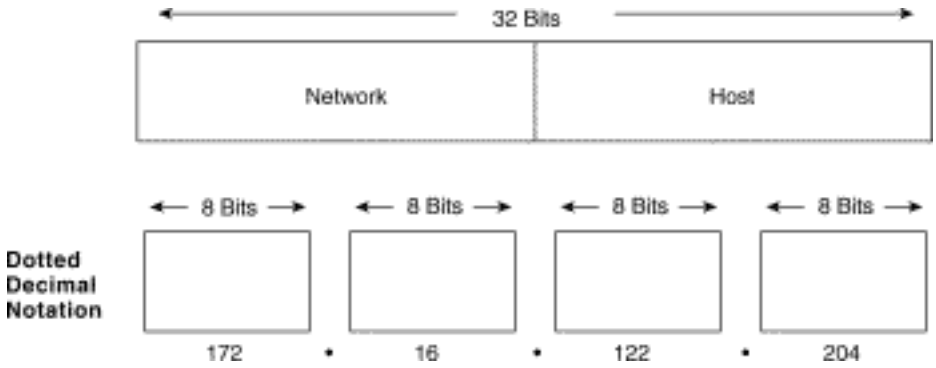
Dos nodos conectados a una misma red no pueden tener la misma dirección IP[15]. Todas las máquinas conectadas a una misma red poseen direciones IP con los primeros bits iguales (bits de

[9] Hay una manera de que la tarjeta funcione en modo promiscuo y escuche las tramas aunque no vayan dirigidas a ella. En este funcionamiento se basan los sniffers o "husmeadores" de la red.
[10] Sobre todo las de la Agencia Tributaria
[11] Hasta que abordemos el estudio del sistema DNS en la siguiente entrega.
[12] Cada número decimal, expresa 8 bits, con un rango del 0 al 255. $2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0$
[13] Conocida como notación decimal separada por puntos (dotted decimal notation).
[14] $(2^7 =)128.(2^1 =)2.(2^2 + 2^1 + 2^0 =)7.(2^3 + 2^0 =)9$
[15] Como si en una calle hubiera dos casas con el mismo número.

red), mientras que los bits restantes[16] son los que identifican a cada máquina concreta dentro de esa red.

↬ Como por ejemplo, el prefijo telefónico identifica la provincia y el resto de números identifica el teléfono concreto dentro de esa provincia.

Las máquinas que pertenecen a una misma red IP, pueden comunicarse directamente unas con otras. Para comunicarse con máquinas de una red IP diferente, deben hacerlo mediante mecanismos de interconexión como routers, proxys o gateways.



A la parte de la dirección IP que es común a todas las direcciones que se encuentran en una red IP, se le llama la parte de la red[17] (network). Los bits restantes son llamados la parte de puesto[18] (o de host).

El tamaño de la parte dedicada al puesto depende del tamaño de la red. Entre estas dos partes deben completar los 32 bits. Para satisfacer diferentes necesidades, se han definido varias clases de redes[19], fijando diferentes sitios donde dividir la dirección IP. Las clases de redes se dividen en las siguientes:

Clase A: Comprende redes desde 1.0.0.0 hasta 127.0.0.0. El número de red está contenido en el primer octeto (byte). Esta clase ofrece una parte para el puesto de 24 bits, permitiendo aproximadamente 1,6 millones de puestos por red.

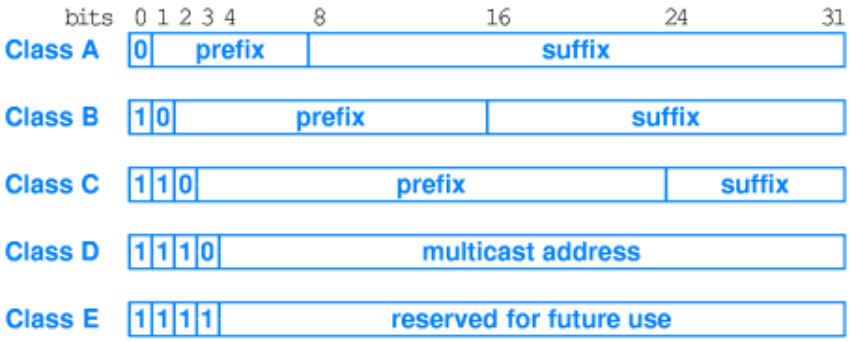
Clase B: Comprende las redes desde 128.0.0.0 hasta 191.255.0.0; el número de red está en los dos primeros octetos. Esta clase permite 16.320 redes con 65.024 puestos cada una.

Clase C: Van desde 192.0.0.0 hasta 223.255.255.0, con el número de red contenido en los tres primeros octetos. Esta clase permite cerca de 2 millones de redes con 254[20] puestos cada una.

Clases D, E, y F: Las direcciones que están en el rango de 224.0.0.0 hasta 254.0.0.0 son experimentales o están reservadas para uso con propósitos especiales. Por ejemplo, IP Multicast, un servicio que permite trasmitir información a muchos puntos en una internet a la vez, tiene direcciones dentro de este rango.

Las redes de clase A se distinguen porque comienzan por 0, las de clase B por 01, las de clase C por 110, las de clase D por 1110 y las de clase E por 1111.

[16] Hasta completar los 32.
[17] Estos bits, se encuentran situados en la parte de más a la izquierda
[18] Los bits situados a la derecha.
[19] Esta clasificación de redes no tiene mucha utilidad actualmente, pero se incluye como reseña histórica y mecanismo de comprensión de los conceptos de *parte de la red* y *parte del puesto*.
[20] Más adelante veremos de dónde sale este número.



Sin embargo, estas clases no deben ya tomarse al pie de la letra, porque lo normal es que se unan o dividan redes según las necesidades de cada organización.

Al número de bits que comparten todas las direcciones de una red se le llama *máscara de red* (netmask), y su papel es determinar qué direcciones pertenecen a la red y cuáles no. Veámoslo con un ejemplo.

	Decimal	Binario
Dirección de puesto	192.168.1.5	11000000.10101000.00000001.00000101
Máscara de red	255.255.255.0	11111111.11111111.11111111.00000000
Parte de red	192.168.1.	11000000.10101000.00000001.
Parte de puesto	.5	.00000101
Dirección de red	192.168.1.0	11000000.10101000.00000001.00000000
Dirección de difusión	192.168.1.255	11000000.10101000.00000001.11111111

Una dirección de puesto IP (192.168.1.5) a la que se aplique una operación "and"[21] de bits con su máscara de red (255.255.255.0[22]), nos dará la dirección de la red a la que pertenece (192.168.1.0). La dirección de red será el menor número de dirección IP dentro del rango de la red porque tiene la parte de puesto toda con ceros.

Una red IP queda definida por la dirección de la red y la máscara de la red. La notación que se usa para referirnos a dicha tupla es DIRECCIÓN DE RED/MÁSCARA DE RED (192.168.1.0/255.255.255.0) o DIRECCIÓN DE RED/NÚMERO DE BITS DE RED (192.168.1.0/24[23]).

En esta red, para la parte de puesto han quedado 8 bits, que nos darán $2^8 = 256$ direcciones posibles. Siempre hay que quitar dos direcciones especiales, que son la dirección de la red (todo a ceros en la parte de host) y la dirección de broadcast (todo a unos en la parte de host). Entonces, quedarán 254[24] direccciones posibles para puestos dentro de la red.

La dirección de difusión (broadcast) es una dirección especial que escuchan todas las máquinas en esa red IP, además de a la suya propia. Esta dirección es a la que se envían los paquetes de datos si se supone que todas las máquinas de la red lo deben recibir.

↬ Por ejemplo, si en un centro se manda una circular para que todos los profesores acudan a una reunión; va dirigida a todos en general y a ninguno en particular.

Ciertos tipos de datos, como la información de encaminamiento y los mensajes de aviso son transmitidos a la dirección de difusión[25] para que cada estación en la red pueda recibirlo simul-

[21] ó "Y lógico"

[22] En este caso, 24 bits para la parte de red y 8 bits para la parte de puesto.

[23] 11111111.11111111.11111111.00000000 tiene 8x3=24 unos

[24] De aquí viene: 256-2=254

[25] Esta es la dirección de difusión de IP. No confundir con la difusión física en Ethernet. Cumplen parecida función, cada una en su nivel correspondiente.

táneamente. Para ello se utiliza la dirección más alta posible en la red, conseguida con la parte de red a la que se añade el resto de bits (de la parte de puesto) todos con valor a uno (1). En el ejemplo anterior sería 192.168.1.255.

Podemos partir las redes en otras más pequeñas[26], es lo que se conoce como *subnetting*. Por ejemplo, tomemos la red de clase C 172.26.1.0/24, en donde estarían incluidas las direcciones desde la 172.26.1.0 hasta la 172.26.1.255, incluyendo la dirección de red y la de difusión. Tendríamos como hosts, desde el 172.26.1.1 hasta el 172.26.1.254. Si hacemos subnetting, utilizando 25 bits para la red en vez de 24, hemos partido esta red en dos. Veamos cuáles son estas redes: la primera sería la red 172.26.1.0/25[27], que iría desde la dirección 172.26.1.0 (dirección de red) hasta la 172.26.1.127 (que seria la dirección de broadcast). Siendo los nodos desde el 172.26.1.1 hasta el 172.26.1.126, en total 126 nodos posibles. La segunda red sería la 172.26.1.128/25, con los datos que aparecen en la tabla.

Red	Broadcast	Primer host	Último host
172.26.1.0/255.255.255.128	172.26.1.127	172.26.1.1	172.26.1.126
172.26.1.128/255.255.255.128	172.26.1.255	172.26.1.129	172.26.1.254

A partir de la dirección IP de destino, un nodo de la red puede determinar si los datos deben ser enviados a través de un router o un gateway hacia el exterior de la red, porque pertenezcan a una red distinta a la suya. Si los bytes correspondientes a la parte de red de la dirección IP de destino, son los mismos que la parte de red de la dirección IP del host origen, los datos no se pasarán al router, porque están en la misma red; si son diferentes, se pasarán a un router, para que los encamine hacia el exterior de la red. En este caso, el router tendrá que determinar el camino de enrutamiento idóneo en base a la dirección de la red de destino de los paquetes y una tabla interna que contiene la información de enrutamiento.

Las direcciones IP son usadas por el protocolo IP para definir únicamente un host en la red. Los datagramas IP (los paquetes de datos elementales intercambiados entre máquinas) se transmiten a través de alguna red física[28] conectada a la interfaz de la máquina y cada uno de ellos contiene la dirección IP de origen y la dirección IP de destino. Para enviar un datagrama a una dirección IP de destino determinada, la dirección de destino debe ser traducida o mapeada a una dirección física. Esto puede requerir transmisiones en la red para encontrar la dirección física de destino (por ejemplo, en una Red de Área Local, el protocolo ARP (*Address Resolution Protocol*), se usa para traducir las direcciones IP a direcciones físicas MAC).

Direcciones de red reservadas.

Desde el punto de vista de su accesibilidad, podemos clasificar las direcciones IP en:

- Direcciones IP públicas: aquellas que son visibles por todos los puestos conectados a Internet. Para que una máquina sea visible desde Internet, debe tener asignada obligatoriamente una dirección IP pública, y no puede haber dos puestos con la misma dirección IP pública.
- Direcciones IP privadas: aquellas que son visibles únicamente por los puestos de su propia red privada. Los puestos con direcciones IP privadas no son visibles desde Internet, por lo que si quieren salir a ésta deben hacerlo a través de un router[29] o un proxy (o intermediario). Las direcciones IP privadas se utilizan en redes de empresas, organismos, centros para interconectar los puestos de trabajo. Se definen en el RFC[30] 1918, que ha convertido en obsoleto al RFC 1597.

[26] O agruparlas, para formar otras más grandes.
[27] La máscara sería 11111111.11111111.11111111.10000000, o 255.255.255.128
[28] Recordemos que una red IP es una red lógica
[29] U otro dispositivo con capacidad de realizar NAT, que básicamente es una traducción de direcciones. La dirección IP privada es traducida a una IP pública en este dispositivo para que pueda salir a Internet.
[30] Los RFC (*Request For Comments*) son documentos técnicos que definen los estándares de Internet.

Si estamos construyendo una red privada y no tenemos intención de conectar nunca esa red a Internet, entonces podríamos elegir las direcciones que queramos, pues no vamos a colisionar con nadie, sin embargo, es buena práctica utilizar direcciones privadas por si en el futuro nos conectáramos. Estas direcciones son:

Dirección	Bits de red	Máscara de red	Número de redes
10.0.0.0	8	10.255.255.255	1 de clase A
172.16.0.0	12	172.31.255.255	16 de clase B
192.168.0.0	16	192.168.255.255	256 de clase C

Estas direcciones no se corresponden con las de ninguna máquina en Internet y no se encaminarán a través de los "routers" de Internet. Podremos utilizarlas de forma interna, con la ventaja de que si conectamos mediante un proveedor a Internet, nuestras direcciones no coincidirán con las de ninguna máquina de Internet.

En este caso, puede que haya multitud de equipos en redes distintas con estas mismas direcciones IP, pero como estas direcciones no se "rutean" no hay por qué coordinar su uso. En el caso de que se conecten a Internet, se habilitan mecanismos como la traducción de direccciones (NAT) o el uso de intermediarios (proxys) para que las direcciones que salgan a Internet sean direcciones válidas.

También la dirección 127.0.0.1, denominada de bucle local (*loopback*), es una dirección especial, que como veremos, utiliza la propia máquina para acceder a sus procesos locales.

4.2.2. Protocolo ARP

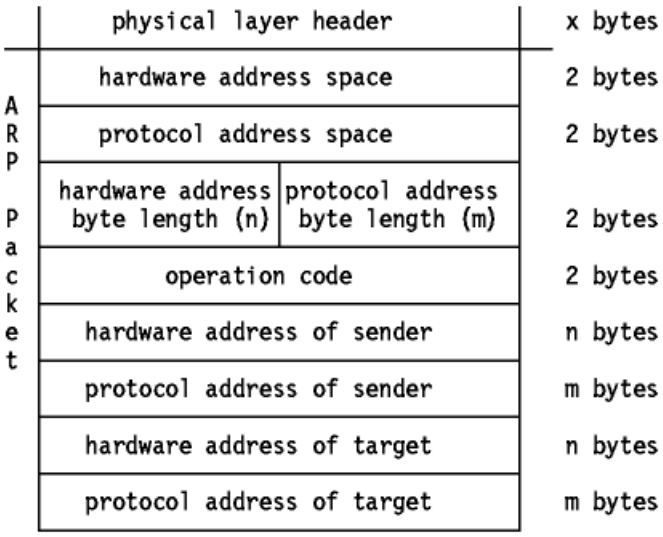
En una red física, los hosts individuales se conocen en la red a través de su dirección física[31]. Los protocolos de más alto nivel[32] direccionan a los hosts de destino con una dirección lógica[33]. Cuando se quiere enviar un mensaje a una dirección IP de destino que se encuentra en nuestra red, 172.26.1.5, el manejador de dispositivo físico no sabe a qué dirección MAC enviarla.

Para resolver este problema, se suministra un protocolo (el ARP) que traducirá la dirección IP a la dirección física del host de destino. No es un protocolo que transporte datos, sino que tiene un propósito específico: **asociar direcciones lógicas con direcciones físicas**.

Existe un protocolo, el RARP (Reverse ARP) cuyo propósito es el complementario: sabiendo una dirección física, obtener la dirección lógica que le corresponde. Se utiliza cuando tenemos un dispositivo (una impresora, por ejemplo) que debe arrancar e integrarse en la red con una dirección lógica, y lo único que conoce es su dirección física. En este caso debe existir un servidor (con el protocolo DHCP o bootp) que se encargue de gestionar estas asignaciones.

Comentemos cómo funciona el protoolo ARP. Utiliza una tabla (llamada caché ARP[34]) para realizar esta traducción. Cuando la dirección física no se encuentra en la caché ARP, se envía un broadcast a la red[35], con un formato especial llamado petición ARP. Si una de las máquinas en la red reconoce su propia dirección IP en la petición, devolverá una respuesta ARP al host que la solicitó. La respuesta contendrá la dirección física del hardware, así como información de encaminamiento (si el paquete ha atravesado puentes[36] durante su trayecto). Tanto esta dirección como la ruta se almacenan en la tabla caché ARP del host solicitante. Todos los posteriores datagramas enviados a esta dirección IP se podrán asociar a la dirección física correspondiente.

[31] Como la dirección MAC.
[32] Del nivel de red hacia arriba.
[33] en este caso, la dirección IP
[34] Que contiene correspondencias entre direcciones lógicas y direcciones físicas.
[35] Correcto, lo recibirán todas las máquinas de esa red.
[36] Los puentes (bridges) conectan varias redes físicas que pertenecen a la misma red lógica. Trabajan a nivel físico y de enlace.



Este es el formato del paquete ARP, donde tras la cabecera de la capa física vienen:

- *Hardware address space*: Especifica el tipo de hardware; ejemplos son Ethernet o Packet Radio.
- *Protocol address space*: Especifica el tipo de protocolo, el mismo que en el campo de tipo EtherType en la cabecera de IEEE 802.
- *Hardware address length*: Especifica la longitud (en bytes) de la dirección hardware del paquete. Para Ethernet e IEEE 802.3 será de 6 bytes.
- *Protocol address length*: Especifica la longitud (en bytes) de la dirección lógica. Para IP será de 4 bytes.
- *Operation code*: Especifica el tipo de operación ARP.
- *Source/target hardware address*: Contiene las direcciones físicas hardware. En IEEE 802.3, son direcciones de 48 bits.
- *Source/target protocol address*: Contiene las direcciones lógicas. En TCP/IP son direcciones IP de 32 bits.

Para el paquete de solicitud, la dirección hardware de destino es el único campo indefinido del paquete, que es el que pretende obtener.

El host solicitante recibirá la respuesta ARP en caso de que la máquina buscada se encuentre en la red, y seguirá el proceso ya comentado para tratarla. Como resultado, la tripleta `<tipo de protocolo, dirección de protocolo, dirección hardware>` para el host en cuestión se añadirá a la caché ARP. La próxima vez que un protocolo de nivel superior quiera enviar un paquete a ese host, el módulo de ARP ya tendrá la dirección hardware, a la que se enviará el paquete.

↬ **Ejemplo:** Mediante el comando `arp` podemos mostrar la caché arp en un sistema linux.

```
[root@linux entrega05-1]# arp -e

Address HWtype HWaddress Flags Mask Iface

172.26.0.1 ether 00:01:38:11:D6:03 C eth0
```

4.2.3. Protocolo IP

IP es el protocolo que oculta la red física subyacente creando una vista de red virtual[37]. Es un protocolo de entrega de paquetes no fiable y no orientado a conexión, y se puede decir que aplica la ley del mínimo esfuerzo.

No aporta fiabilidad, control de flujo o recuperación de errores. Los paquetes (datagramas) que envía IP se pueden perder, desordenarse, o incluso duplicarse, e IP no manejará estas situaciones. Proporcionar estos servicios depende de protocolos de capas superiores.

IP asume pocas cosas de las capas inferiores, sólo que los datagramas "probablemente" serán transportados a la estación de destino.

La versión actual es la IP versión 4 (IPv4). Desde mediados de los años 90 se escucha que será sustituida por la nueva generación IPv6, pero diversos factores han hecho que no se haya producido dicha sustitución.

1.- IPv4 ha ido desarrollando mecanismos para resolver sus deficiencias, como el agotamiento de direcciones IP contrarrestado con el uso de direccionamiento privado, o la aparición de protocolos como SSL para añadirle la seguridad de la que carecía.

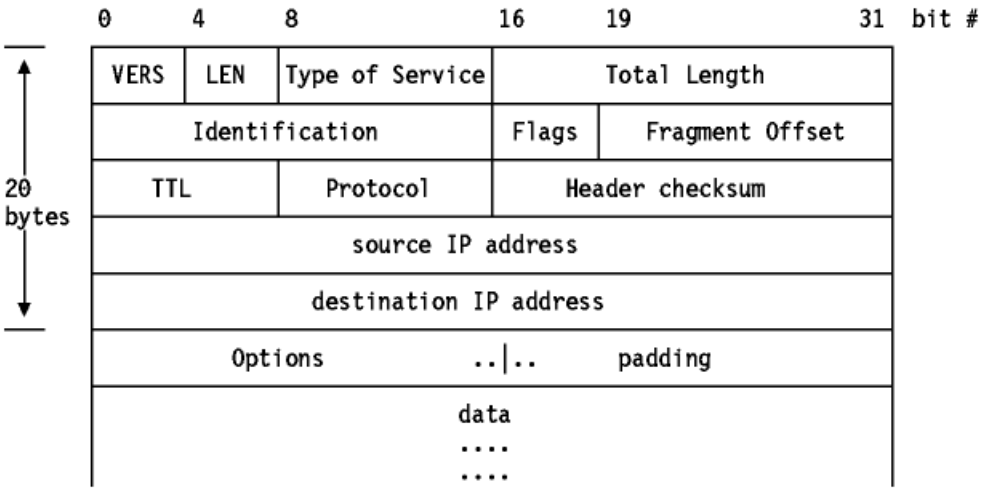
2.- IPv6 ha tardado en desarrollarse y probarse más de lo esperado, además de no proveer mecanismos suaves de evolución. Por ejemplo, al pasar de redes Ethernet a 10Mbps a redes de 100Mbps ó de 1000Mbps, el cambio en la mayoría de los sitios casi ha sido imperceptible.

3.- Las empresas y organismos han sentido pereza ante el cambio: si lo que tengo me funciona, porqué voy a cambiarlo, unido a la mayor complejidad de IPv6 que necesitará una formación y entrenamiento.

Hablaremos por tanto, todavía del "anciano" IPv4 y en el apéndice A, avanzaremos conceptos sobre IPv6.

Estructura del datagrama IP El datagrama IP es la unidad de transferencia en la pila IP. Tiene una cabecera con información para IP, y su carga de datos para los protocolos superiores.

La cabecera del datagrama IP es de un mínimo de 20 bytes de longitud:



En la figura se muestra la estructura del datagrama IP, donde:

- *VERS*: La versión del protocolo IP. La versión actual es la 4. La 5 es experimental y la 6 es la nueva generación IPv6[38].

[37] o red lógica.
[38] Véase A en la página 109

- *LEN*: La longitud de la cabecera IP contada en cantidades de 32 bits. No incluye el campo de datos.
- *Type of Service*: El tipo de servicio es una indicación de la calidad del servicio solicitado para este datagrama IP.
- *Total Length*: La longitud total del datagrama, cabecera y datos, especificada en bytes.
- *Identification*: Un número único que asigna el emisor para ayudar a reensamblar un datagrama fragmentado. Los fragmentos de un datagrama tendrán el mismo número de identificación.
- *Flags*: Varios flags de control. Donde:

  0 está reservado, debe ser cero

  DF: No fragmentar (*Don't Fragment*): con 0 se permite la fragmentación, con 1 no.

  MF: Más fragmentos (*More fragments*): 0 significa que se trata del último fragmento del datagrama, 1 que no es el último.

- *Fragment Offset*: Usado con datagramas fragmentados, para ayudar al reensamblado de todo el datagrama. El valor es el número de partes de 64 bits (no se cuentan los bytes de la cabecera) contenidas en fragmentos anteriores. En el primer (o único) fragmento el valor es siempre cero.
- *Time to Live*: Especifica el tiempo (en saltos de router) que se le permite viajar a este datagrama. Cada "router" por el que pase este datagrama ha de sustraer uno de este campo. Cuando el valor alcanza cero, se asume que este datagrama ha estado viajando sin llegar a su destino por alguna razón y se desecha. El valor inicial lo deberá fijar el protocolo de alto nivel que crea el datagrama. Es una forma de eliminar los paquetes zombies vagando eternamente por la red.
- *Protocol Number*: Indica el protocolo de alto nivel al que IP deberá entregar los datos del datagrama. Algunos valores son:

  0 Reservado

  1 ICMP (*Internet Control Message Protocol*)

  2 IGMP (*Internet Group Management Protocol*)

  3 GGP (*Gateway-to-Gateway Protocol*)

  4 IP (IP encapsulation)

  5 Flujo (*Stream*)

  6 TCP (*Transmission Control*)

  8 EGP (*Exterior Gateway Protocol*)

  17 UDP (*User Datagram*)

  89 OSPF (*Open Shortest Path First*).

- *Header Checksum*: Es el checksum de la cabecera. Si su comprobación no es válida, el datagrama se desecha, ya que al menos un bit de la cabecera está corrupto, y el datagrama podría haber llegado al destino equivocado. No tenemos la seguridad de que lo que esté mal pueda ser la dirección de destino.

- *Source IP Address*: La dirección IP de 32 bits del host emisor.
- *Destination IP Address*: La dirección IP de 32 bits del host receptor.
- *Options*: No requiere que toda implementación de IP sea capaz de generar opciones en los datagramas que crea, pero sí que sea capaz de procesar datagramas que contengan opciones. El campo "Options" (opciones) tiene longitud variable. Puede haber cero o más opciones.

➥ **Ejemplos:**

1. El **comando ping** se utiliza para comprobar la conectividad a nivel IP entre nuestro sistema y un sistema destino. Por ejemplo, desde la máquina mileto.cica.es hasta el servidor web del Instituto Tecnológico de Massachussets (MIT).

```
[root@mileto root]# ping www.mit.edu
PING DANDELION-PATCH.mit.edu (18.181.0.31) from 150.214.5.11 : 56(84) bytes of data.
64 bytes from DANDELION-PATCH.MIT.EDU (18.181.0.31): icmp_seq=1 ttl=243 time=126 ms
64 bytes from DANDELION-PATCH.MIT.EDU (18.181.0.31): icmp_seq=2 ttl=243 time=127 ms
64 bytes from DANDELION-PATCH.MIT.EDU (18.181.0.31): icmp_seq=3 ttl=243 time=126 ms
64 bytes from DANDELION-PATCH.MIT.EDU (18.181.0.31): icmp_seq=4 ttl=243 time=126 ms
64 bytes from DANDELION-PATCH.MIT.EDU (18.181.0.31): icmp_seq=5 ttl=243 time=128 ms
64 bytes from DANDELION-PATCH.MIT.EDU (18.181.0.31): icmp_seq=6 ttl=243 time=126 ms
--- DANDELION-PATCH.mit.edu ping statistics ---
6 packets transmitted, 6 received, 0% loss, time 5053ms
rtt min/avg/max/mdev = 126.358/126.871/128.440/0.799 ms
```

2. El **comando traceroute** nos indica los routers o pasarelas por las que tiene que atravesar un paquete para llegar desde nuestro host hasta el destino que especifiquemos. Para ello, utiliza una pequeña argucia utilizando el valor del campo TTL del paquete IP. Envía primero un paquete al sistema destino con un valor de 1 en el campo TTL. El router al que llega descuenta 1 y al llegar el valor TTL a 0, devuelve un paquete de error ICMP. El siguiente paquete posee un valor de 2 en el TTL, con lo cual llegará a 0 al pasar por el segundo router, que será el que envíe el paquete de error. Así se descubre la ruta que siguen nuestros paquetes hasta el sistema destino.

   Por ejemplo, desde mileto.cica.es hasta www.mit.edu pasa por los siguientes routers:

```
[root@mileto root]# traceroute www.mit.edu
traceroute to DANDELION-PATCH.mit.edu (18.181.0.31), 30 hops max, 38 byte packets
1 150.214.5.28 (150.214.5.28) 0.622 ms 0.466 ms 0.460 ms
2 rt1 (150.214.3.4) 0.460 ms 0.606 ms 0.460 ms
3 GE0-1-0.EB-Sevilla0.red.rediris.es (130.206.194.1) 0.601 ms 0.763 ms 0.460 ms
4 AND.SO4-1-0.EB-IRIS2.red.rediris.es (130.206.240.17) 8.037 ms 8.036 ms 8.104 ms
5 SO0-0-0.EB-IRIS4.red.rediris.es (130.206.240.2) 8.106 ms 8.066 ms 8.134 ms
6 rediris.es1.es.geant.net (62.40.103.61) 8.332 ms 8.339 ms 8.312 ms
7 es.it1.it.geant.net (62.40.96.186) 30.908 ms 31.126 ms 30.848 ms
8 it.de2.de.geant.net (62.40.96.61) 40.035 ms 40.076 ms 40.018 ms
9 abilene-gw.de2.de.geant.net (62.40.103.254) 133.684 ms 133.718 ms 133.687 ms
10 nycmng-washng.abilene.ucaid.edu (198.32.8.84) 120.839 ms 121.198 ms 120.859 ms
11 ATM10-420-OC12-GIGAPOPNE.nox.org (192.5.89.9) 125.835 ms 125.970 ms 125.919 ms
12 192.5.89.90 (192.5.89.90) 125.996 ms 126.120 ms 126.059 ms
13 NW12-RTR-2-BACKBONE.MIT.EDU (18.168.0.21) 126.043 ms 126.590 ms 126.435 ms
14 DANDELION-PATCH.MIT.EDU (18.181.0.31) 126.536 ms * 130.035 ms.
```

4.3. Nivel de transporte: TCP

4.3.1. Puertos y Sockets

Puertos

Lejos quedan aquellos tiempos del MS-DOS (por ejemplo) en los que en un ordenador se podía hacer una sola cosa a la vez. Afortunadamente, hoy podemos estar escuchando música con el ordenador, a la vez que navegamos por Internet y realizamos complejos cálculos. De esta forma, cuando una comunicación llega a nuestro ordenador, además debe saber a qué proceso va dirigida.

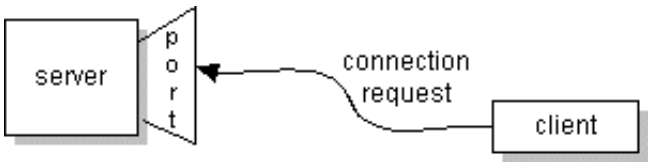
Un puerto es un número de 16 bits (de 1 a 65.535), empleado para identificar a qué protocolo del nivel superior o programa de aplicación se deben entregar los mensajes recibidos. Podemos asimilarlo a un número de atraque donde puede situarse un barco dentro de un puerto[39] .

Hay números de puerto que por convención se asignan a determinadas aplicaciones y/o protocolos[40], como el 80 para el http, el 23 para telnet o el 25 para smtp.

Sockets

En castellano la palabra *socket* significa enchufe, y esa es precisamente la función que realiza: un enchufe que conecta dos procesos. Un socket está formado por una dirección IP y un puerto. Un ejemplo, en el formato *dirección IP:puerto*, sería 172.26.0.2:80.

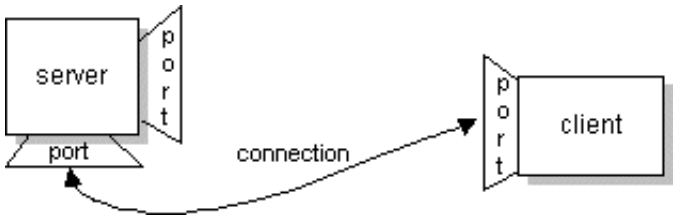
Cuando un proceso servidor se inicia, abre un socket en la máquina y se pone a esperar a que alguien se conecte.



Del mismo modo, cuando un proceso cliente quiere conectarse, abre otro socket en su máquina. La conexión se establece cuando ambos sockets se conectan, formando la tupla

DirecciónOrigen:PuertoOrigen⟶DirecciónDestino:PuertoDestino

Indicamos con la flecha la dirección del cliente al servidor, o más concretamente, desde quién ha solicitado la apertura de la conexión hasta quién la ha recibido.

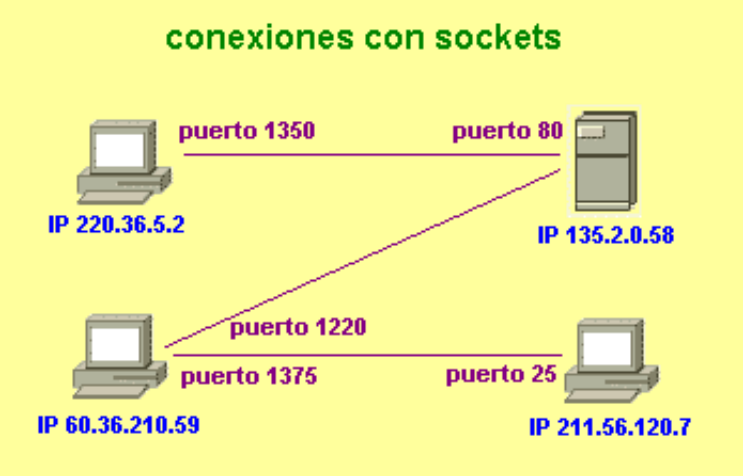


En la figura siguiente hay tres conexiones. Vamos a identificarlas, empezando por las que están situadas más arriba. Suponemos para el ejemplo que las máquinas cliente se sitúan a la izquierda y las máquinas servidoras a la derecha.

La primera conexión sería 220.36.5.2:1350⟶135.2.0.58:80. Probablemente sea una conexión a un servidor web[41], ya que el puerto 80 normalmente es utilizado para ello. El servidor web primero abre el socket 135.2.0.58:80, al que se conecta el cliente desde el socket 220.36.5.2:1350. Con este mecanismo, cada conexión tiene una identificación única. Si el cliente abriera una nueva conexión

[39] de los de verdad, con barcos.
[40] En el fichero `/etc/services` podemos encontrar un gran número de ellos
[41] Pero no tenemos la total seguridad, los puertos son convenciones que pueden respetarse o no.

con el mismo servidor web, su identificación sería 220.36.5.2:1351⟶135.2.0.58:80. Observamos que el puerto del cliente ha cambiado (de 1350 a 1351) y las dos conexiones tienen identificaciones distintas. Cuando un cliente abre un nuevo socket, se le asigna un número de puerto más alto y que no esté ocupado. Si llegara al más alto, empezaría de nuevo por el más bajo que esté libre.



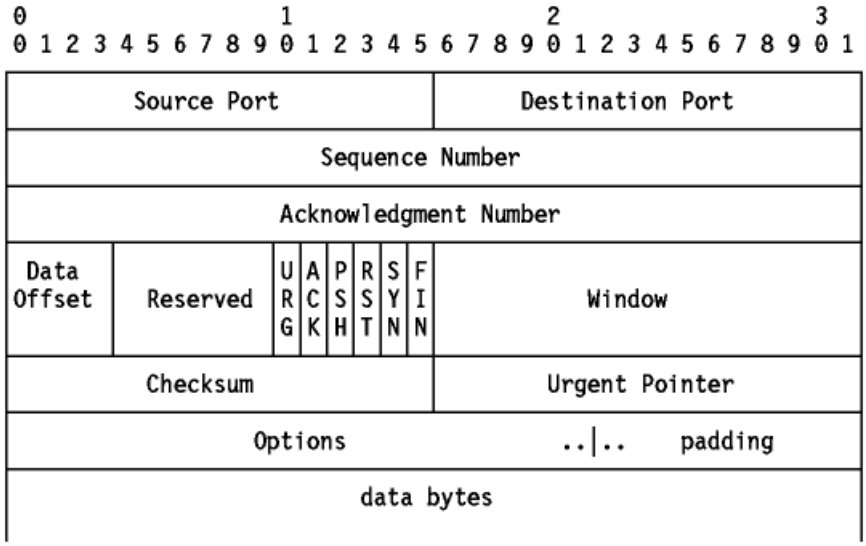
La siguiente conexión de la figura sería: 60.36.210.59:1220⟶135.2.0.58:80, correspondiente a otra máquina cliente conectándose al mismo servidor web de antes. Como vemos, el servidor web y los clientes pueden identificar perfectamente cada una de las conexiones.

La última conexión, definida por 60.36.210.59:1375⟶211.56.120.7:25, podría ser a un servidor de correo SMTP (porque ese es el número de puerto al que se conecta).

4.3.2. Protocolo TCP

El principal propósito de TCP es proporcionar una conexión lógica fiable entre dos procesos. Asume que los protocolos de niveles inferiores (como IP) no garantizan la fiabilidad, por lo que debe ocuparse de garantizarla.

Formato de segmento TCP:



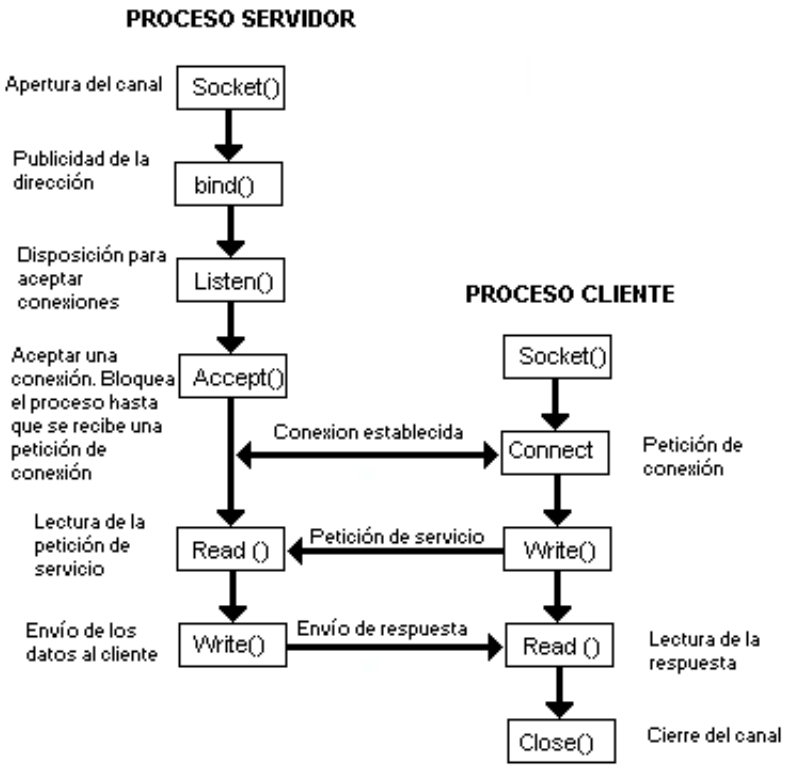
Esta es la estructura de un segmento TCP, donde:

- *Source Port*: Es el número de puerto de 16 bits del emisor, que el receptor usa para responder.
- *Destination Port*: El número de puerto de 16 bits del receptor.
- *Sequence Number*: El número de secuencia del primer byte de datos del segmento. Si el byte de control SYN está a 1, el número de secuencia es el inicial (n) y el primer byte de datos será el n+1.
- *Acknowledgment Number*: Si el bit de control ACK está a 1, este campo contiene el valor del siguiente número de secuencia que se espera recibir.
- *Data Offset*: El número de palabras de 32 bits de la cabecera TCP. Indica dónde empiezan los datos.
- *Reserved*: Seis bits reservados para su uso futuro; deben ser cero.
- *URG*: Indica que el campo *urgent pointer* es significativo en el segmento.
- *ACK*: Indica que el campo de reconocimiento (acuse de recibo) es significativo en el segmento.
- *PSH*: Función *Push*.
- *RST*: Resetea la conexión.
- *SYN*: Sincroniza los números de secuencia.
- *FIN*: No hay más datos del emisor.
- *Window*: Usado en segmentos ACK (de acuse de recibo). Especifica el número de bytes de datos que el receptor está dispuesto a aceptar hasta que le llegue el siguiente acuse de recibo.
- *Checksum:* Campo de 16 bits que permite verificar tanto la cabecera como los datos TCP.
- *Urgent Pointer*: Apunta al primer octeto de datos que sigue a los datos importantes. Sólo es significativo cuando el bit de control URG está a uno.
- *Options:* Sólo para el caso de opciones en los datagramas IP.
- *Padding Bytes*: Todos a cero para rellenar la cabecera TCP a una longitud total que sea un múltiplo de 32 bits.

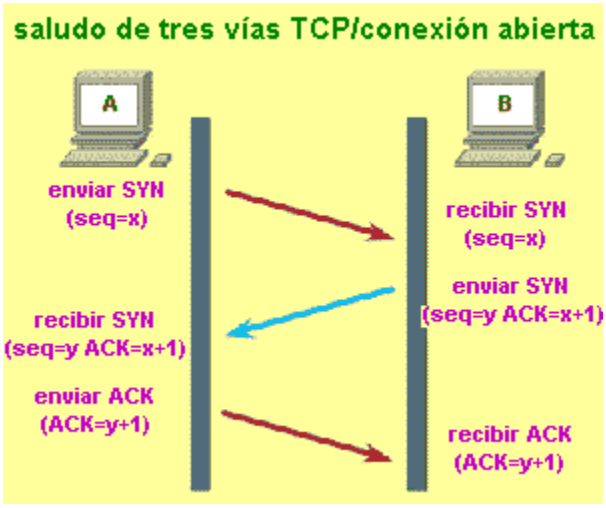
Como vemos, las direcciones IP de origen y destino no son necesarias, eso se ha quedado para el protocolo IP.

Estableciendo una conexión TCP

Antes de que se pueda transferir cualquier dato, se ha de establecer una conexión entre los dos procesos. Uno de los procesos (normalmente el servidor) lanza una llamada para abrir un socket pasivo. El servidor permanece en espera hasta que otro proceso intenta comunicarse con él a través de una solicitud de conexión. El proceso cliente lanza una llamada de apertura de socket y conexión al servidor.



En la red, en el momento de establecimiento de la conexión, se intercambian tres segmentos TCP:



Este proceso completo se conoce como *three-way handshake*, o acuerdo en tres fases. Notar que los segmentos TCP intercambiados incluyen los números de secuencia iniciales de ambas partes, para ser usados en posteriores transferencias. El cliente envía un paquete para iniciar la conexión con el bit SYN activo. Esta característica la utilizarán los cortafuegos, que veremos en una sección posterior, para denegar el establecimiento de conexiones TCP. Sin el envío de este bit SYN no se podrá establecer una nueva conexión. El servidor, en el paquete de vuelta realiza un acuse de recibo (ACK) del paquete anterior y envía a su vez un SYN al cliente. El tercer paquete consiste

en el envío del acuse de recibo por parte del cliente. A partir de ahí la conexión TCP se encuentra activa para que cliente y servidor se transfieran datos.

El cierre de la conexión se hace de forma implícita enviando un segmento TCP con el bit FIN activo. Como la conexión es *full duplex*[42], el segmento FIN sólo cierra la conexión en un sentido del canal. El otro proceso enviará los datos restantes si quedaran, seguidos de un segmento TCP en el que el bit FIN está activo. La conexión se borra (es decir, la información de estado en ambos extremos) una vez que el canal se ha cerrado en ambos sentidos.

➥ **Ejemplo:** Para ver los sockets y conexiones de nuestro sistema, utilizamos la **orden *netstat*.** En el comando siguiente, le indicamos que nos muestre todos los sockets (opción `-a`, los conectados y los que están a la escucha), tanto del protocolo tcp (opción `t`) como del protocolo udp (opción `u`).

```
[root@mileto root]# netstat -atu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 *:7937 *:* LISTEN
tcp 0 0 *:http *:* LISTEN
tcp 0 0 *:ssh *:* LISTEN
tcp 0 0 *:smtp *:* LISTEN
tcp 0 0 *:https *:* LISTEN
tcp 0 0 mileto.cica.es:ssh 106.Red-81-44-35.p:1394 ESTABLISHED
tcp 0 0 mileto.cica.es:19094 dolmene.cica.es:7945 ESTABLISHED
udp 0 0 localhost.localdo:32769 localhost.localdo:32769 ESTABLISHED
udp 0 0 *:7938 *:*
udp 0 0 *:863 *:*
```

Comentemos toda la información interesante que nos presenta. En las primeras cinco líneas, vemos que hay cinco procesos servidores esperando conexiones en los puertos de http, https, smtp, ssh y 7937. Los que reciben un nombre, por ejemplo http (que es el 80) es porque han encontrado una línea en el fichero `/etc/services` y nos muestra ese nombre. El puerto 7937 no ha tenido esa suerte y no ha sido "bautizado". Habría que mirar qué proceso es el que se encuentra ahí escuchando para quedarnos tranquilos:

```
[root@mileto root]# fuser -n tcp 7937
7937/tcp: 879
```

Con el comando `fuser` podemos identificar procesos que utilizan ficheros o sockets (al fin y al cabo todo en Unix/Linux son ficheros, hasta los sockets). La opción `-n tcp` le indica que busque en el espacio de tcp[43]. En este caso nos ha dicho que el socket tcp que escucha en el puerto 7937 le corresponde al proceso 879. Pues vamos a mirar qué proceso es ese.

```
[root@mileto root]# ps aux|grep 879
root 879 0.0 0.1 3008 996 ? S Jan18 0:00 /usr/sbin/nsrexec
```

¡Ah!, resulta que es el proceso que hace las copias de seguridad de la máquina (el Legato Networker). Podemos estar tranquilos por dos razones, porque no es un proceso que un intruso nos ha colocado en nuestra máquina[44] y además, vuestras prácticas[45] no se perderán aunque el disco duro se rompa. Las recuperaremos de la copia de seguridad ;-).

[42] En ambos sentidos simultáneamente.
[43] Otras opciones serían *udp* o *file* para ficheros.
[44] Aunque para eso deberíamos asegurarnos de que el programa es realmente quien dice ser y al terminar el curso seguro que estaréis en condiciones de saberlo.
[45] Cuando las coloquéis.

4.4. Nivel de Aplicación

Como ejemplo de nivel de aplicación, veremos el protocolo HTTP, utilizado entre los navegadores y los servidores web.

4.4.1. Protocolo HTTP

El Protocolo de Transferencia de HiperTexto (*Hypertext Transfer Protocol*) es un protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1.0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el *World Wide Web*.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP e IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor[46] escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia) es conocido por su URL[47].

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Dirección (*Location*) del cliente Web.

2. El cliente Web decodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso (http), la dirección DNS o IP del servidor (`mileto.cica.es`), el puerto (como en este caso no aparece, toma el valor por defecto, que es 80) y el objeto requerido del servidor (en este caso el recurso raíz /).

3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.

4. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del browser o datos opcionales para el servidor.

5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME[48] de la información de retorno, seguido de la propia información.

6. Cuando no hay más solicitudes, se cierra la conexión TCP.

➥ Veamos este proceso con un ejemplo:

1. Desde un navegador web, solicitamos la dirección `http://thales.cica.es`. El navegador (mozilla en este caso), se encarga de abrir una conexión tcp con el servidor web de la máquina `thales.cica.es` en el puerto 80. El navegador envía la solicitud siguiente, que cumple con el protocolo HTTP:

   ```
   GET / HTTP/1.1
   ```

[46] En los entornos Unix, el proceso servidor que se ejecuta en segundo plano, esperando conexiones de los clientes se denomina demonio (*daemon*). Profundizaremos en ello en la segunda entrega.

[47] Localizador Universal de Recursos. Por ejemplo: `http://mileto.cica.es`

[48] Identifica el tipo de objeto multimedia (audio, vídeo, texto...)

(es una solicitud GET, del objeto / y cumpliendo la versión HTTP/1.1)

```
Host: thales.cica.es
User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-ES; rv:1.4.1) Gecko/20031030
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-
mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: es-es,es;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

2. El servidor `thales.cica.es` recibe la petición, la procesa y devuelve lo siguiente:

   ```
   HTTP/1.1 200 OK
   ```

   (El servidor devuelve en el protocolo HTTP/1.1 con el código de retorno 200, sin errores)

   ```
   Date: Sat, 10 Jan 2004 23:19:47 GMT
   Server: Apache/1.3.27 (Unix) (Red-Hat/Linux) mod_ssl/2.8.12 OpenSSL/0.9.6b DAV/1.0.2
   PHP/4.0.6 mod_perl/1.26
   Last-Modified: Fri, 19 Dec 2003 22:12:36 GMT
   ETag: "bd98c-1e9e-3fe377d4"
   Accept-Ranges: bytes
   Content-Length: 7838
   Content-Type: text/html
   Age: 46842
   ```

   (Aquí comienza el contenido HTML, con la cabecera y el cuerpo)

   ```
   <html>
   <head>
   <meta http-equiv="Content-Type"
   content="text/html; charset=iso-8859-1">
   <meta name="Author" content="Antonio Aranda Plata">
   <meta name="GENERATOR" content="Microsoft FrontPage Express 2.0">
   <title>S.A.E.M. THALES</title>
   </head>
   <body background="http://thales.cica.es./thales/BACK.GIF"
   bgcolor="#DDFFFE" text="#000000" link="#004091" vlink="#303031"
   alink="#3366FF" nosave>
   <p align="center"> </p>
   <div align="center"><center>
   <table border="0" cellpadding="0" cellspacing="0" width="100%"...
   ```

4.5. Ver para creer: Ethereal

La mejor forma de comprender todos estos conceptos es viéndolos, y ése va a ser nuestro objetivo, desmenuzar y ver cada uno de los protocolos.

Para ello, utilizaremos Ethereal. Ésta es una herramienta muy útil a la hora de comprobar el tráfico de la red con objeto de solucionar problemas. Contiene una serie de funcionalidades que nos permiten localizar rápidamente los problemas de nuestra red:

- Permite centrarnos en paquetes y protocolos concretos.
- Soporta un gran número de protocolos.

- Proporciona una visión clara del tráfico de nuestra red, proporcionando herramientas que nos permiten distinguir y aislar los paquetes que lo constituyen por distintos criterios.

Además, esta herramienta proporciona varios decodificadores[49] de protocolos que permiten realizar filtrados selectivos por protocolos. En total, Ethereal maneja 335 protocolos de red, entre los que se encuentran TCP, SMB, Telnet, LDAP o SNMP.

Ethereal también proporciona gráficos basados en el tráfico de red así como en el RTT[50]. Estos gráficos son fáciles de obtener y son interactivos, pulsando sobre un punto del gráfico podemos ver el paquete correspondiente en la ventana de análisis.

4.5.1. tcpdump

Realmente, Ethereal es una GUI de `tcpdump`[51], una herramienta de dominio público que imprime las cabeceras de los paquetes que pasan por una interfaz de red. Es posible ejecutar `tcpdump` en modo promiscuo, con lo que capturaremos los paquetes que viajen por la red. Así, podemos acceder a la distinta información que se almacena en los campos que constituyen un paquete. Es una herramienta que entraría dentro de las que denominamos como *sniffers* de red.

Tanto en la captura como en la visualización es posible aplicar filtros por protocolo (TCP, UDP, IP, ARP, RARP, ...), puertos (pueden ser números o los nombres según aparecen en `/etc/services`), direcciones fuente, direcciones destino, direcciones de red y, mediante el uso de operadores, una combinación de éstos.

Puede tambien usarse para comprobar los datos obtenidos en una captura realizada con anterioridad, pudiendo ser la fuente de estos datos otro programa[52]. Del mismo modo, la captura obtenida por `tcpdump` puede exportarse a un formato que puedan leer estos programas.

No entraremos en el detalle de cómo ejecutar esta utilidad ya que nos será más cómodo realizar la captura de paquetes utilizando ethereal, nos limitaremos a describir brevemente el uso básico de la misma, así como el significado de la salida que obtenemos.

Obteniendo datos con tcpdump

Hay ocasiones en que la máquina donde estamos no tiene instalado Ethereal. En este caso es mucho más cómodo realizar la captura en modo texto, volcándolo en un fichero, para su posterior análisis detallado con Ethereal en otra máquina.

Al realizar la captura con `tcpdump` hay que tener en cuenta que el resultado de la captura tendrá los paquetes truncados. Por defecto, `tcpdump` únicamente captura los primeros 68 *bytes* de cada paquete. La ejecución que haremos con `tcpdump` para evitar esto será de la forma:

```
tcpdump -i <interface> -s 1500 -w <fichero>
```

Con el tamaño de 1.500, nos aseguramos la captura de la totalidad del paquete en una red ethernet. La captura la finalizamos pulsando `^C`.

```
root@guadalinex:~# tcpdump -i eth0 -s 1500 -w /tmp/captura.dat
tcpdump: listening on eth0, link-
type EN10MB (Ethernet), capture size 1500 bytes
^C
88 packets captured
88 packets received by filter
0 packets dropped by kernel
```

[49] Ethereal los denomina *dissectors*.

[50] *Round Trip Time*

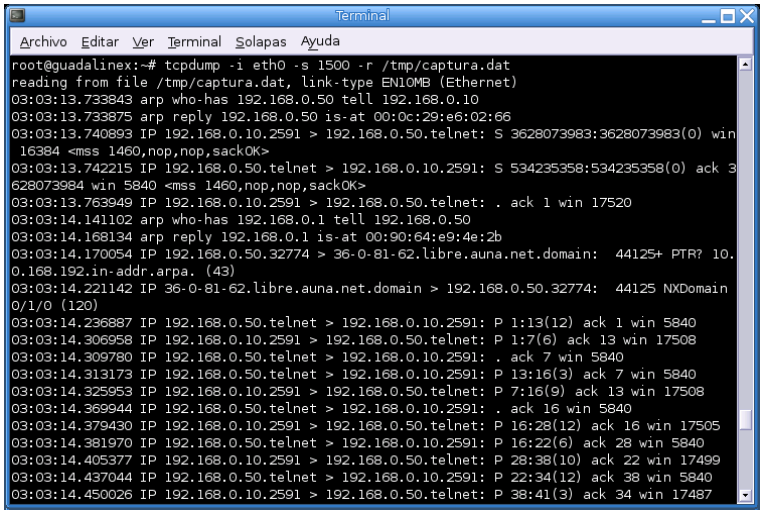
[51] Hay que tener en cuenta que `tcpdump` no viene con la distribución de Ethereal, es un software independiente. Puede obtenerse de `http://www.tcpdump.org`, o con nuestro socorrido `apt-get`.

[52] Además de las capturas realizadas con tcpdump, puede utilizar las realizadas por programas como Network Associates Sniffer y Sniffer Pro, LANanalyzer, Microsoft Network Monitor y otros. Mirar la documentación asociada a esta utilidad para obtener una lista de compatibilidad más amplia.

Interpretando la salida

El formato de la salida que proporcionan los datos capturados por `tcpdump` dependerá del protocolo, aunque todos tienen en común el primer campo que es una marca de tiempo que indica cuándo se realizó la captura.

Figura 4.1: Visualizar captura con `tcpdump`



Veamos ahora cómo se interpretan los paquetes de una captura del protocolo TCP realizado con `tcpdump`. La línea general de un paquete TCP es como sigue:

```
src > dst: flags [dataseq ack window urgent options]
```

En principio `src`, `dst` y `flags` están siempre presentes, dependiendo el resto del tipo de conexión TCP de que se trate. El significado de dichos parámetros es:

- `src`: Dirección y puerto origen. En caso de no especificar el parámetro `-n` se intenta resover el nombre vía DNS y se busca el nombre del puerto en `/etc/services`.
- `dst`: Dirección y puerto destino. En caso de no especificar el parámetro `-n` se intenta resover el nombre vía DNS y se busca el nombre del puerto en `/etc/services`.
- `flags`: Indica los flags de la cabecera TCP. Puede ser un `.` cuyo significado es que no hay flags, o bien una combinación de `S` (SYN), `F` (FIN), `P` (PUSH), `W` (reducción de la ventana de congestión), `E` (ECN eco).
- `dataseq`: El número de secuencia del primer byte de datos en este segmento TCP. El formato es `primero:ultimo(n)`, que significa que desde `primero` a `ultimo` (sin incluir `ultimo`) hay un total de `n` bytes de datos.
- `ack`: El número de asentimiento. Indica el número siguiente de secuencia que se espera recibir.
- `win`: Tamaño de la ventana de recepción.
- `urgent`: Existen datos urgentes.
- `options`: Indica la existencia de opciones. En caso de que haya van entre <y >.

Veamos con datos reales cómo sería una captura sencilla realizada con `tcpdump`:

```
root@guadalinex:~# tcpdump -i eth0 -s 1500
tcpdump: listening on eth0
21:32:16.657523 172.26.0.1.1863 > 172.26.0.40.ssh: S 1473470138:1473470138(0) win 16384 <mss 1460,nop,nop,sackOK> (DF)
21:32:16.657762 172.26.0.40.ssh > 172.26.0.1.1863: S 2409899396:2409899396(0) ack 1473470139 win 5840 <mss 1460,nop,nop,sackOK> (DF)
21:32:16.665792 172.26.0.1.1863 > 172.26.0.40.ssh: . ack 1 win 17520 (DF)

3 packets received by filter
0 packets dropped by kernel
```

Esto simula el inicio de una conexión originada por la máquina `172.26.0.1` con destino a `172.26.0.40`, con el servicio `ssh`.

El significado de las líneas anteriores es:

1. Inicio de conexión de `172.26.0.1` por el puerto `1863` con `172.26.0.40` por el puerto `ssh`. El flag `SYN` está activado, el paquete tiene un número de secuencia `1473470138`, no contiene datos y la ventana de recepción es de `16384` *bytes*.

2. Envío de paquete `SYN` de `172.26.0.40` por el puerto `ssh` a `172.26.0.1` por el puerto `1863`. El número de secuencia es `2409899396`, ventana de `5840` y constituye el `ACK` del `SYN` anterior.

3. `ACK` del `SYN` mandado por `172.26.0.40`. No hay *flags*

En una red formada por concentradores (hubs), el tráfico se replica por todas las puertas (bocas). Si estamos en una red conmutada (formada por switches), los paquetes solamente salen por la boca en la que se encuentra la máquina de destino. Es un poco más segura, pues no es tan fácil capturar los paquetes, aunque herramientas hay para ello.

4.5.2. Arrancando ethereal

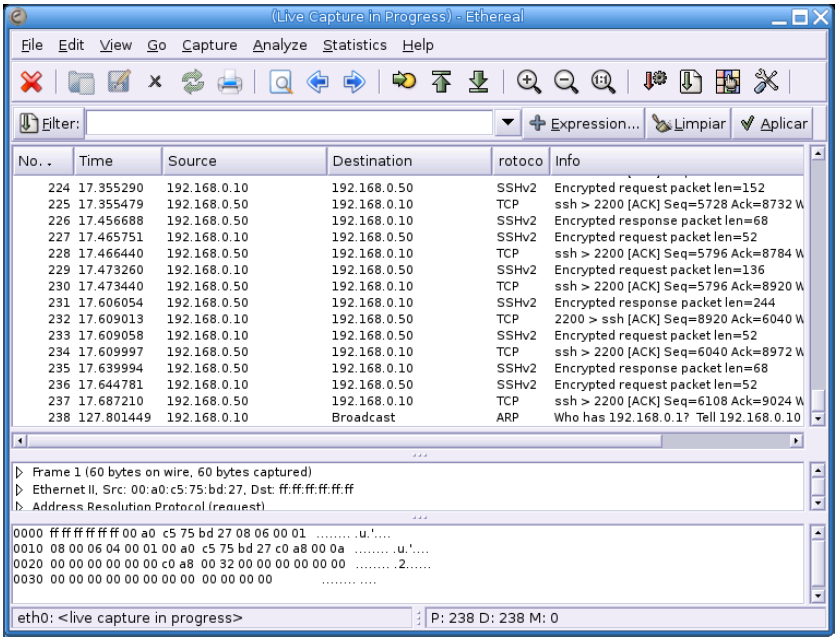
Ethereal está compuesto por tres ventanas o paneles:

1. El panel superior contiene el listado de paquetes. Muestra un resumen de cada paquete capturado. Pulsando sobre los paquetes que aparecen en este panel podemos controlar lo que se muestra en los otros paneles.

2. El panel central muestra una visión en árbol con más detalle del paquete seleccionado en el panel anterior.

3. El panel inferior muestra los datos del paquete. Muestra los datos correspondientes al paquete seleccionado en el panel superior y resalta el campo seleccionado en la visión en árbol.

Figura 4.2: Pantalla de inicio ethereal



Además de estos paneles, hay cuatro elementos en la parte superior/inferior de Ethereal que pueden sernos de utilidad:

Botón Filter. Pulsando este botón se muestra la ventana de filtrado de paquetes, donde pueden contruirse filtros tan complicados como queramos.

Cadena de filtrado. El campo de entrada de texto nos permite editar los filtros. Aquí se muestra también el filtro que se usa en la captura actual.

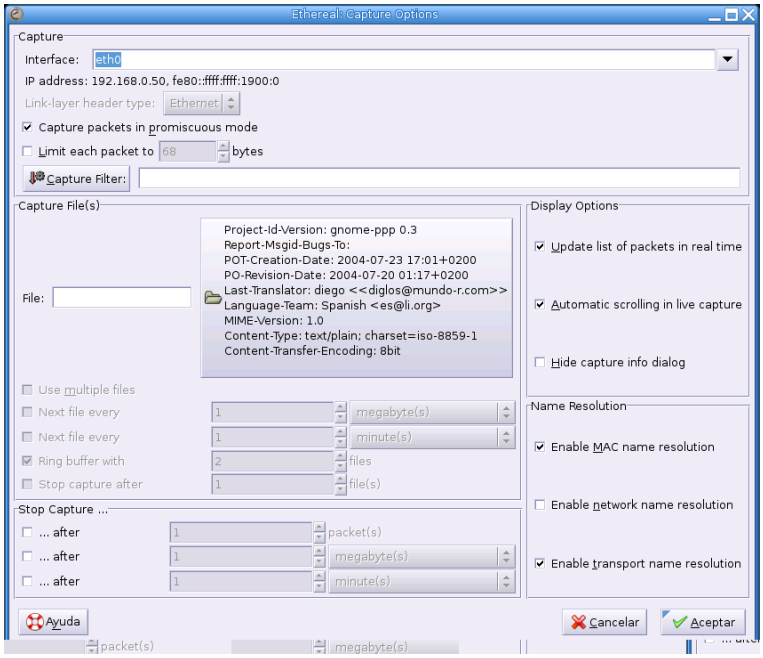
Reset. Limpia el filtro que se está utilizando en la captura actual.

Información general sobre la captura. Muestra información relativa al proceso que realiza en un momento dado Ethereal. Se encuentra en la parte inferior.

4.5.3. Capturando paquetes

Para comenzar una captura de los datos será necesario entrar en el menú **Capture→Start**. Aparecerá un nuevo cuadro de diálogo donde se configurarán algunos aspectos relativos a la captura.

Figura 4.3: Configuración de la captura



- **Interface**. Selecciona el interfaz de red por el que realizaremos la escucha. Sólo podemos realizar capturas por un interfaz de red al mismo tiempo.
- **Limit each packet to**. Indica el tamaño máximo en bytes que van a tener los paquetes que capturemos. En caso de no estar seleccionada la casilla se establece el límite de 65535 bytes por defecto.
- **Capture packets in promiscuous mode**. En caso de no seleccionar esta opción ethereal únicamente capturará los paquetes que circulen entre nuestro equipo y el resto de la red.
- **Filter**. Permite especificar un filtro de captura.
- **File**. Especifica el fichero que guardará la captura. En caso de que no indiquemos ninguno, la realizará en un fichero temporal.
- **Use ring buffer**. En caso de estar seleccionado, indica el número de ficheros que van a utilizarse para guardar las capturas.
- **Rotate capture file every**. Indica el tiempo, en segundos, que transcurre hasta que se realiza la captura en un nuevo fichero.
- **Update list of packets in real time**. Si se selecciona, mostrará las capturas en tiempo real.
- **Automatic scrolling in live capture**. Sólo puede activarse en caso de que la opción anterior lo esté, mostrando así los últimos paquetes que se han capturado realizando un desplazamiento automático de la pantalla con este objeto.
- **Stop capture after**. Establece un punto en el que la captura va a detenerse.
- **Enable resolution**. Indica que las direcciones MAC, las direcciones de red y los números de puerto de la capa de transporte se traduzcan a nombres.

4.5.4. Filtrado durante la captura

Dependiendo del tráfico del segmento de red donde nos encontremos podemos obtener capturas exageradamente grandes. Es recomendable el uso de filtros de forma que únicamente capturemos los datos que necesitemos (con destino/origen en un determinado servidor, los referidos a un protocolo en concreto, ...).

Ethereal usa el lenguaje de filtrado proporcionado con la librería `libpcap`. Pueden encontrarse más detalles al respecto en la página del manual de `tcpdump`.

El formato general que se utiliza para definir los filtros es el siguiente:

```
[not] primitiva [and|or [not] primitiva ...]
```

Entrando en más detalle, las primitivas que pueden utilizarse en la creación de filtros para la visualización de paquetes son:

[src|dst] host <host>

Permite filtrar por una dirección IP o nombre de host, indicando con src y dst si es dirección fuente o destino respectivamente[53]. En el caso que no se indique, se mostrarán los paquetes en los que aparezca la dirección indicada, ya sea fuente o destino.

ether [src|dst] host <host>

Permite filtrar por dirección ethernet.

gateway host <host>

Permite filtrar los paquetes que utilizan host como gateway. Es decir, donde la fuente o el destino ethernet es host pero ni la IP de la fuente ni del destino es host.

[src|dst] net <net>[mask <mask>]

Permite filtrar por número de red, pudiendo especificar la máscara de red.

[tcp|udp] [src|dst] port <port>

Permite filtrar por puertos, ya sean TCP o UDP.

less|greater <length>

Permite filtrar paquetes cuya longitud sea menor/mayor o igual que la longitud especificada respectivamente.

ip|ether proto <protocol>

Permite filtrar en el protocolo específico en la capa Ethernet o la capa IP.

ip|ether broadcast|multicast

Permite filtrar el tráfico broadcast o multicast.

<expr>relop <expr>

Permite crear expresiones de filtrado más complejas que seleccionan bytes o rangos de bytes en paquetes.

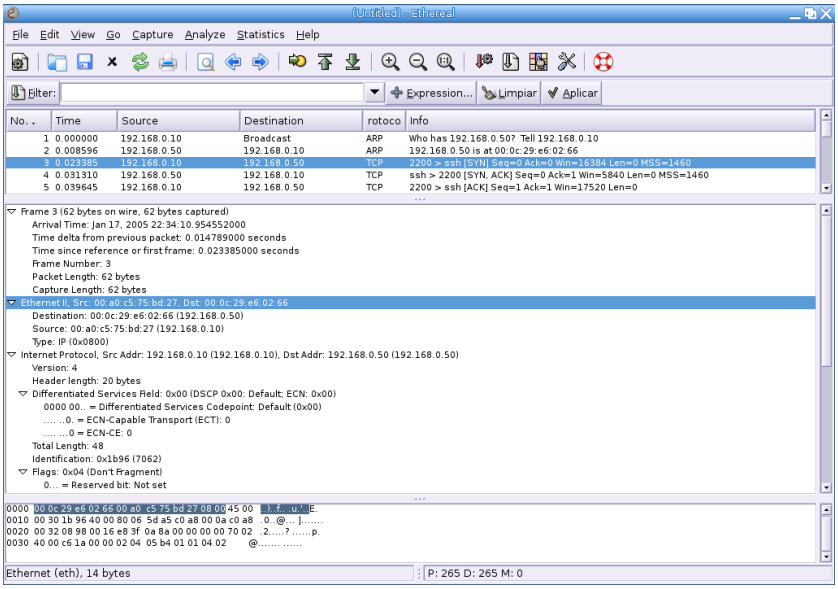
4.5.5. Visualización de los datos capturados

Una vez que hemos finalizado la captura de los paquetes, o estamos visualizando una captura almacenada previamente, tenemos la posibilidad de ver los paquetes con más detalle. Para ello, únicamente tenemos que pulsar sobre un paquete en el panel superior para obtener los detalles del mismo en el panel central.

Podemos expandir cualquier parte del árbol que constituye el paquete pulsando sobre + en la parte izquierda.

[53] Los modificadores src y dst tienen el mismo efecto en las demás combinaciones de filtros.

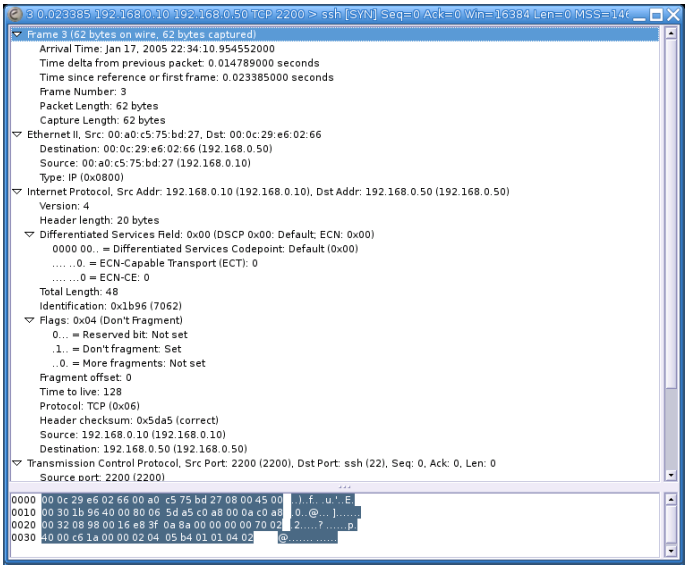
Figura 4.4: Detalles de un paquete capturado



También podemos seleccionar y visualizar paquetes con detalle mientras se está realizando la captura, si hemos seleccionado la opción "**Update list of packets in real time**".

Otra posibilidad para la visualización de paquetes es utilizando una ventana auxiliar, para ello sólo tenemos que pulsar con el botón derecho sobre el paquete previamente seleccionado y aparecerá una nueva ventana con los detalles.

Figura 4.5: Detalles de un paquete capturado en ventana independiente



Hasta aquí la herramienta nos presenta información muy útil del tráfico de la red, pero aún podemos sacarle algo más de jugo. Tal como acabamos de ver, al pulsar sobre cualquier paquete que esté seleccionado con el botón derecho aparece un nuevo menú:

- **Follow TCP Stream**. Permite ver todos los datos que una cadena de paquetes produce entre dos nodos. En una ventana separada muestra todos los segmentos TCP capturados que están en la misma conexión TCP como un solo paquete. Los datos de la secuencia se ordenan y se eliminan los paquetes duplicados, mostrándose en formato ASCII.
- **Decode as**. Permite al usuario forzar a ethereal a decodificar ciertos paquetes como un protocolo en particular.
- **Display Filters**. Permite especificar y manejar filtros.
- **Colorize Display**. Permite colorear los paquetes en el panel de la lista de paquetes de acuerdo a los filtros que elijamos.
- **Print**. Permite imprimir todos los paquetes de una captura.
- **Print Packet**. Permite imprimir el paquete seleccionado.

Además del filtrado de paquetes durante la captura, podemos realizar un filtrado posterior en la visualización de los paquetes capturados o que se están capturando. El lenguaje de filtrado de paquetes en la visualización de los mismos es distinto del utilizado en la captura.

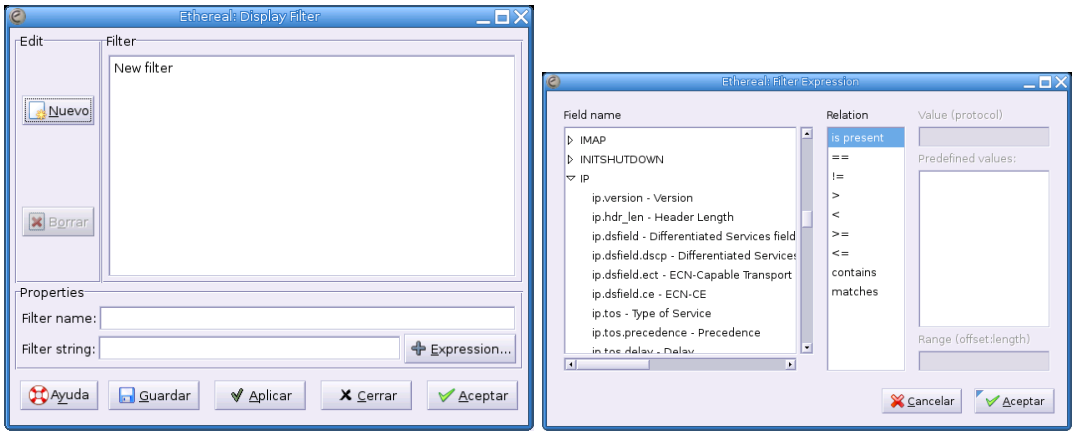
El filtrado de paquetes mientras los visualizamos nos permite concentrarnos en los paquetes en los que estamos interesados. Nos permite seleccionar paquetes según distintos criterios.

- Protocolo
- Presencia de un campo
- Valor de un campo
- Comparación entre campos

4.5.6. Filtrado durante la visualización

Los filtros utilizados durante la visualización de los paquetes tienen más potencia que los utilizados durante la captura. Esto hace que su uso sea algo más complicado. Sin embargo, una vez que nos hayamos peleado con estos filtros, veremos que no es algo difícil y que, normalmente, utilizamos un pequeño subconjunto de etiquetas y expresiones, lo que simplifica su uso. Para acostumbrarnos a su uso es muy útil la ventana **"Filter Expresion"**.

Figura 4.6: Creación de expresiones de filtrado



Para acceder a la ventana de creación de filtros basta pulsar el botón **"Filter"** que aparece en la parte superior.

La ventana de creación de filtros tiene tres listas de campos. La primera de ellas nos muestra los protocolos disponibles y la segunda la relación existente con el valor del tercer campo:

- **Field Name**. Selecciona un campo de protocolo del árbol existente. Cada protocolo tiene una serie de campos por los que se puede filtrar.
- **Relation**. Permite seleccionar una relación de la lista de disponibles. Todas son relaciones binarias que requieren datos adicionales, excepto "is present" que es unaria y que devolverá verdadero/falso dependiendo de si se cumple o no.
- **Value**. En este campo se introduce el valor con el que se quiere comparar. Como ayuda, en el título del campo se indica el tipo de dato que se espera[54].
- **Predefined values**. Algunos protocolos presentarán la opción de elegir entre una serie de valores predeterminados.

Cuando se selecciona un campo de protocolo y una relación binaria, se espera que se proporcione un valor adicional con el que comparar.

Un ejemplo de filtro sería:

```
(ip.addr eq 192.168.0.10 and ip.addr eq 192.168.0.50) and (tcp.port eq 2552 and tcp.port eq 23)
```

Con este filtro se visualizarán/capturarán los paquetes que cumplan que la dirección IP origen o destino sea `192.168.0.10` y `192.168.0.50` y que utilicen los puertos TCP `23` y `2552`.

4.5.7. Capturando una sesión telnet

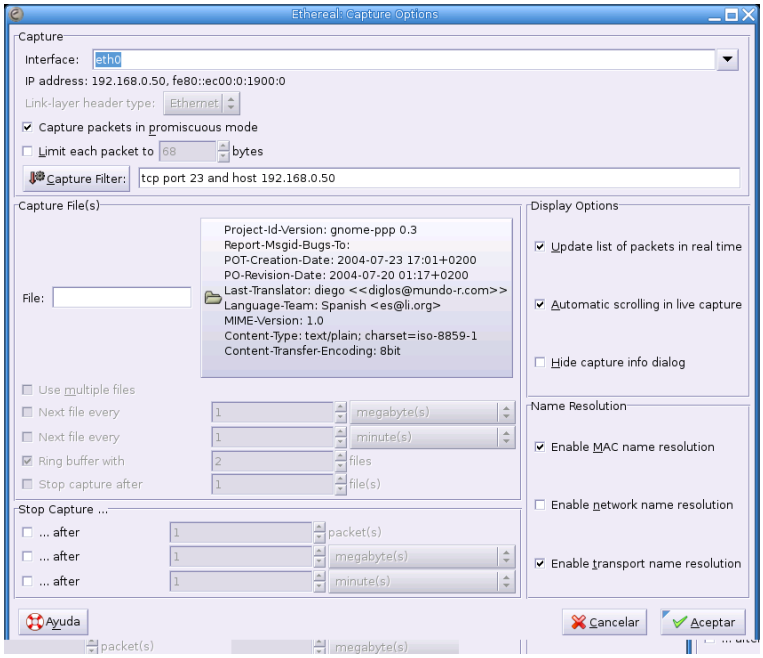
Veamos ahora el resultado de una captura de una sesión telnet que realiza el usuario `legolas`. Mediante la opción "**Follow TCP Streams**" veremos los datos de una sesión de telnet tal y como los vería la capa de aplicación.

Al capturar paquetes en modo promiscuo estaremos capturando todo el tráfico que hay en la red, por lo que utilizaremos los filtros para que capture únicamente los paquetes que circulan hacia/desde el ordenador que tiene el servicio telnet activado con dirección IP `172.26.0.40`. Del mismo modo, al querer capturar únicamente la sesión telnet, indicaremos que el protocolo que queremos capturar es el que utiliza el puerto 23.

```
tcp port 23 and host 172.26.0.40
```

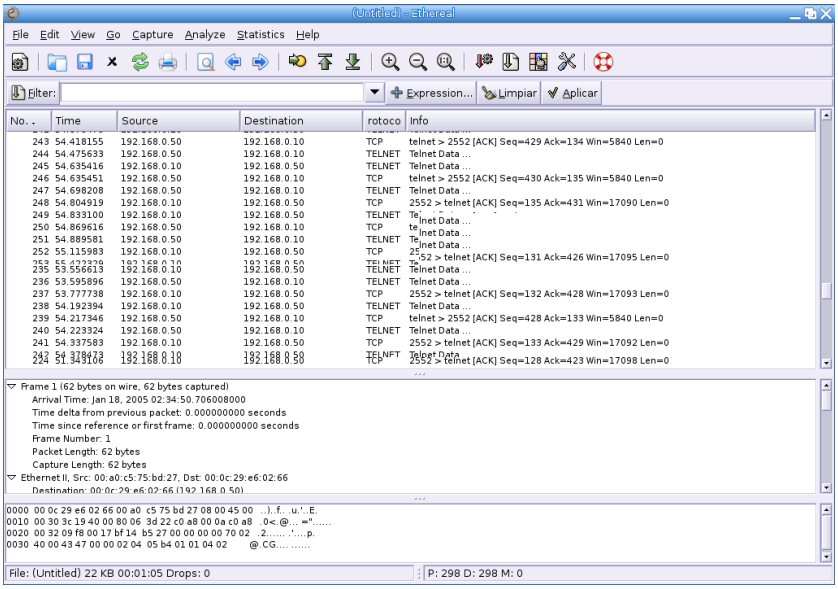
[54] En el caso de seleccionar `ip.addr` se esperará un valor que sea una dirección IPv4.

Figura 4.7: Filtro de captura de sesión de telnet



Después de capturar unos segundos, tendremos una captura de la que podemos sacar completa información de los mensajes que se han transmitido entre el ordenador cliente y el servidor.

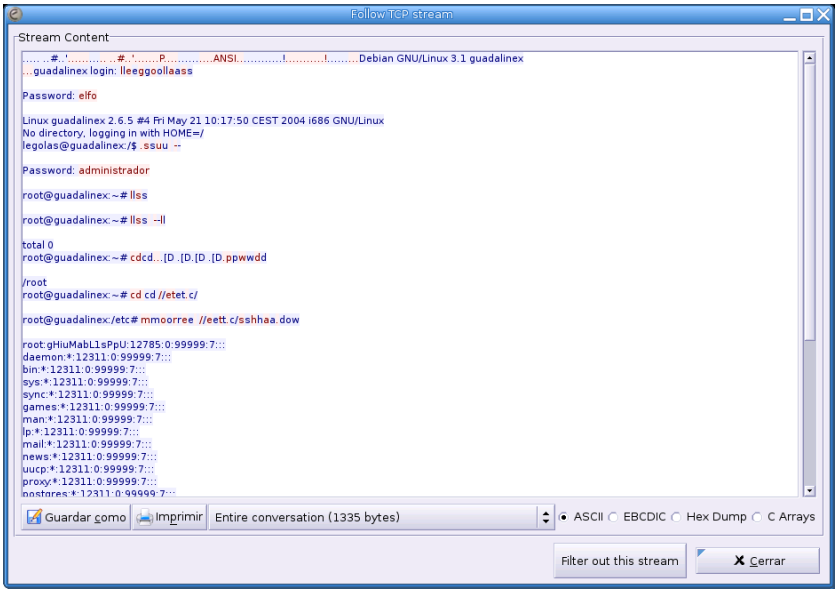
Figura 4.8: Captura de sesión de telnet



Aquí tenemos todos los datos en crudo de la captura, mostrando los paquetes TCP tal y como se transmiten por la red. Sin embargo, es posible hacer que estos datos sean más comprensibles y nos proporcionen datos más jugosos. Para ello utilizaremos el menú **Tools** → **Follow TCP**

Stream.

Figura 4.9: Seguir la trama TCP de la sesión telnet



Como se observa en la imagen anterior, podemos ver exactamente lo que ha estado haciendo el usuario. Hemos obtenido datos sensibles y que vulneran la seguridad del sistema tales como la clave del usuario y la del `root`. Esto último es especialmente grave, ya que nos daría control total sobre el sistema.

Moraleja: no debemos utilizar protocolos no cifrados.

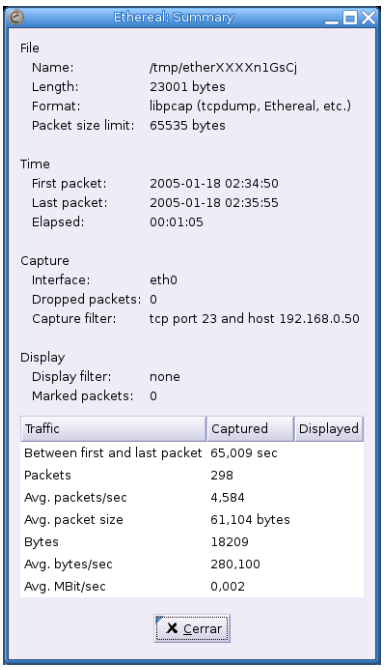
4.5.8. Estadísticas acerca de la captura

Hasta el momento, lo contado sobre Ethereal le proporciona una gran potencia a priori, haciendo su uso muy recomendable en cualquier análisis de los datos que circulan por la red. Sin embargo aún podemos obtener más datos.

Siempre puede ser interesante obtener los datos globales acerca del número de paquetes, tamaño de los mismos, duración de la captura, ... Todos estos datos y algún otro pueden obtenerse con las opciones que presenta el menú **Statistics**.

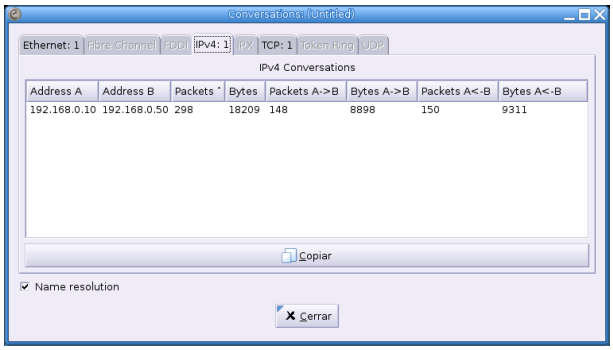
Con la opción **Summary** obtendremos un resumen de la captura realizada.

Figura 4.10: Sumario de la captura



Otra opción que nos puede proporcionar datos interesantes dentro de este menú es **Conversations**. Un ejemplo para entender el concepto de conversación sería el tráfico existente entre dos direcciones IP, este tráfico constituiría una conversación IP.

Figura 4.11: Conversación IPv4



Dentro del menú **Statistics** existen aún más opciones, pero nosotros nos quedaremos aquí, aunque os animamos a investigar sobre el resto de opciones que os proporcionarán datos muy interesantes sobre la captura.

4.6. Monitorizacion de red con EtherApe

El programa `etherape` es una herramienta de monitorizacion gráfica de redes. La principal característica de este software es que es muy intuitivo y sencillo de utilizar. A la hora de monitorizar una red, es un complemento más a otro tipo de programas como el anteriormente visto, `ethereal`.

No tiene la finalidad de un análisis exhaustivo del tráfico de la red como otras herramientas, pero sí ofrece un enorme potencial a la hora de monitorizar en tiempo real el tipo de tráfico existente. Esto último nos va a permitir detectar problemas puntuales de una forma muy rápida y sencilla. Otra funcionalidad importante es obtener estadísticas del tipo de tráfico que tenemos en nuestra red para posibles mejoras de nuestra infraestructura.

Es muy útil a la hora de detectar una máquina que sobrecarga la red, una conexión no deseada o errores de determinados servicios. Y todo ello, de una manera muy sencilla.

4.6.1. Instalación

La instalación de `Etherape` se puede realizar con la propia distribución. Si ya tenemos la distribución sin `etherape` su instalación es similar a la de cualquier otro paquete.

Podemos utilizar la herramienta `apt`:

```
#apt-get install etherape
```

o, en Fedora, el correspondiente paquete rpm:

```
#rpm -Uhv etherape-0.9.1-1.1.fc1.rf.i386.rpm
```

Con esta última opción tendremos que tener en cuenta las dependencias con los paquetes `libpcap`, `gtk+`, `libglade` y la interfaz de escritorio gnome.

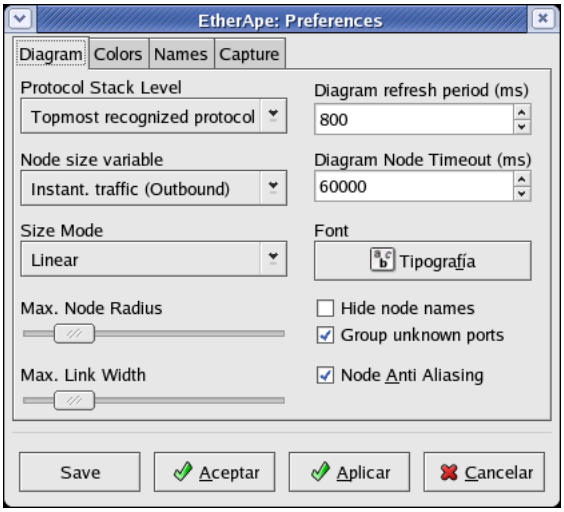
Una vez instalado, nos aparecerá en la opción de Internet del menú principal, listo para utilizarse.

4.6.2. Configuración

Una vez lanzado, ya está listo para monitorizar el tráfico. Según el tráfico que queramos monitorizar, debemos situar la máquina utilizada en un determinado segmento de red.

`Etherape` nos permite en la opción de **preferencias**, variar los parámetros de configuración.

Figura 4.12: Cuadro de preferencias de Etherape



Entre los parámetros que podemos modificar se encuentran:

- Nivel de la pila de protocolos que queremos monitorizar

- Periodo de actualización
- Tráfico y Nodos que se mostrarán en el diagrama y su tamaño
- Aspecto
- Protocolos específicos
- Interfaces y el modo de visualización.

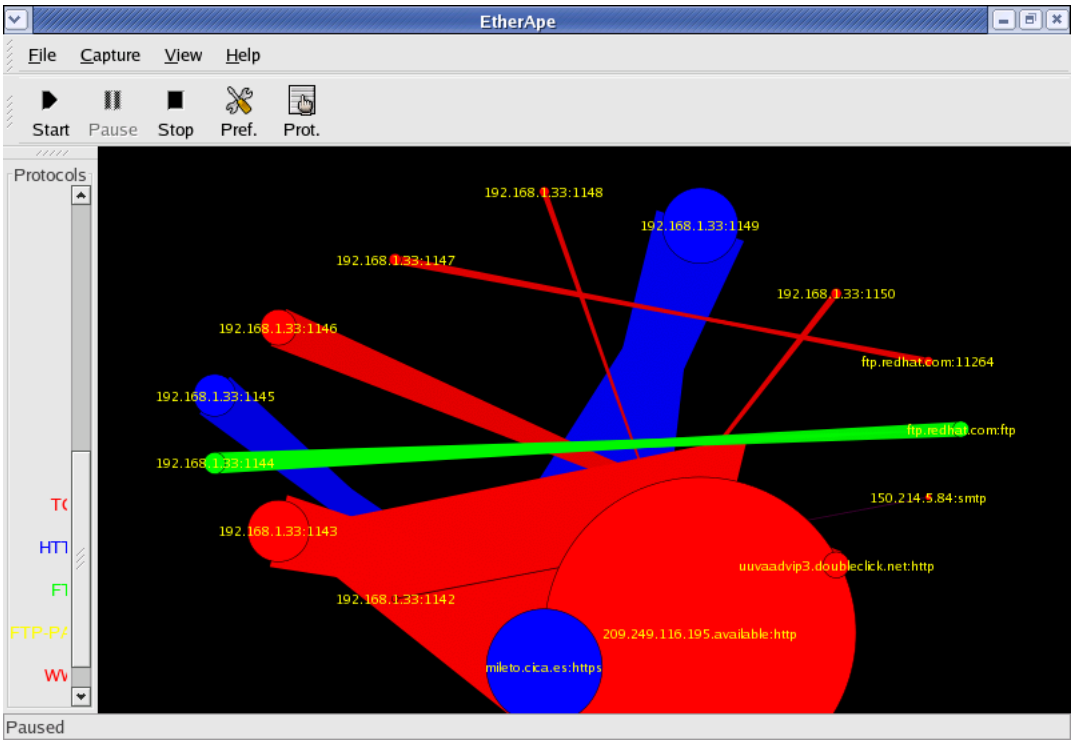
4.6.3. Funcionamiento

El funcionamiento es intuitivo y sencillo. Una vez que comenzamos a monitorizar nos aparecerá cada conexión desde el nodo origen al nodo destino, con un color diferente según cada protocolo y, en función del tráfico, un círculo en torno al nodo destino.

Con los botones de pausa y parada podemos fijar la imagen o resetearla respectivamente. Si dejamos funcionando `Etherape` en un segmento de red con gran cantidad de tráfico, es posible que tengamos que parar de forma periódica o afinar los parámetros para poder ver el tráfico que nos interese.

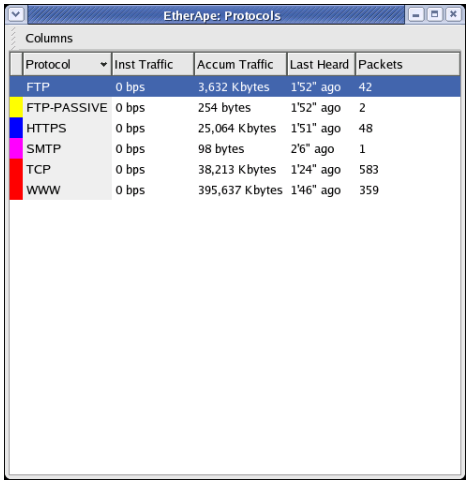
De esta forma, por ejemplo, una máquina que está provocando gran tráfico en la red se podrá detectar de inmediato, incluso saber el destino y el tipo de tráfico (por ejemplo, descarga por ftp a un servidor externo, o de emule).

Figura 4.13: Captura de Etherape



En la opción de **Protocolos**, se nos muestran los diferentes tipos de tráfico junto con las estadísticas totales e instantáneas de cada uno de ellos.

Figura 4.14: Estadísticas de protocolos de Etherape



Capítulo 5

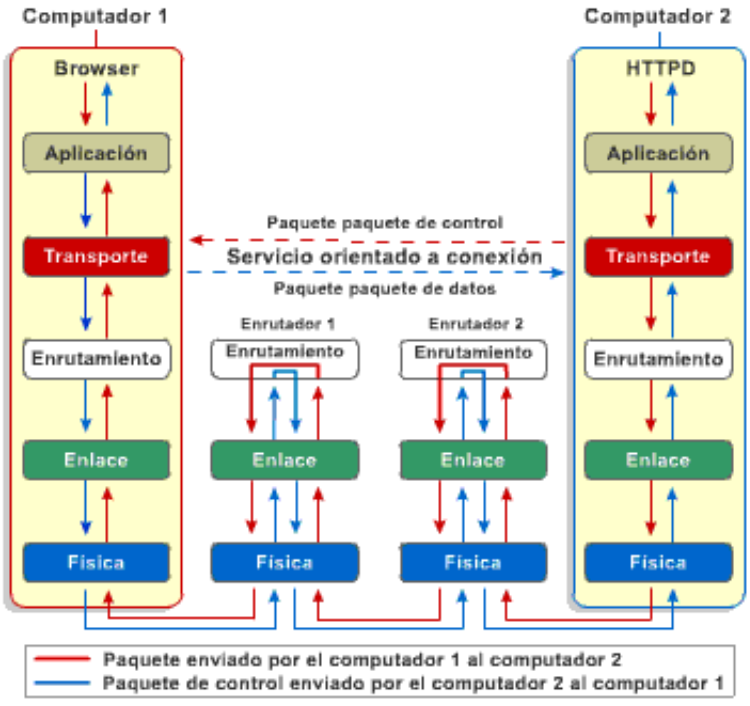
Conectando al mundo exterior

> Siempre me ha fascinado Internet, incluso antes de que existiera.
>
> (*Servidor Apache*, Rich Bowen & Ken Coar, Prentice Hall)

Hasta ahora, hemos visto cómo nuestro sistema se sitúa en nuestra red, nuestro pequeño mundo, sin visibilidad con el exterior. Veamos cómo nos podemos conectar con otras redes y el tan ansiado mundo exterior de Internet.

5.1. Routing o encaminamiento IP

Con el término *Routing* (encaminamiento) se designa al proceso de escoger el camino por el cual van a ser enviados los paquetes IP. El routing sucede cuando el host destino no está en nuestra red IP. Un router es una máquina o un dispositivo que reenvía los paquetes desde una red lógica a otra. A los routers también se les denomina *gateways*[1].



[1]Estrictamente, cuando el sistema que une las diferentes redes trabaja en el nivel de red, se denomina router y cuando trabaja en un nivel superior, se denomina gateway o pasarela.

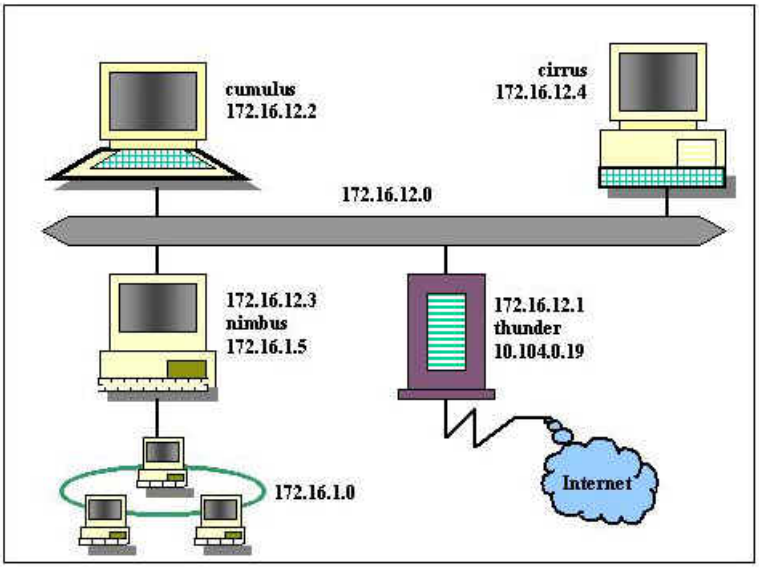
Veamos el proceso que se sigue cuando hay que enviar un paquete de datos en una red TCP/IP.

Para ello, vamos a utilizar la red de ejemplo de la figura siguiente. Veamos los componentes y redes presentes. Hay dos redes internas (que pueden ser la de administración y la de alumnos en un centro educativo): la red 172.16.12.0/24 y la red 172.16.1.0/24.

En la red 172.16.1.0/24[2] hay varias máquinas. Una de ellas es `nimbus`, con dirección IP 172.16.1.5. Otra máquina podría ser la 172.16.1.4, que llamaremos `marte`.

En la red 172.16.12.0/24 se encuentran `cumulus` (172.16.12.2), `cirrus` (172.16.12.4), `nimbus` (172.16.12.3) y `thunder` (172.16.12.1).

Observamos que `nimbus` se encuentra en las dos redes. Dispone de dos interfaces de red y uno de ellos se encuentra en una red y el otro en otra red. También `thunder` pertenece a dos redes, a la 172.16.12.0/24 y posee otro interfaz con la dirección 10.104.0.19, que provee el acceso a Internet.



Veamos los distintos casos que pueden ocurrir.

1. Si la dirección IP de destino está en nuestra misma red lógica, el envío se realiza directamente[3]. Por ejemplo, una comunicación entre `cumulus` (172.16.12.2) y `cirrus` (172.16.12.4).

2. Si la IP de destino se encuentra en otra red, debe tomarse la decisión de por dónde hay que enviarlo. Esta decisión deben tomarla todos los hosts, a los que llega el paquete IP, bien sea nuestro propio host o cualquier router por los que el paquete atraviese. Para tomar la decisión de enrutamiento, la capa IP consulta una tabla de rutas[4] que está almacenada en memoria.

   Mostramos la tabla de rutas de la máquina `nimbus` y veamos qué significa.

Red Destino	Máscara	Gateway	Interfaz
172.16.12.0	255.255.255.0	0.0.0.0	eth0
172.16.1.0	255.255.255.0	0.0.0.0	eth1
0.0.0.0	0.0.0.0	172.16.12.1	eth0

   Para llegar a la red 172.16.12.0/24[5], lo puede hacer a través de su primer interfaz, que se

[2] Aparece dibujada en forma circular, representando que es una red con topología en anillo.

[3] Solamente hay que encontrar mediante ARP la dirección física.

[4] Una tabla de rutas contiene entradas que relacionan la dirección IP buscada y la interface a utilizar para llegar a ella.

[5] Es la red en la que se encuentra con dirección 172.16.12.3

llama eth0[6]. Para llegar a la red 172.16.1.0/24[7], puede hacerlo directamente a través de su interfaz eth1. Para el resto de redes no especificadas explícitamente, lo que se llama *ruta por defecto*, manda los paquetes al gateway 172.16.12.1 y que éste se encargue de tomar las siguientes decisiones de rutas.

Como ejemplo, veamos lo que sucedería en una comunicación entre `cirrus` y `marte`.

La tabla de rutas de la máquina `cirrus` es la siguiente:

Red Destino	Máscara	Gateway	Interfaz
172.16.12.0	255.255.255.0	0.0.0.0	eth0
172.16.1.0	255.255.255.0	172.16.12.3	eth0
0.0.0.0	0.0.0.0	172.16.12.1	eth0

Para llegar a la red 172.16.12.0/255.255.255.0, en la que se encuentra, lo puede hacer directamente a través de su interfaz eth0[8]. Para llegar a la red 172.16.1.0/24, debe hacerlo a través de la dirección 172.16.12.3, por el interfaz eth0, como recoge la segunda línea de la tabla de rutas. Para el resto de rutas, lo hará a través de la dirección 172.16.12.1, que es su ruta por defecto.

Recordemos ahora que nuestro objetivo es la comunicación entre `cirrus` y `marte`. `Cirrus` (172.16.12.4) tiene que comunicarse con `marte` (172.16.1.4). Como no está en su red, mira la tabla de rutas. Para la red en la que se encuentra `marte` (172.16.1.0/24), comprueba que tiene un camino para llegar, que es enviarlo a `nimbus` (172.16.12.3), al que puede llegar directamente porque está en su misma red.

El paquete llega a `nimbus`, al cual se le presenta nuevamente el problema del encaminamiento. En este caso es más fácil, porque `marte` (172.16.1.4) está en una red a la que `nimbus` pertenece (en este caso, 172.16.1.5) y a la que accede por su interfaz eth1. Así el paquete llega a `marte`. Hemos de notar que deben existir las rutas en el camino inverso para que los paquetes de vuelta[9] entre `marte` y `cirrus` puedan llegar.

3. Si no hay una ruta explícita, IP utiliza el gateway por defecto para enviar el paquete al router. Por ejemplo, si `cirrus` (172.16.12.4) quiere conectarse con el servidor web de `mileto.cica.es` (150.214.5.11), mira sus tablas de rutas y no ve una conexión directa con la red 150.214.5.0 a la que pertenece `mileto.cica.es`. Entonces, lo manda por la ruta por defecto, al gateway `thunder` (172.16.12.1), que lo encaminará hacia Internet. En el gateway, otra vez es consultada su tabla de rutas, para seguir buscando un camino al host remoto. Si no existe un camino explícito, el router reenviará otra vez el paquete a su propio gateway por defecto para continuar la cadena y que sea este siguiente router el encargado de repetir el ciclo.

   Cuando vamos pasando por un router y el paquete se envía al siguiente router, esto se denomina un "salto" y se descuenta una unidad del valor del campo TTL[10] del paquete IP. Finalmente, el proceso acaba si el paquete es entregado en el host destino. Si alguna ruta no se encuentra[11], se envía un mensaje de error, mediante el protocolo ICMP al host origen, diciendo que el destino ha sido inalcanzable.

[6] Es la primera (se empieza por 0) interfaz con protocolo ethernet (eth).
[7] En la que tiene dirección 172.16.1.5
[8] De hecho, sólo tiene una tarjeta de red y es el único sitio por el que puede salir.
[9] Lo normal es que cualquier comunicación sea en ambos sentidos con paquetes de envío y de confirmación por parte del destinatario. Si las rutas en sentido inverso no existen o no están bien configuradas, fallará la comunicación.
[10] Tiempo de Vida (*Time To Live*).
[11] Eso ocurre cuando el valor de TTL llega a cero. Es un mecanismo para evitar que los paquetes vaguen eternamente como almas en pena de router en router sin llegar a ningún destino.

5.1.1. Estático y dinámico

El mecanismo de actualización de las tablas de rutas puede ser estático o dinámico. Esto normalmente sólo afecta a los routers, ya que los hosts solamente suelen tener en sus tablas de rutas la red a la que pertenecen y el gateway por defecto.

Los routers estáticos necesitan que las tablas de rutas sean construidas y actualizadas manualmente. Si una ruta cambia, los routers estáticos no se enteran de este cambio automáticamente. En el ejemplo anterior, las tablas de rutas son pequeñas y podríamos haberlas creado de forma estática.

Sin embargo, cuando el número de redes a las que dan acceso los routers es elevado, actualizar las tablas de forma estática sería una labor muy compleja. Para ello, existen los protocolos dinámicos de routing, que hacen que los routers puedan anunciarse los cambios en sus tablas de rutas de una forma automática.

Por ejemplo, son protocolos de routing el *Routing Information Protocol* (RIP) o el *Open Shortest Parh First* (OSPF). Los protocolos de routing periódicamente intercambian rutas a sus redes conocidas a lo largo de los routers dinámicos. Si una ruta cambia, todos los routers dinámicos son informados de dicho cambio.

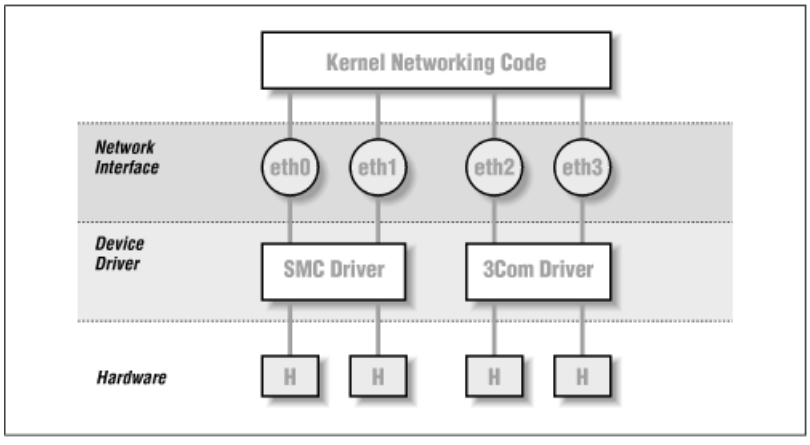
5.2. Vale, pero yo quería un curso de Linux.

Pasemos a ver cómo funciona todo esto de forma práctica en un sistema Linux. Como probablemente ya sabéis, todo en Linux es un fichero, o asimilable a un fichero. Por lo tanto, los interfaces de red, no serán una excepción. La siguiente figura, nos muestra, los elementos tanto hardware como software necesarios para que nuestro sistema linux acceda a la red:

El hardware: puede ser un módem, una tarjeta de red ethernet, tarjeta RDSI...

Los drivers: que son la parte del kernel dependiente del hardware específico. Por ejemplo, para un modelo de tarjeta SMC será diferente al que requiere una tarjeta 3Com.

El interfaz de red dependerá del tipo de red a la que nos conectemos: para una red Ethernet será *eth0*[12], para un módem será *ppp0*, o para el caso del interfaz de loopback, *lo*.



Normalmente, no hay que preocuparse por asuntos de hardware como las direcciones base de Entrada/Salida[13] o las IRQ[14], porque al arrancar el kernel, intenta detectar la tarjeta y los valores que utiliza. Esto se denomina prueba automática[15], que significa que el kernel lee en varias

[12] Si tenemos otra tarjeta de red, sería eth1 y así sucesivamente

[13] En las que el dispositivo físico escribe y lee

[14] Interrupciones que utiliza para comunicarse con el resto del sistema físico, pedir permiso para utilizar recursos comunes.

[15] *Autoprobe*, en inglés

posiciones de memoria y compara los datos que ha encontrado con los que espera de una tarjeta de red en concreto. Solamente en determinados casos[16] habría que especificar valores al kernel para que las encuentre.

En el arranque del equipo[17], podemos observar si ha sido detectado. Por ejemplo:

```
...
Intel(R) PRO/100 Network Driver - version 2.3.18-k1
Copyright (c) 2003 Intel Corporation
PCI: Guessed IRQ 9 for device 00:06.0
PCI: Sharing IRQ 9 with 00:06.1
divert: allocating divert_blk for eth0
e100: selftest OK.
e100: eth0: Intel(R) PRO/100 Network Connection
Hardware receive checksums enabled
cpu cycle saver enabled
...
```

5.2.1. Activando interfaces

Cuando vayamos a conectar nuestra máquina a una red, necesitamos poder ubicarla dentro de la red sin que cause conflictos y se integre con el resto de sistemas que se encuentran en la red. Debemos solicitar a los administradores de la red[18] una serie de valores. Estos valores son: Dirección IP de nuestro sistema, máscara de la red, dirección IP del router[19] de salida y dirección de los servidores DNS. En este ejemplo, utilizaremos 172.26.0.2/255.255.255.0, con el gateway 172.26.0.1 y servidores de DNS 195.235.113.3, 80.58.0.33 y 150.214.4.34.

Hay casos en los que esto, si cabe, puede ser más fácil. Si en nuestra red existe un servidor de DHCP[20], solamente tendremos que decirle a nuestra máquina que coja la configuración de red de forma dinámica mediante este protocolo y estos valores se asignarán automáticamente.

Hay una serie de comandos que se usan con objeto de configurar las interfaces de red e inicializar la tabla de encaminamiento. Esas tareas son ejecutadas generalmente por el script de inicialización de red cada vez que el sistema es arrancado. Las herramientas básicas son `ifconfig`[21], y `route`.

- El comando `ifconfig` se usa para dar acceso al kernel a una interfaz física. Esto incluye la asignación de una dirección IP y la activación de la interfaz. Por activación nos referimos a permitir que el kernel pueda enviar y recibir datagramas IP a través de esa interfaz.
- El comando `route` permite añadir o quitar rutas de la tabla de encaminamiento del kernel.

Con el propósito de que sirva de ejemplo, nos referiremos a la interfaz de *loopback* y a una tarjeta de red ethernet.

Comando ifconfig

El interfaz de bucle local (loopback) es usado para realizar tests, y para simular aplicaciones en red, aun cuando no poseamos hardware de red. Funciona como un circuito cerrado que devuelve cualquier datagrama recibido a la capa de red del host. Siempre hay un dispositivo loopback presente en el kernel, denominado *lo*.

Si nuestra máquina no va a estar conectada a ninguna red, sólo se necesita la dirección de la interfaz de bucle local, que es siempre 127.0.0.1.

[16] Cada vez menos habituales, porque el soporte en el kernel para cada tipo de hardware suele venir ya incorporado, bien en el propio kernel o en módulos que se cargan cuando se necesitan.
[17] También en el fichero `/var/log/messages` o con el comando `dmesg`
[18] Eh, pero si ese soy yo.
[19] O Gateway
[20] Para el mismo propósito serviría bootp, aunque menos utilizado y más antiguo.
[21] Que viene de i*nterface configuration*

Es la primera interfaz en ser activada[22] y el comando que se utiliza es:

```
# ifconfig lo 127.0.0.1
```

que significa, activar la interfaz *lo* con la dirección ip 127.0.0.1

Para ver la configuración de una interfaz, usamos `ifconfig`, pasándole como argumento el nombre de la interfaz:

```
[root@linux entrega04-1]# ifconfig lo
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:2185 errors:0 dropped:0 overruns:0 frame:0
TX packets:2185 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:2125782 (2.0 Mb) TX bytes:2125782 (2.0 Mb)
```

Comentemos la salida obtenida:

- El interfaz *lo* tiene la dirección 127.0.0.1, con máscara de red 255.0.0.0, que se ha asignado automáticamente al ser una dirección de Clase A. Vemos que el estado es UP, es decir, activa.
- El máximo tamaño de unidad de transferencia será de 16.436 bits, quiere decir que si el protocolo superior necesita transmitir paquetes de tamaños superiores, deberá fragmentarlos en trozos más pequeños.
- La métrica es una indicación de la distancia necesaria para llegar a dicha red, como nuestra máquina está directamente en ella es 1.
- Nos dice también el número de paquetes transmitidos (TX) y recibidos (RX) desde que se activó la interfaz, así como el número de errores y colisiones.

Lo siguiente es comprobar que todo funciona como es debido, por ejemplo usando el comando `ping`. Esta orden se usa para verificar que una dirección dada es accesible y para medir el retraso entre el envío de un datagrama y su recepción de vuelta. Este tiempo es conocido como tiempo de ida y vuelta[23].

```
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=0.4 ms ^C
--- localhost ping statistics --- 3 packets transmitted, 3 packets received, 0%
packet loss round-trip min/avg/max = 0.4/0.4/0.4 ms
```

Cuando se ejecuta `ping` según se muestra aquí, la emisión de paquetes continúa a menos que sea interrumpida por el usuario. Cuando se pulsa **Ctrl-C**, interrumpimos el comando.

Este ejemplo muestra que los paquetes dirigidos a la máquina 127.0.0.1 están siendo entregados correctamente y la respuesta a `ping` es recibida de forma casi instantánea (0.4 milisegundos). Esto significa que ha establecido con éxito su primera interfaz de red.

La configuración de una interfaz Ethernet es similar a la de la interfaz de bucle local. En nuestro ejemplo:

```
# ifconfig eth0 172.26.0.2 netmask 255.255.255.0
```

Esto asigna a la interfaz *eth0* la dirección IP 172.26.0.2 con la máscara de red 255.255.255.0:

```
[root@linux entrega04-1]# ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:D0:59:0F:05:1C
```

[22] Lo normal es que ya se encuentre activada en nuestro sistema.

[23] ¿Por qué será que me suena al ping-pong?

```
inet addr:172.26.0.2 Bcast:172.26.0.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:127 errors:0 dropped:0 overruns:0 frame:0
TX packets:138 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:99853 (97.5 Kb) TX bytes:20272 (19.7 Kb)
Interrupt:9 Base address:0x9000 Memory:81200000-81200038
```

En esta salida, se nos muestra la dirección MAC correspondiente a la tarjeta `Ethernet HWaddr 00:D0:59:0F:05:1C`. Se ha fijado la dirección de broadcast automáticamente (que se puede obtener sabiendo la dirección IP y la máscara de red). Se fija la unidad de transferencia de mensajes (MTU) a un máximo de 1.500 bytes. Igualmente, podemos hacer un ping para comprobar que se encuentra operativa.

5.2.2. Estableciendo rutas

Debemos indicar en la tabla de encaminamiento qué redes son accesibles mediante los interfaces lo y eth0. Lo hacemos con los comandos:

```
# route add -net 127.0.0.0 netmask 255..0.0.0 lo
# route add -net 172.16.0.0 netmask 255.255.255.0 eth0
```

Estamos indicando que en nuestra máquina, a la red 172.26.0.0/24[24], se llega a través de la interfaz eth0. Y a la red 127.0.0.0 llegamos a través del interfaz lo. La opción `add` del comando `route` añade rutas, con la opción `del`, las eliminamos y con la opción `-n` nos las mostraría.

Veámoslo con el siguiente comando:

```
[root@linux entrega05-1]# route -n
Kernel IP routing table
Destination    Gateway    Genmask    Flags Metric Ref Use Iface
172.26.0.0     0.0.0.0  255.255.255.0  U     0     0   0   eth0
127.0.0.0      0.0.0.0  255.0.0.0      U     0     0   0   lo
```

El comando route nos dice que la red 172.26.0.0/255.255.255.0 está activa (UP) y llegamos a ella a través de la interfaz eth0. Igualmente sería para la red de loopback.

Hasta aquí nos hemos ubicado en la red local y podemos comunicarnos con otras máquinas de la red. Nos falta indicar un camino de salida para alcanzar otras redes. Esto se realiza indicando un gateway.

```
#route add default gw 172.26.0.1
```

Veamos cómo se ha modificado la tabla de rutas.

```
[root@linux entrega05-1]# route -n
Kernel IP routing table
Destination     Gateway    Genmask      Flags Metric Ref Use Iface
172.26.0.0      0.0.0.0    255.255.255.0   U      0    0   0   eth0
127.0.0.0       0.0.0.0    255.0.0.0       U      0    0   0   lo
0.0.0.0         172.26.0.1 0.0.0.0        UG      0    0   0   eth0
```

Se ha añadido una ruta nueva a la red (0.0.0.0), que significa "cualquier red no especificada anteriormente". A ella se llega por medio del gateway 172.26.0.1, que se encuentra accesible por la interfaz eth0. O sea, cualquier paquete que no vaya dirigido a la red 172.26.0.0/24 ni a la red 127.0.0.0/8, se mandará a la dirección 172.26.0.1 para que "se ocupe de ello", que esa se supone que es la función de un router. Es una bonita forma de liberarnos del trabajo[25].

[24] El 24 es más cómodo de teclear que 255.255.255.0

[25] Si esto pudiera aplicarse en más casos ...

5.2.3. Resolución de nombres.

Hasta ahora, de forma premeditada, hemos obviado las direcciones simbólicas (de la forma `thales.cica.es`) y hemos trabajado exclusivamente con direcciones numéricas (172.26.0.2). Veamos cómo funciona en un host la resolución de direcciones simbólicas (nombres) a numéricas.

En una red pequeña, podemos mantener tablas de asignación de nombres a direcciones IP numéricas. Esta información se mantiene en un fichero llamado `/etc/hosts`. Cuando se añaden o se eliminan puestos, o se reasignan direcciones, lo que se hace es actualizar el fichero `/etc/hosts` en todos los puestos. Obviamente, esto solamente funciona en redes muy pequeñas.

```
$more /etc/hosts
127.0.0.1 localhost.localdomain localhost
172.26.0.2 linux linux.midominio.org
```

Para resolver este problema, hacemos uso de los servidores de nombre DNS, a los cuales les preguntamos por una dirección simbólica y nos devuelven la dirección numérica. Esta es una de las labores de los servidores de nombres, que ampliaremos en la segunda entrega, donde hablaremos del sistema BIND (*Berkeley Internet Name Domain*) y el servidor *named*.

El fichero `/etc/nsswitch.conf` es el que guía qué sistema utilizaremos y en qué orden. Por ejemplo, si la entrada de este fichero es la siguiente:

```
hosts: files dns
```

nos está indicando que primero vaya a resolver buscando en el fichero `/etc/hosts` (opción *files*) y luego vaya al sistema de DNS (opción *dns*).

La configuración del sistema de búsqueda para el DNS se encuentra en el fichero `/etc/resolv.conf`.

```
[root@linux entrega05-1]# more /etc/resolv.conf
nameserver 195.235.113.3
nameserver 80.58.0.33
nameserver 150.214.4.34
```

En este ejemplo, se encuentran indicados los servidores de nombres a utilizar en el orden de preferencia, del primero al último.

5.3. Configuración gráfica

5.3.1. Con Fedora

Creación de las interfaces de red.

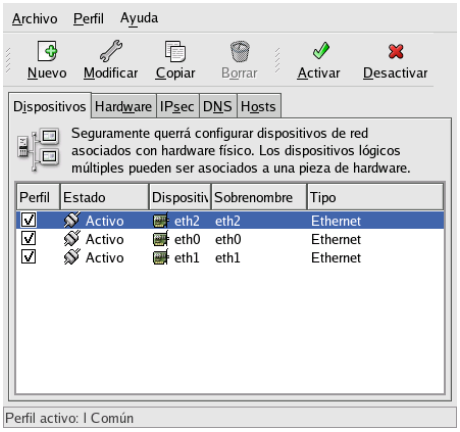
Para un sistema Fedora en modo gráfico, podemos lanzar desde Gnome

 **→Configuración del Sistema→Red**

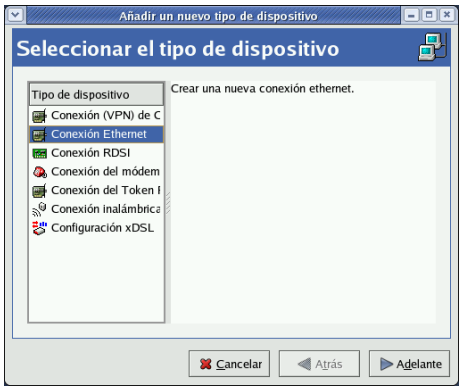
o directamente desde una xterm[26]

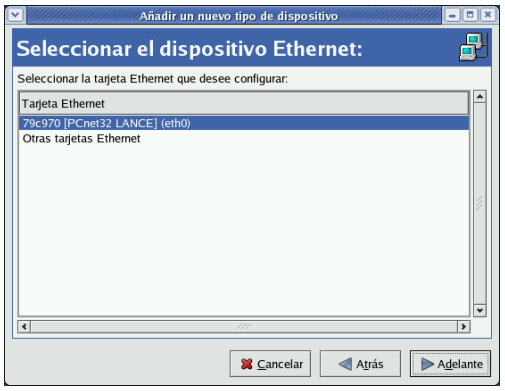
```
# neat &
```

[26] O con `system-config-network`. La captura gráfica no se corresponde con la que aparecería por defecto.

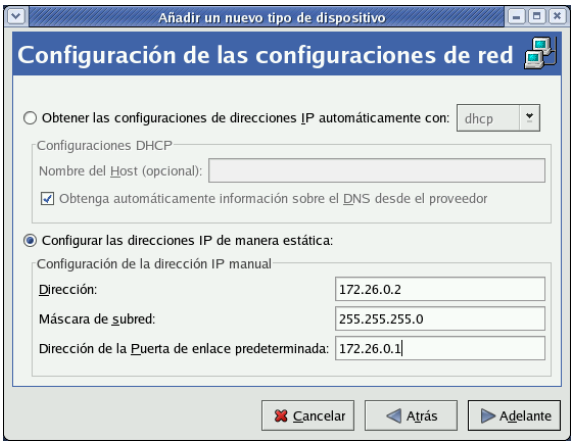


Lo usual es que la tarjeta haya sido detectada y configurada en el arranque/instalación y desde esta ventana podremos editarla. Si no es así, pulsando sobre **Nuevo** podemos configurar nuestra nueva interfaz de red de área local. Seleccionamos **Conexión Ethernet** y debemos elegir la tarjeta correspondiente





Una vez definidos correctamente los valores hardware de nuestra tarjeta, tendremos que configurar los dispositivos de red asociados con el hardware físico.

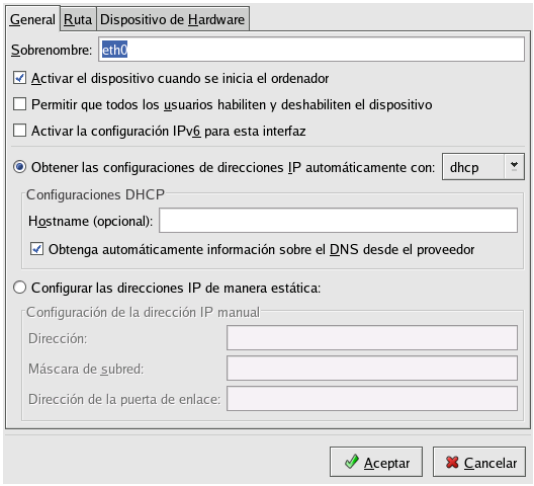


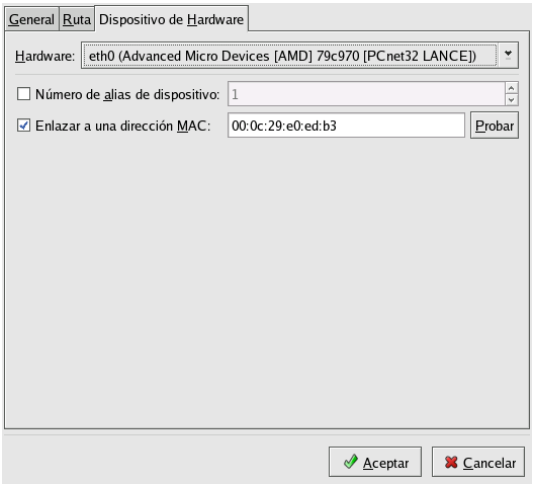
Tendremos la posibilidad de permitir que la configuración se obtenga de un servidor de alguno de estos protocolos (DHCP, BOOTP) que se la proporcionará al arrancar, o bien, si marcamos la

casilla **Configurar las direcciones IP de manera estática** podremos introducir la dirección IP (172.26.0.2), la máscara de red (255.255.255.0) y la **Dirección de la Puerta de enlace predeterminada** (172.26.0.1).[27]

Si tenemos un router ADSL u otro Linux que hace de pasarela, es el momento de poner aquí su dirección IP para que podamos salir al exterior. Si no, debemos dejar esta casilla en blanco.

Si en la ventana principal de **neat** hacemos doble clic sobre un dispositivo ya instalado en el sistema (o pulsamos sobre el botón **Modificar**) podemos, además de cambiar las opciones anteriores, acceder a más posibilidades de configuración[28]:





Desde la pestaña **Dispositivo de Hardware,** se nos permitirá asociar nuestro dispositivo a una determinada dirección MAC, o asignar un alias a nuestro interfaz de red (permite la posibilidad de que un interfaz de red tenga varias direcciones IP distintas).

Si tenemos que añadir rutas adicionales[29], seleccionamos la pestaña **Ruta**. Para **Añadir** una nueva ruta, debemos especificar la dirección de la red, su máscara y la puerta de enlace para llegar a ella.[30] Así de fácil hemos configurado nuestra interfaz de red Ethernet, sin tener que utilizar los comandos de bajo nivel `ifconfig` y `route` vistos anteriormente.

Configuración del sistema DNS

Pulsamos en la pestaña **DNS** e introducimos las direcciones IP de nuestros servidores de nombres. Se trata de rellenar los datos necesarios en estos campos. Necesitamos conocer el nombre de nuestro servidor de Internet, que lo escribiremos en el campo **Nombre del host** (no es necesario) y los números DNS de nuestros servidores de nombres. En el caso de la red de ejemplo con la que estamos trabajando, escribiríamos como DNS 195.235.113.3, 80.58.0.33 y 150.214.4.34, que serían los DNS primario, secundario y terciario. Quedaría:

[27] Recordemos que el encaminamiento IP es el proceso por el que una máquina decide por dónde dirigir un paquete IP que haya recibido.

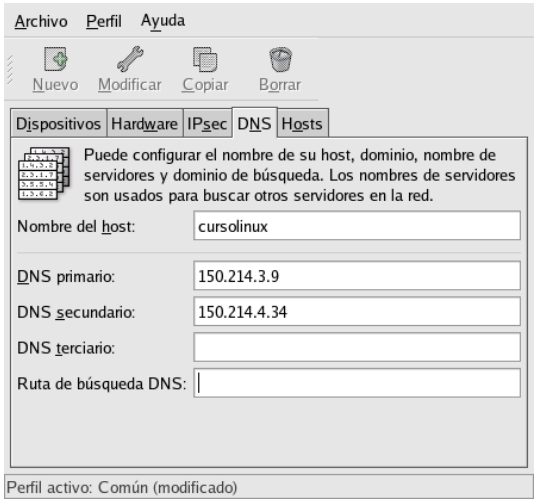
[28] Normalmente seleccionaremos la opción de activar el interfaz en el arranque (**Activar dispositivo cuando arranca el sistema**), pero no permitiremos que cualquier usuario pueda desactivarla en un sistema en el que puede haber usuarios que no son administradores del sistema (**Permitir a todos los usuarios ...**).

[29] No es lo normal. En el 99 % de los casos debería bastarnos con la ruta de nuestra red local y la Puerta de Enlace predeterminada.

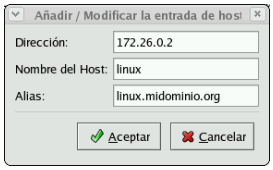
[30] Para comprobar que funciona correctamente podemos hacer `ping` a una máquina de fuera de nuestra red y ver si realmente los paquetes pueden salir al exterior.

```
$ping 150.214.4.34
```

Si nos llegan los paquetes de vuelta, estupendo. En caso contrario, debemos asegurarnos de que hemos realizado correctamente la configuración de las rutas, y por si acaso, reiniciando nuestra máquina.



Si deseamos resolver nombres localmente (modificar el fichero `/etc/hosts` en modo gráfico) pulsaremos sobre la pestaña **Hosts**. Podemos modificar las entradas de ese fichero o añadir más pulsando sobre **Nuevo**



Llegados a este punto, pulsamos **Activamos** el interfaz y cerramos. Para comprobar que realmente resolvemos los nombres, podemos hacer

```
$ping thales.cica.es
```

Lo primero que hace la máquina será traducir el nombre `thales.cica.es` a su dirección IP que es con la que trabajan las tarjetas de red. Después mandará los paquetes a la dirección indicada, a través del router si no estamos en la misma red.

A modo de resumen

Para un sistema RedHat o Fedora, la configuración que hemos hecho se guardaría en el directorio `/etc/sysconfig/`, que contiene los ficheros que leerá el sistema al arrancar y activar la red.

El fichero `/etc/sysconfig/network` contendrá algo parecido a esto[31]:

```
NETWORKING=yes
HOSTNAME="linux"
FORWARD_IPV4="no"
GATEWAYDEV="eth0"
GATEWAY="172.26.0.1"
```

El fichero `/etc/sysconfig/network-scripts/ifcfg-eth0` contendrá la configuración para la tarjeta de red (`eth0`)[32]:

[31] Para conocer las opciones de este fichero se puede consultar `/usr/share/doc/initscript-x.x.x/sysconfig.txt`

[32] Además de poder asignar la IP al interfaz de forma estática (`static`), podemos optar por asignar a la variable BOOTPROTO los valores `dchp` o `bootp`.

```
DEVICE=eth0
BOOTPROTO=static
IPADDR=172.26.0.2
NETMASK=255.255.255.0
ONBOOT=yes
TYPE=Ethernet
USERCTL=no
NETWORK=172.26.0.0
BROADCAST=172.26.0.255
PEERDNS=no
GATEWAY=172.26.0.1
```

Si modificamos con un editor alguno de estos ficheros y deseamos releer la configuración ejecutaremos

```
# /etc/rc.d/init.d/network reload
```

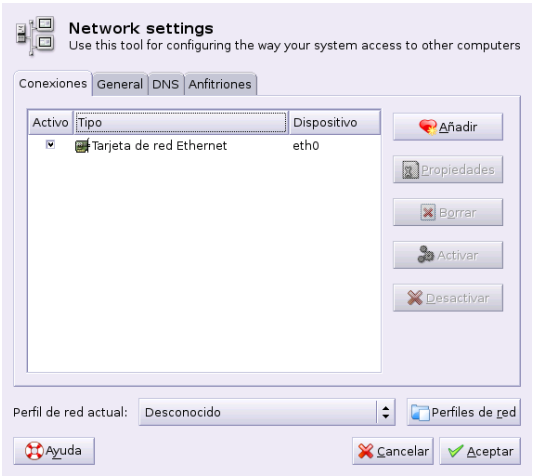
5.3.2. Con Guadalinex (Debian)

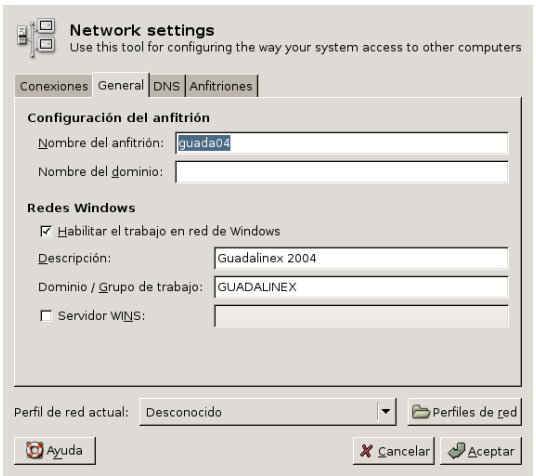
Creación de las interfaces de red.

Para un sistema Guadalinex podemos configurar la red en modo gráfico lanzando desde Gnome  **Aplicaciones→Configuración→Sistema→Red**

o directamente desde una **xterm**:

```
# network-admin &
```





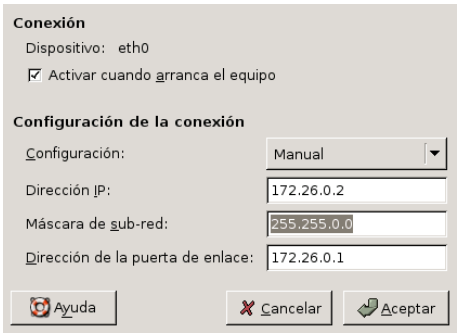
Si pulsamos sobre la pestaña **Conexiones** podremos optar por configurar nuestra red. Lo usual es que la tarjeta haya sido detectada y configurada en el arranque/instalación y desde esta ventana podremos editarla.

Si no es así, pulsando sobre **Añadir** podemos configurar nuestra nueva interfaz de red de área local. Seleccionamos **Conexión Ethernet** y debemos elegir la tarjeta correspondiente

Tendremos la posibilidad de permitir que la configuración se obtenga de un servidor de alguno de estos protocolos (DHCP, BOOTP) que se la proporcionará al arrancar, o bien, si optamos por mantener la **Configuración Manual** podremos introducir la dirección IP (172.26.0.2), la máscara de red (255.255.255.0) y la **Dirección de la Puerta de enlace** (172.26.0.1)[33].

Si tenemos un router ADSL u otro Linux que hace de pasarela, es el momento de poner aquí su dirección IP para que podamos salir al exterior. Si no, debemos dejar esta casilla en blanco.

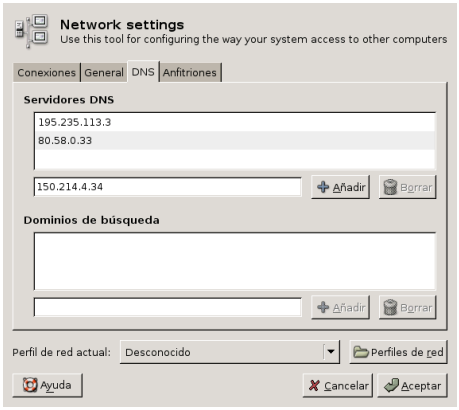
Si en la ventana principal del programa nos situamos sobre un dispositivo ya instalado en el sistema y pulsamos sobre el botón **Propiedades,** podemos cambiar las opciones anteriores o acceder a otras posibilidades de configuración. Deberíamos dejar marcada la opción de activar el interfaz en el arranque (**Activar cuando arranca la computadora**):



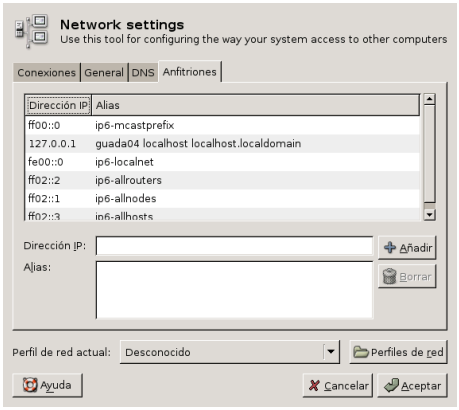
Configuración del sistema DNS

Pulsamos en la pestaña **DNS** e introducimos las IP de nuestros servidores de nombres. Se trata de rellenar los datos necesarios en estos campos, necesitamos conocer el nombre de nuestro servidor de Internet, que lo escribiremos en el campo **Nombre del dominio** (no es necesario) y los números DNS de nuestros servidores de nombres. En el caso de la red de ejemplo con la que estamos trabajando escribiríamos como DNS 195.235.113.3, 80.58.0.33 y 150.214.4.34, que serían los DNS primario, secundario y terciario. Quedaría:

[33]El encaminamiento IP es el proceso por el que una máquina decide por dónde dirigir un paquete IP que haya recibido.



Si deseamos resolver nombres localmente (modificar el fichero `/etc/hosts` en modo gráfico) pulsaremos sobre la pestaña: **Anfitriones**. Podemos modificar las entradas de ese fichero o añadir más pulsando sobre **Añadir**



Llegados a este punto, después de **Aceptar** y volver a la pestaña **General**, nos garantizamos que esté activo el interfaz verificando que esté marcada la casilla **Estado** y cerramos. Para comprobar que realmente resolvemos los nombres, podemos hacer

`$ping thales.cica.es`

Lo primero que hace la máquina será traducir el nombre `thales.cica.es` a su dirección IP que es con la que trabajan las tarjetas de red. Después mandará los paquetes a la dirección indicada, a través del router si no estamos en la misma red.

A modo de resumen

Para un sistema Guadalinex, la configuración que hemos hecho se guardaría en el directorio `/etc/network/`, contiene los ficheros que leerá el sistema al arrancar y activar la red. El contenido del fichero `/etc/network/interfaces` será similar a[34]:

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

[34]Para conocer las opciones de este fichero
`$man interfaces`
El fichero `/etc/hostname` contendrá el nombre de la máquina.

```
auto eth0
iface eth0 inet static
        name Tarjeta de red Ethernet
        address 172.26.0.2
        netmask 255.255.255.0
        broadcast 172.26.0.255
        network 172.26.0.0
        gateway 172.26.0.1
```

Listado 5.1: /etc/network/interfaces

Si modificamos con un editor este fichero y deseamos releer la configuración ejecutaremos

```
# /etc/init.d/networking restart
```

5.4. Conexión a Internet: RTB y ADSL.

Linux e Internet van cogidos de la mano, sin Internet Linux posiblemente estaría "arrumbado" en el cajón de alguna universidad y no sería lo que es hoy. En este apartado vamos a configurar (en modo gráfico[35]) la conexión a Internet de nuestro equipo. Se va a realizar la conexión intentando que sea lo más estándar y guiada posible.

5.4.1. Conexión con módem

Supondremos que nos asignan la dirección de forma dinámica, como ocurre con la mayoría de proveedores de Internet.

Antes de proceder a realizar la conexión a Internet usando un módem necesitamos una serie de datos:

1. Módem:
   - *a*) Tipo de módem, puerto serie[36] al que está conectado.
   - *b*) IRQ y direcciones de E/S.
   - *c*) Velocidad del módem.
2. Datos relativos al proveedor (entre paréntesis los que usaremos de ejemplo[37]):
   - *a*) Dominio de acceso (cica.es)
   - *b*) Número de teléfono de acceso (950542000)
   - *c*) Nombre de usuario (codigo_centro@cica)
   - *d*) Contraseña (********)
   - *e*) Método de autentificación (CHAP o PAP)
   - *f*) Dirección IP del servidor de nombres de dominio (DNS: 195.235.113.3, 80.58.0.33 y 150.214.4.34).

[35] Si se desea realizar la conexión a Internet con script se puede consultar el curso de Linux Thales-CICA 2003, disponible en `http://www.iesmurgi.org`, sección de descargas.

[36] Si no lo sabemos y tenemos Windows instalado, podemos usarlo para conocerlo.

[37] En general sólo necesitaremos los 4 primeros

Configuración del módem.

Lo primero que tenemos que conocer antes de iniciar el proceso de conexión a internet es saber si nuestro módem funcionará con Linux. Además, puede que necesitemos saber a qué puerto serie está conectado.

En Linux todo son ficheros, y los puertos serie también. Así, cada "fichero" `/dev/ttySx` se corresponde con el puerto de comunicaciones del MS-DOS

Linux	MS-DOS
ttyS0	COM1
ttyS1	COM2
ttyS2	COM3
ttyS3	COM4

El mejor sitio para saber si nuestro módem funciona con Linux:

- `http://freewebhosting.hostdepartment.com/g/gromitkc/winmodem.html`, o en castellano
- `http://freewebhosting.hostdepartment.com/g/gromitkc/winmodem_es.html`

Una página en la que encontrar información si tenemos problemas con el módem:

- `http://wiki.escomposlinux.org/Escomposlinux/EscomposlinuxHardware`

En líneas generales, para los distintos tipos de módem podemos establecer que:

Módem Internos:

Si nuestro módem no es PCI no debería haber ningún problema. Pero la mayoría de ellos son:

Winmódem:

La mayoría de los módem internos PCI no son módem completos y sólo son módem "software". Han aparecido drivers para que algunos modelos de pseudomódem puedan funcionar bajo Linux. Para saber si el nuestro es uno de los que están soportados lo mejor es mirar en las páginas

- Linux Winmodem Support `http://linmodems.org/`
- Winmodems no son modems `http://freewebhosting.hostdepartment.com/g/gromitkc/pci_list.html`
- Linmodem-HOWTO `http://www.tldp.org/HOWTO/Linmodem-HOWTO.html`

En general, y aunque estén soportados, no son fáciles de configurar y nuestra experiencia es que incluso los soportados dan bastantes problemas.

De Linux Winmodem Support `http://linmodems.org/` podemos bajarnos la utilidad `ScanModem`, con ella podemos testear nuetro WinModem y con la información obtenida intentar configurarlo. Al estar traducida su forma de uso en `http://linmodems.technion.ac.il/linmodems_support_sp.htm` os remitimos a esa Web en el caso de que necesitéis usarla.

Módem Externos:

Al puerto serie: En general no presentan ningún problema, se autodetectan.

USB Muchos son winmodems, aunque están mejor soportados que sus "hermanos" internos o PCI. Para saber si nuestro modelo está soportado, podemos revisar `http://freewebhosting.hostdepartment.com/g/gromitkc/usblist.html`. Si nuestro módem es de este tipo y al ejecutar (como root)

```
#modprobe cdc-adm
#dmesg
```

obtenemos de salida algo similar a:

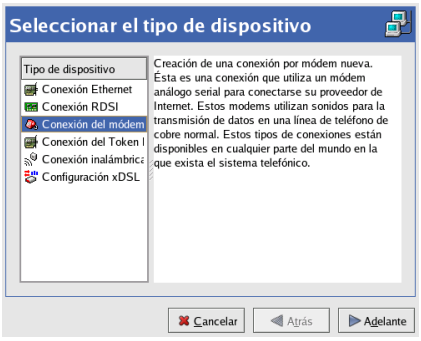
```
KERNEL: usb.c: ttyACM0: USB ACM device
```

es que nuestro núcleo lo detecta y podremos trabajar con él como si de un módem serie se tratase.[38]. ¿Y si no sale nada?, casi que mejor pensar en otro modelo.

Conexión con Fedora

La conexión a Internet en modo gráfico es un "juego de niños". Sólo hay que ejecutar (como root)

→Herramientas del sistema→Asistente de configuración de Internet[39]



En la ventana que aparece seleccionamos **Conexión del módem** y, al pulsar sobre el botón **Adelante,** comenzará el proceso de autodetección del módem[40].

Puede ocurrir que no detecte el módem, en este caso aparecerá un mensaje de advertencia. Si nos ocurre esto, podemos usar el programa `minicom` para, una vez seguros de que funciona bien y del puerto en el que está conectado, optar por introducirlo de forma manual.

El paso siguiente es configurar la conexión de acceso a partir de los datos de nuestro servidor. En esta pantalla introduciremos el número de teléfono del nodo local al que llamar (por ejemplo

[38] En este caso hay que comprobar si nuestro dispositivo es `/dev/usb/ttyACM0` o `/dev/ttyACM0`

[39] Equivale a ejecutar desde un terminal el comando:

```
#internet-druid
```

o bien

```
#system-config-network-druid
```

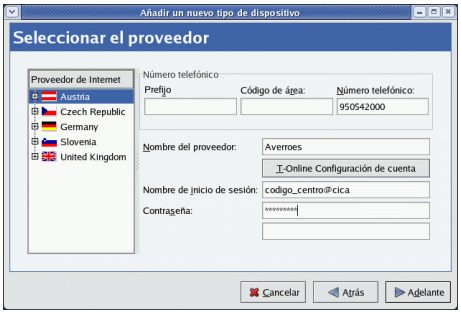
o el ya visto

```
#neat
```

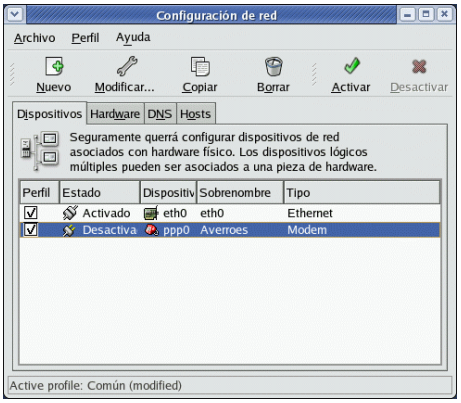
¡Todos los caminos conducen a Roma!

[40] El programa `wvdial` es el encargado de "chequear" los puertos.

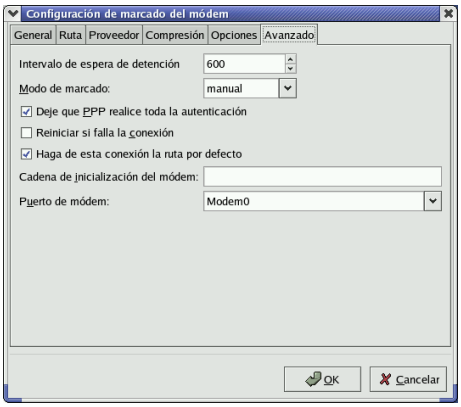
950542000), el nombre que le vamos a dar a esta conexión, el nombre de usuario de nuestra conexión a Internet y la contraseña de acceso. Después se nos preguntará si obtenemos la IP de forma dinámica (es lo habitual), en general dejaremos las opciones por defecto y **Adelante**

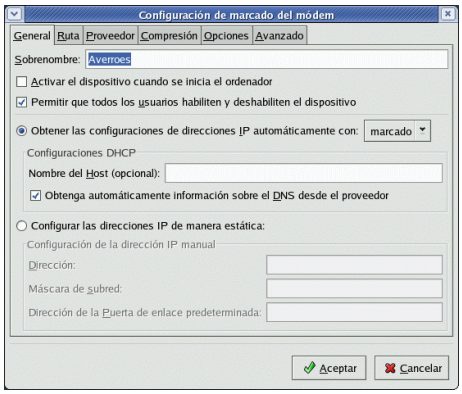


Si continuamos, aparecerá una ventana solicitándonos confirmación sobre los datos introducidos y, por último, la que indica que ya hemos terminado el proceso



Mediante el botón **Nuevo** podemos añadir cuantas cuentas de acceso a Internet tengamos a nuestra disposición. El resto son de uso común y sólo comentaremos un poco las posibilidades que se nos brindan al ejecutar **Modificar**





De esta ventana lo interesante es que permite que activemos la posibilidad de que todos los usuarios puedan conectar a internet (opción por defecto) y que debemos dejar activa la casilla de obtener automáticamente información sobre el DNS desde el proveedor[41].

[41] Si optamos por hacerlo manualmente: véase 5.3.1 en la página 66

Desde la pestaña correspondiente a **Proveedor** podemos modificar los datos relativos a nombre de usuario, contraseña, etc. Por último, en la ventana a la que se accede en **Avanzado** hay que dejar activas las opciones que aparecen marcadas y pulsar **OK**

Ahora vamos a testear que nuestra conexión funciona como debe. En la ventana final de configuración hay que pulsar sobre **Activar** (se nos pedirá confirmación de guardar los cambios) y se inicia el proceso de activación del dispositivo de red *ppp0*. Si todo ha ido bien, en el campo **Estado** aparecerá la palabra **Activar**.

Listo, ya podemos comprobar con Mozilla ( ) o el navegador que más nos guste que podemos navegar por la red. Para desconectar de internet sólo tenemos que **Desactivar** el dispositivo.

Cuando queramos conectarnos usaremos el programa `system-control-network` o con menús:

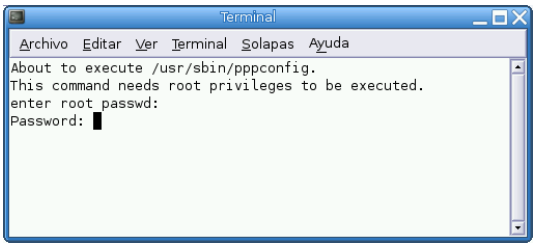
→Herramientas del sistema→Control de dispositivos de red

Desde esta ventana podemos Activar/Desactivar la conexión a internet así como cualquier otro dispositivo de red de nuestra máquina.

Conexión con Guadalinex

La conexión a Internet es sencilla usando el programa `pppconfig`[42]. Podemos acceder a la aplicación de dos modos diferentes:

- **Aplicaciones→Menú Debian→Aplicaciones→Sistema→Administración→pppconfig**

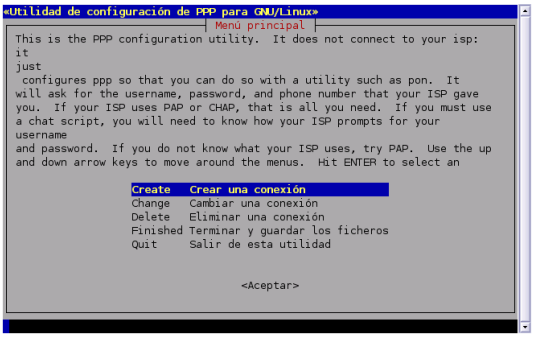


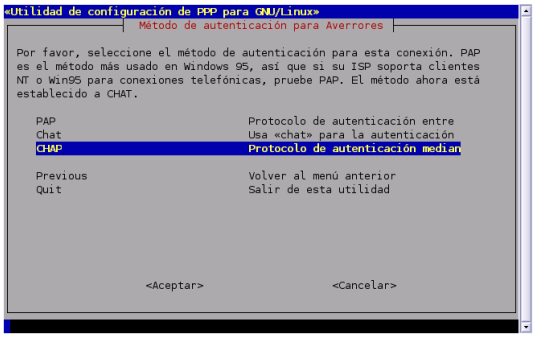
- Desde un terminal de texto **xterm** escribimos (como root):

```
#pppconfig &
```

Si optamos por crear una cuenta nueva, la primera pregunta que nos va a hacer es el nombre que le vamos a dar a esta conexión

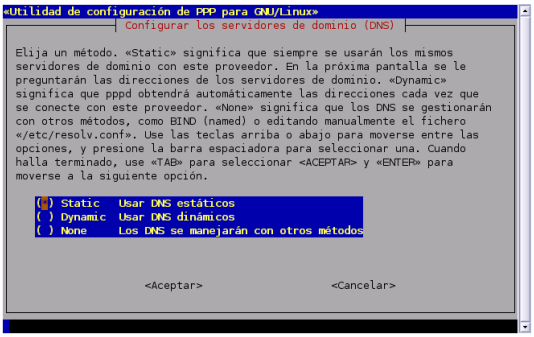
[42] Es mejor que el resto de utilidades gráficas de Guadalinex, es igual de fácil trabajar con ella pero su fiabilidad es mayor.

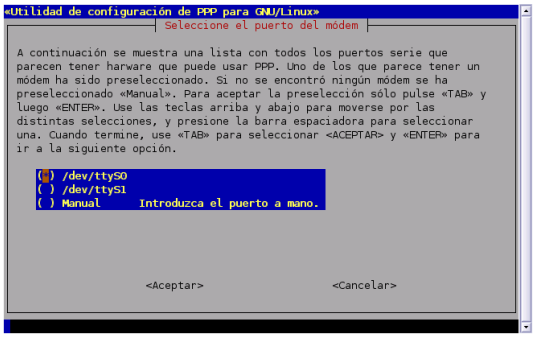




Una vez que "manifestamos" nuestro acuerdo tenemos que introducir las IPs de los servidores de nombres (uno a uno) y el modo de autentificación. Primero deberíamos probar con CHAP y, si no funciona, intentarlo con PAP.

A continuación debemos optar por la forma en que nuestro servidor de acceso nos va a facilitar la IP de los servidores de nombres: de forma estática o dinámica. Vamos a suponer que lo hace de forma estática aunque en la mayoría de los servidores actuales podríamos optar por la segunda opción (dinámica) y después escribiremos el nombre de usuario de nuestra conexión a Internet.





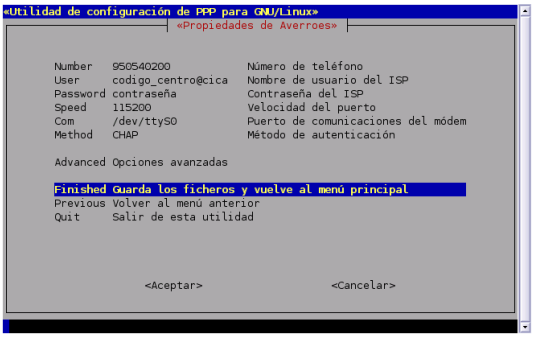
Casi hemos terminado, ahora introducimos la contraseña de acceso. Lo siguiente es optar por seleccionar la velocidad entre el módem y el puerto serie (no es la velocidad del módem; si el ordenador es antiguo quizá haya que poner 57600).

Seleccionamos ahora el tipo de marcado, mejor por "tonos"[43], y pasamos a introducir el número de teléfono del nodo local al que llamar (en el ejemplo 950542000)

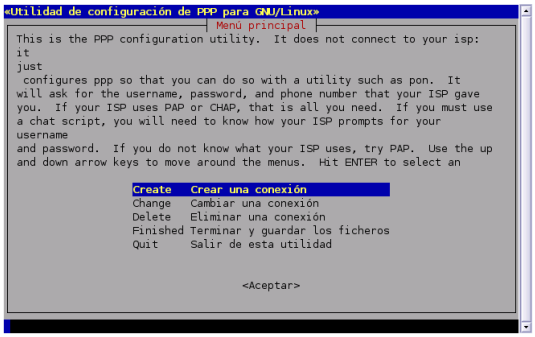
Llega el momento de "la verdad", el de la autodetección del módem. Si lo autodetecta, felicidades: ya es "coser y cantar". Si no es así tendremos que intentar configuarlo de forma manual y, para eso, hay que echar mano de las páginas antes comentadas.

Ya sólo nos falta comprobar que los datos introducidos son correctos, arreglar aquello que esté mal y guardar los datos de esta conexión a Internet. Si todos los datos son correctos seleccionaremos la opción [**Finished**] y [**Aceptar**].

[43] **Marcación decádica por pulsos**
De Wikipedia, la enciclopedia libre.
"La marcación decádica por pulsos consiste en el envío por el teléfono de la información numérica, en forma de pulsos, a la central telefónica automática para que ésta le conecte con el teléfono deseado.
Los pulsos los genera el teléfono mediante un dispositivo mecánico denominado disco de marcar, el cual consiste en un disco giratorio provisto de diez agujeros, de aqui lo de decádica, numerados del 0 al 9.
La marcación decádica por pulsos se ha venido utilizado en exclusividad desde los orígenes de la telefonía automática hasta tiempos relativamente recientes.
En la actualidad, aunque las modernas centrales digitales siguen aceptando este tipo de marcación, se utiliza mayoritariamente la marcación por tonos multifrecuencia, mucho más eficiente que la aquí descrita."



Notar que volvemos a la pantalla inicial de la aplicación. De modo que, si tenemos más de una cuenta de acceso, podemos introducirla ahora:



Cuando terminemos de configurar nuestras cuentas seleccionamos **Quit→<Aceptar>** y pasamos a intentar conectar, para eso escribiremos desde un terminal de texto:

`$pon Averroes`

Listo, ya podemos comprobar con Mozilla o el navegador que más nos guste que podemos navegar por la red.

Para desconectar de Internet sólo tendremos que ejecutar desde una **xterm**:

`$poff Averroes`

Ya hemos configurado la conexión a Internet

Por defecto, en nuestro escritorio tenemos un icono que nos da acceso a un navegador web, Mozilla Firefox, pero hay otros muchos más[44].

Pistas para detectar problemas

Puertos serie en Linux Disponemos de un comando que nos permite configurar el puerto serie, se trata del comando `setserial`

Para conocer cómo trabajar con él podemos ejecutar (desde un terminal de órdenes):

`$setserial --help`

para obtener una ayuda básica de los parámetros que admite:

`$man setserial`

para obtener la ayuda completa sobre el programa[45].

Para conocer el estado de un puerto serie podemos ejecutar (como root)

`#setserial -a /dev/ttySx`

donde `ttySx` es el correspondiente al puerto de comunicaciones del DOS .

setserial

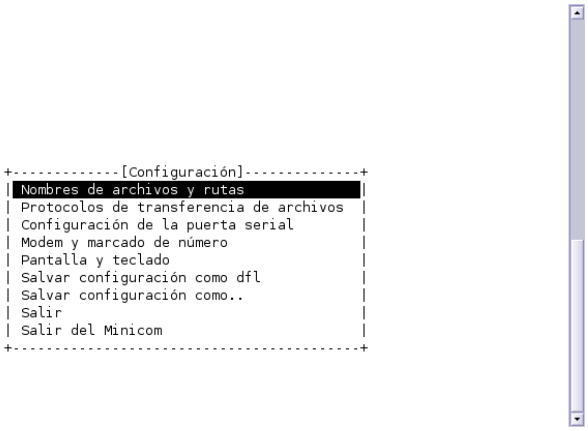
[44] Además de los comentados: konqueror, amaya, ... o navegadores en modo texto (lynx, ...)

[45] Se sale de la ayuda con `q`.

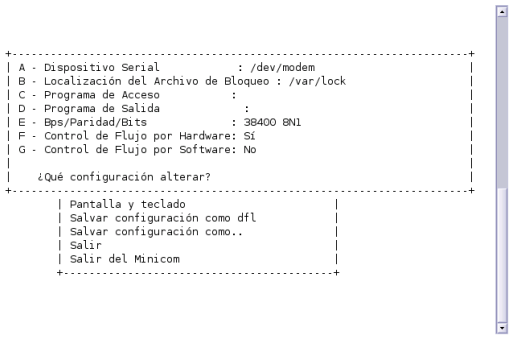
Programa minicom Se trata de una utilidad que, en caso de tener dificultades con la configuración del módem, nos puede ayudar a detectar en dónde puede estar el problema, se trata del programa `minicom`[46]. Con él podemos comprobar si el módem está bien conectado.

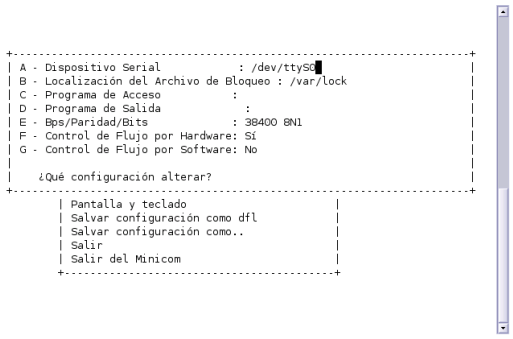
Para activarlo tenemos que ejecutar desde un terminal gráfico el comando:

```
$ minicom -s
```

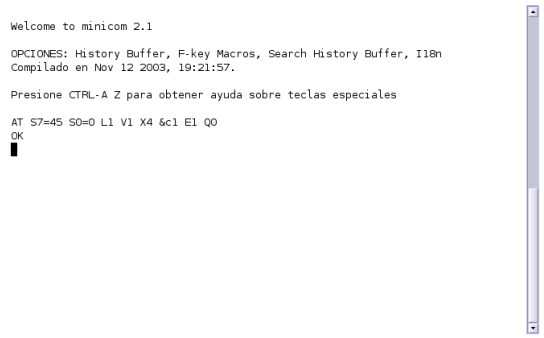


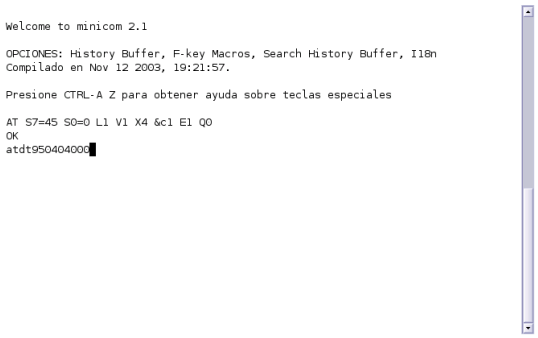
Con la opción `-s` optamos por entrar en el menú de configuración anterior[47]. Si marcamos en **Configuración de la puerta serial**, accederemos a:





Debemos optar por seleccionar el puerto serie al que está conectado nuestro módem, observar que en el ejemplo hemos sustituido `/dev/modem` por `/dev/ttyS0`. Una vez seleccionado el puerto adecuado (que no se nos olvide pulsar la tecla [**Intro**]) optaremos por [**Salvar configuración como dfl**] y después [**Salir**]. Si nos aparece una pantalla similar a la que sigue, y siempre que nos aparezca el [**OK**] final, es que todo ha ido bien. Si no es así habrá que reconfigurar la conexión del módem y volver a comprobarlo.



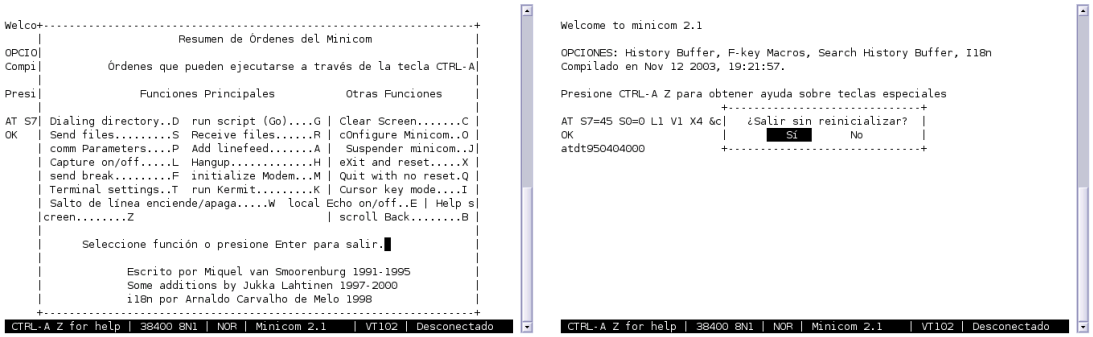


[46] Programa terminal de comunicaciones

[47] Sólo tendremos que usar esta opción la primera vez que ejecutemos el programa.

En la segunda captura, estamos comprobando que el teléfono está bien configurado, para eso, llamamos a un número de telefono y comprobamos que da tono de marcado, el comando a usar es `ATDTnúmero_teléfono`.

Para acceder al menú de este programa hay que utilizar la combinación de teclas **[Ctrl]+[a]** y después pulsar la letra **[z]**. Por ejemplo, para salir del programa hay que pulsar **[Ctrl]+[a]**, después **[z]** y por último **[q]**.



Si hemos configurado correctamente el programa, para acceder de nuevo a él, ya sólo hemos de escribir:

```
$minicom
```

Conocer la IP asignada Si deseamos comprobar si la conexión se produce con éxito y conocer la dirección IP que se nos ha asignado dinámicamente, podemos escribir desde un terminal la orden[48]:

- Debian:

  ```
  #plog averroes
  ```

- Red Hat (y Debian)

  ```
  #tail -f /var/log/messages
  ```

  Para cancelar el comando `tail` y dejar de visualizar las líneas que van saliendo hay que pulsar **[ctrl]+[c]**

veremos entonces una serie de mensajes que nos muestran cuál es el estado de la conexión, si ésta ha tenido éxito nos tienen que aparecer dos líneas del tipo:

```
local  IP address xxx.xxx.xxx.xxx
remote IP address xxx.xxx.xxx.xxx
```

donde esos números indican las direcciones IP asignadas dinámicamente a nuestra máquina y al servidor.

ping Una forma de saber si realmente hemos conectado bien, es hacer un `ping`. Este comando comprueba que llegamos a la máquina remota que queremos comprobar. Por ejemplo, `$ping 150.214.5.11`, para llegar al servidor de los cursos. El comando nos dirá si llegamos o devuelve error.

```
$ping 150.214.5.11
PING 150.214.5.11 (150.214.5.11) from 195.24.23.44 : 56(84) bytes of data.
64 bytes from 150.214.5.11: icmp_seq=1 ttl=64 time=0.040 ms
64 bytes from 150.214.5.11: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 150.214.5.11: icmp_seq=3 ttl=64 time=0.035 ms
```

[48] `$ /sbin/ifconfig` nos da información similar.

```
64 bytes from 150.214.5.11: icmp_seq=4 ttl=64 time=0.039 ms
--- 150.214.5.11 ping statistics ---
4 packets transmitted, 4 received, 0% loss, time 3006ms
rtt min/avg/max/mdev = 0.035/0.037/0.040/0.007 ms
```

Para terminar pulsamos [**Ctrl**]+[**c**]

Conectamos pero no salimos fuera Si, pese a que conectamos, no podemos visualizar páginas web, revisar:

- El fichero `/etc/resolv.conf`. Puede que pese a tener marcada la opción de que obtengamos la información sobre los DNS desde el proveedor, esto no sea así. En este caso lo mejor es que configuremos esto de forma manual. Podemos conseguirlo de dos formas:
  - Gráfica: véase 5.3.1 en la página 66 y 5.3.2 en la página 69
  - Texto Utilizando un editor de textos escribiremos en el fichero `/etc/resolv.conf`:

    ```
    domain cica.es
    nameserver 195.235.113.3
    nameserver 80.58.0.33
    nameserver 150.214.4.34
    ```

    cambiando los datos del ejemplo anteriores por los de nuestro servidor de acceso.
- Si tenemos una tarjeta de red, revisar la salida del comando:

  `#netstat -ar`

  si sale una línea del tipo:

  `default 192.168.0.254`

  u otra IP local, es que hemos configurado como *Gateway* una máquina local. Debemos eliminarla.

5.4.2. ADSL

Hoy en día suele ser más habitual la conexión a internet a través de una línea ADSL que a través de un módem RTB. Como en el caso anterior, debemos saber primero si nuestro módem ADSL está soportado por Linux o no. Disponemos de dos tipos básicos de módems ADSL:

- USB: son los más difíciles de configurar, serían el equivalente a los Winmodems RTB. No todos funcionan correctamente.
- Módem-router: no presentan ningún problema.

Usando un módem router

El proceso es similar a lo expuesto en 5.3 en la página 64, por tanto, sólo daremos una pinceladas para aclarar esta cuestión. La configuración gráfica de la conexión ADSL la podemos realizar con esas mismas herramientas, usaremos:

Debian

Aplicaciones→Configuración→Sistema→Red

o directamente desde una **xterm**:

```
# network-admin &
```

Fedora

→Configuración del Sistema→Red

o directamente desde una xterm

```
# neat &
```

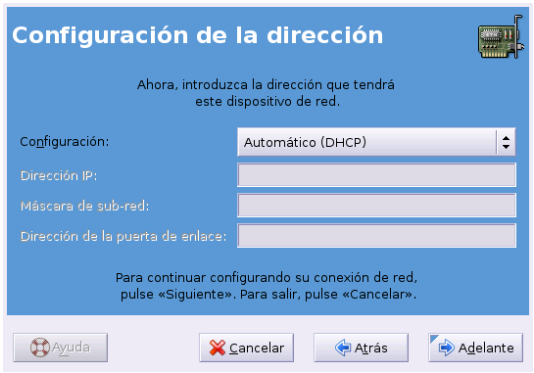
Lo único a tener en cuenta en ambos casos es cómo obtenemos la configuración desde el servidor (DHCP, BOOTP) o manual.

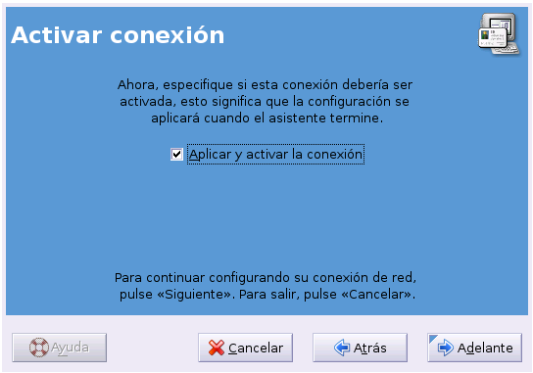
Cuando contratamos una ADSL debemos fijarnos si nuestro ISP[49] (proveedor de acceso a Internet) nos facilita una IP fija o IP dinámica.

En un principio cuando se contrataba una línea ADSL te *"regalaban"* la IP fija, ahora no (salvo ofertas). De todos modos, en cualquier momento, si lo deseamos, podemos solicitar una IP fija a nuestro ISP, previo pago mensual, claro.

Aclarado ésto, configuraremos nuestra conexión ADSL en función de cómo tengamos el módem router configurado:

Monopuesto sólo hay que decirle a la tarjeta de red que obtenga la configuración de direcciones IP automáticamente con DHCP y marcar la opción de obtener automáticamente información sobre el DNS desde el proveedor (capturas de GuadaLinex).



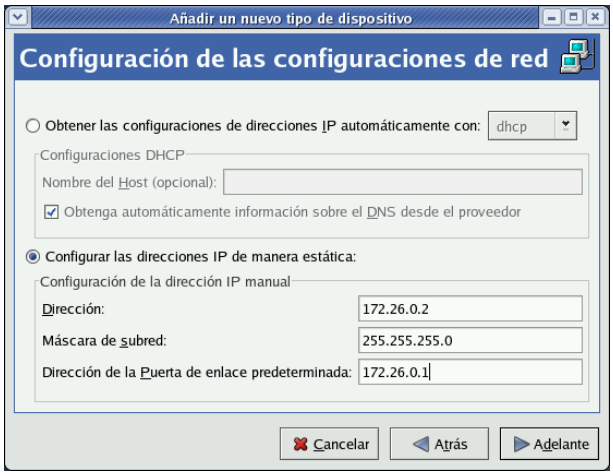


Multipuesto debemos diferenciar dos posibilidades en función de la forma en que esté configurado nuestro router:

En modo DHCP es la forma más usual, la conexión se realiza igual que en el caso Monopuesto.

Asignando la IP de forma Manual: en este caso tendremos que poner una dirección IP a nuestro interfaz de red de la misma red que el router (este dato lo tenemos que conocer a través del proveedor de acceso). Así por ejemplo, si nuestro módem router tiene la IP local 172.26.0.1 y como máscara de red 255.255.255.0 sólo le diremos al interfaz de red que use como *Gateway* la IP local del router y las IP de los servidores de nombres. Si optamos por poner de IP a nuestra máquina linux la dirección 172.26.0.2, quedaría (captura para Fedora):

[49] Del inglés *Internet Service Provider*.



Y para añadir los servidores de nombres (fichero `/etc/resolv.conf`), en la ventana principal de ambos programas, pulsaremos sobre la pestaña DNS e introducimos las IP de nuestros servidores de nombres. Se trata de rellenar los datos necesarios en estos campos. Necesitamos conocer el nombre de nuestro servidor de Internet. En el caso de la red del ejemplo con la que estamos trabajando escribríamos como DNS 195.235.113.3, 80.58.0.33 y 150.214.4.34, que serían los DNS primario, secundario y terciario (véase 5.3.1 en la página 66 y 5.3.2 en la página 69).

Llegados a este punto, y tras activar el interfaz de red, para saber si todo está bien podemos:

- Abrir un navegador Web y comprobar que salimos fuera.
- Hacer un ping a una máquina remota:

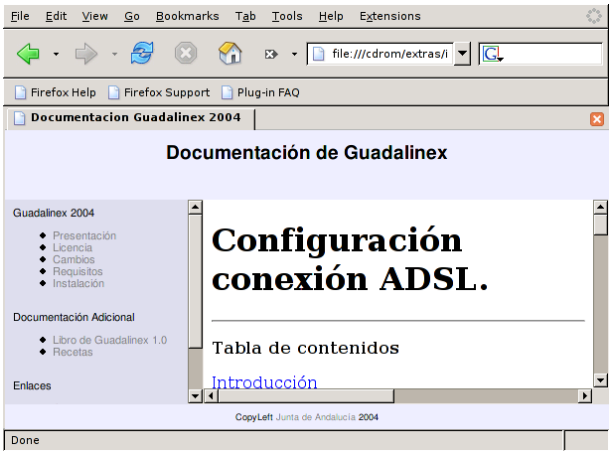
```
$ping mileto.cica.es
PING mileto.cica.es (150.214.5.11) from 80.30.154.77 : 56(84) bytes of data.
64 bytes from mileto.cica.es (150.214.5.11): icmp_seq=1 ttl=53 time=101 ms
64 bytes from mileto.cica.es (150.214.5.11): icmp_seq=2 ttl=53 time=97.3 ms
64 bytes from mileto.cica.es (150.214.5.11): icmp_seq=3 ttl=53 time=113 ms
64 bytes from mileto.cica.es (150.214.5.11): icmp_seq=4 ttl=53 time=93.8 ms
64 bytes from mileto.cica.es (150.214.5.11): icmp_seq=5 ttl=53 time=101 ms
--- mileto.cica.es ping statistics ---
5 packets transmitted, 5 received, 0% loss, time 4033ms
rtt min/avg/max/mdev = 93.849/101.676/113.561/6.657 ms
```

Pistas para conectar con módem USB usando Guadalinex.

Si nuestro módem es USB, tendremos que "confiar" en que algún "alma caritativa" haya resuelto ya el problema ya que sólo algunos modelos (cada vez más) son fácilmente configurables.

La dificultad de trabajar con modelos basados en esta tecnología reside en que tienen que estar soportados por Linux. En primer lugar deberíamos revisar la información contenida en el CD de Guadalinex, accesible desde el escritorio  **Conexión a Internet** fichero `LEAME-PRIMERO.txt`. En él, se nos informa de que si nuestro modelo es un Comtrend CT-350, Sagem Fast 800 o un Conexant Accessrunner USB estamos de suerte ya que podremos trabajar con ellos en Linux. Como añadido deberíamos consultar

- La Web `http://www.guadalinex.org/guadapedia/index.php/Receta:_C%C3%B3mo_configurar_una_conexi%C3%B3n_ADSL_%28Guadalinex_2004%29`
- La ayuda contenida en el CD, subdirectorio `extras/info/recetas`,



Podemos obtener más información para nuestro modelo USB en:

- `http://wiki.escomposlinux.org/Escomposlinux/EscomposlinuxHardware`

También pueden ser de ayuda las páginas:

- `http://personal.telefonica.terra.es/web/adslusb/`
- `http://cp4218.sourceforge.net/`

Para los tres modelos de módem USB antes comentados, previo a iniciar la configuración debemos instalar el paquete adecuado.

Capítulo 6

Linux como Router y Cortafuegos

Estaría perdido sin mis cortafuegos (Mark J. Cox, Director adjunto de Red Hat)

6.1. Router Linux

Como ya hemos visto, un sistema Linux puede funcionar haciendo la función de router. Simplemente, se conecta a dos o más redes y sabiendo, a partir de su tabla de rutas, por qué interfaz puede alcanzar cada red, dirige los paquetes entre una y otra red. Éste sería un funcionamiento como router estático[1]. Para ello es necesario que la característica *IP forwarding* (o routing entre interfaces) esté activada en el núcleo de nuestro sistema. Si no está activada, cuando llegue un paquete por un interfaz, no podrá dirigirse hacia otro interfaz, porque no estará permitido el routing entre ellos.

Podemos comprobar si está activo el IP forwarding con el comando:

```
[root@linux entrega04-1]# cat /proc/sys/net/ipv4/ip_forward
0
```

En este caso, la salida `0` indica que no está activado el routing del kernel.

Podemos activarlo mediante el comando[2]:

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

Los dos métodos siguientes, harían el cambio permanente. El primero consiste en cambiar el fichero de configuración del kernel (`/etc/sysctl.conf`), poniendo el siguiente valor:

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

En Red Hat o Fedora, también podremos hacerlo en el fichero `/etc/sysconfig/network`, poniendo lo siguiente:

```
FORWARD_IPV4=true
```

Otra opción para que Linux funcione como router, es la de instalar un servidor especializado de routing en nuestro sistema Linux. Por ejemplo, *zebra* (`www.zebra.org`), que incorpora protocolos de routing dinámicos como BGP-4, RIP u OSPF.

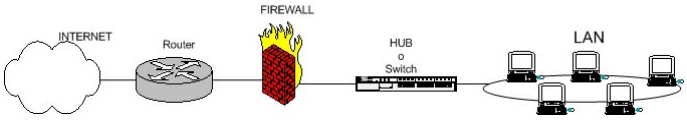
La sola capacidad de routing, nos permite conectar varias redes, pero nos surge otro problema mayor: la seguridad. Esto hace que la mayoría de sistemas Linux (y no Linux) que interconectan redes, incorporen capacidades de seguridad, como los cortafuegos.

[1] Aprovechando incluso un viejo 486, podemos tener un router funcional.

[2] El problema es que con este método, el cambio no sería permanente. Al reiniciar el ordenador lo habremos perdido.

6.2. Cortafuegos Linux

Un cortafuegos (o firewall) es un componente o conjunto de componentes que restringen el acceso entre una red protegida e Internet, o entre varias redes. Incluye tanto componentes hardware como software, configuraciones y definición de políticas de seguridad[3].



Su propósito es doble: proteger los sistemas y servicios internos de los ataques del exterior, y controlar los accesos hacia Internet de nuestros usuarios.

6.2.1. Clasificación de cortafuegos

Según el nivel de la pila de protocolos sobre el que trabajan, podemos clasificar los cortafuegos en:

Cortafuegos de Nivel de red: El control de tráfico a nivel de red consiste en analizar todos los paquetes que llegan a un interfaz de red, y decidir si se les deja pasar o no, en base al contenido de los mismos: protocolo, dirección de origen, dirección de destino, puerto origen y puerto destino fundamentalmente.

↬Su operación se asemeja a la de un guardia de tráfico que en un cruce decide qué coches pueden pasar y cuáles no, dependiendo de su origen y su destino.

Puesto que analizar esta información es muy sencillo, este tipo de cortafuegos suelen ser muy rápidos y transparentes al usuario. Se suelen denominar de filtrado de paquetes (*packet filter*). Una mejora sobre este tipo de cortafuegos, serían los de Inspección de Estado (*Stateful Inspection*) que además, inspeccionan en el interior de los paquetes para comprobar si cumplen las políticas de seguridad.

Cortafuegos de Nivel de aplicación: Se basan en la instalación de intermediarios (proxies), también conocidos como pasarelas (*application gateways*). El cliente, situado en un lado del cortafuegos, habla con un intermediario situado en el propio cortafuegos. Este intermediario lo identifica, registra sus peticiones y, si está permitido, las encamina hacia el verdadero servidor situado al otro lado del cortafuegos. La contestación regresa por el mismo camino, quedando igualmente registrada.

El control se hace interceptando las comunicaciones a nivel de aplicación, modificando el protocolo para incluir medidas adicionales de seguridad. El cortafuegos debe conocer los detalles del protocolo de cada servicio que intercepta, analizar su correcto funcionamiento y añadir los pasos de control precisos a los mismos. Por ejemplo, *squid* es un proxy que debe conocer el protocolo HTTP para recoger las peticiones de los navegadores cliente y redirigirlas al servidor destino. El cortafuegos es, por tanto, mucho más inteligente y posee un control más fino de todo el proceso de comunicación, aunque esto supone una mayor carga de trabajo y penalización en eficiencia. Además, normalmente exigen realizar modificaciones en la aplicación del usuario, como por ejemplo, decirle al navegador que utilice el proxy.

6.2.2. Terminología de cortafuegos

En una arquitectura de sistema cortafuegos, encontramos una serie de términos o componentes como son:

host bastión: (también se denomina *gates*) es un sistema especialmente asegurado, pero que puede recibir ataques por estar accesible desde Internet. Tiene como función ser el punto de

[3]Una política de seguridad nos dice qué es lo que se puede hacer y qué no se puede hacer en una red.

contacto de los usuarios de la red interna de una organización con otro tipo de redes. El host bastión filtra tráfico de entrada y salida, y también oculta la configuración de la red hacia fuera.

El *filtrado* también se conoce como *screening*, y a los dispositivos que lo implementan se les denomina *chokes*.

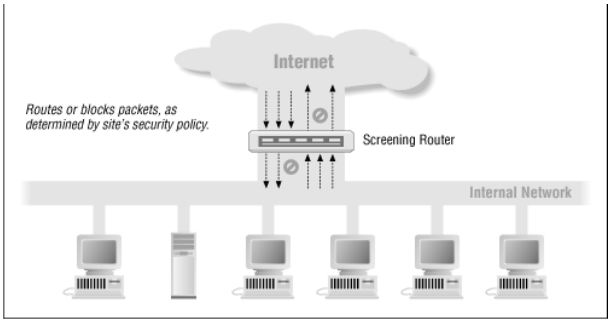
De la combinación de estos elementos, filtrado y host bastión, surgen las siguientes arquitecturas de cortafuegos.

6.2.3. Arquitecturas de cortafuegos

Según los componentes que incluya el cortafuegos y su ubicación, obtenemos las siguientes arquitecturas:

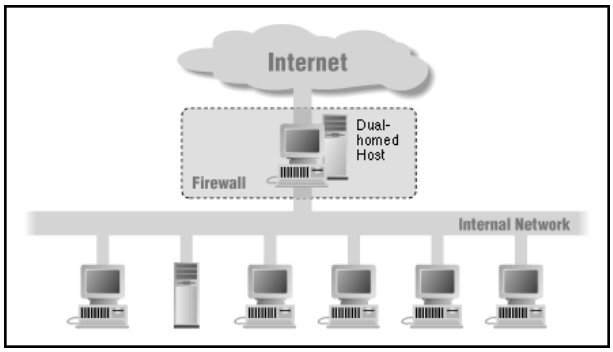
1. Cortafuegos de filtrado de paquetes (*Screening Router*)

   Un firewall sencillo puede consistir en un dispositivo capaz de filtrar paquetes, un choke. Basado en aprovechar la capacidad de algunos routers - denominados screening routers - para hacer un enrutado selectivo, es decir, para bloquear o permitir el tránsito de paquetes mediante listas de control de acceso en función de ciertas características de las tramas, de forma que el router actúe como pasarela de toda la red.



2. *Dual-Homed Host* (Host en dos zonas)

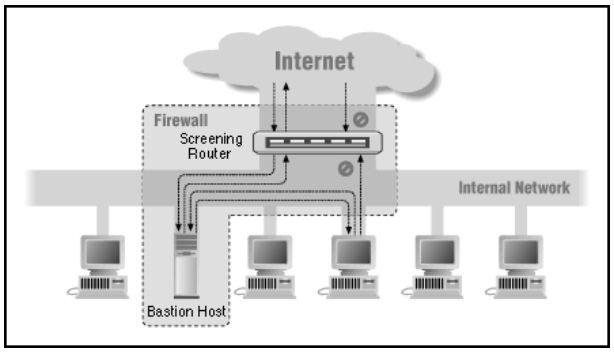
   Este modelo de cortafuegos está formado por máquinas equipadas con dos tarjetas de red y denominadas dual-homed hosts, en las que una de las tarjetas se conecta a la red interna a proteger y la otra a la red externa. En esta configuración el choke y el bastión coinciden en el mismo equipo.



3. *Screened Host*

   Un paso más en términos de seguridad de los cortafuegos es la arquitectura screened host o choke-gate, que combina un router con un host bastión, y donde el principal nivel de seguridad proviene del filtrado de paquetes (es decir, el router es la primera y más importante línea de defensa). En la máquina bastión, único sistema accesible desde el exterior, se ejecutan los

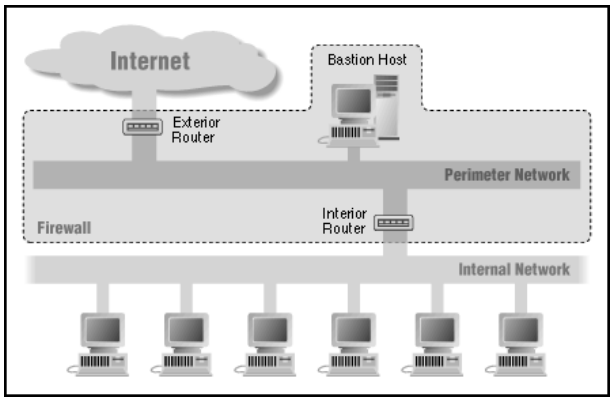
proxies de las aplicaciones, mientras que el choke se encarga de filtrar los paquetes que se puedan considerar peligrosos para la seguridad de la red interna, permitiendo únicamente la comunicación con un reducido número de servicios.



4. *Screened Subnet* (DMZ)

   La arquitectura Screened Subnet, también conocida como red perimetral o Zona Desmilitarizada (*De-Militarized Zone* o DMZ) añade un nivel más de seguridad en las arquitecturas de cortafuegos, situando una subred (la DMZ) entre las redes externa e interna, de forma que se consiguen reducir los efectos de un ataque con éxito al host bastión. En los modelos anteriores, si la seguridad del host bastión se ve comprometida[4], la amenaza se extiende automáticamente al resto de la red. Como la máquina bastión es un objetivo interesante para muchos piratas, la arquitectura DMZ intenta aislarla en una red perimetral de forma que un intruso que accede a esta máquina, no consiga un acceso total a la subred protegida.

   Screened subnet es la arquitectura más segura, pero también la más compleja; se utilizan dos routers, denominados exterior e interior, conectados ambos a la red perimetral como se muestra en la figura. En esta red DMZ, que constituye el sistema cortafuegos, se incluye el host bastión y también se podrán incluir sistemas que requieran un acceso controlado, como baterías de modems, el servidor web o el servidor de correo, que serán los únicos elementos visibles desde fuera de nuestra red.



6.2.4. Firewalls sobre linux

Los cortafuegos pueden estar basados en la combinación de un hardware y un software especializado, constituyendo "cajas" preparadas para realizar esa función, como por ejemplo los Nokia IP, los Cisco PIX o las cajas de StoneSoft. Otros, sin embargo, están construidos sobre un sistema operativo de propósito general, que se blinda[5] y prepara para funcionar como cortafuegos. En este

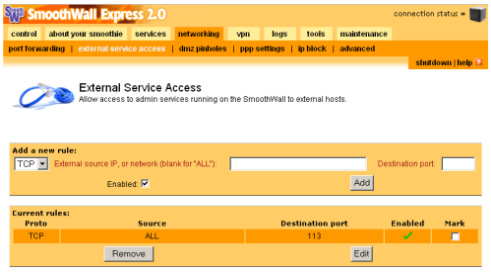
[4] Alguien consigue tomar el control o ejecutar comandos desde él.
[5] Se refuerzan las características de seguridad. *Hardening* en inglés.

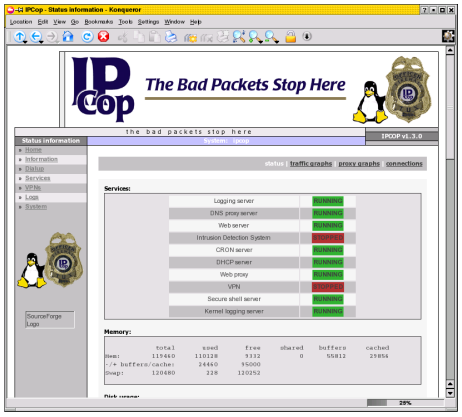
caso, podemos tener un sistema operativo Solaris, Windows o Linux sobre los que se ejecuta un software como el Firewall-1 de Check Point.

Linux se puede encontrar en las dos opciones. Hay empresas que utilizan linux sobre un hardware específico, dando una caja[6] lista para ser utilizada como cortafuegos, como el iForce de Sun Microsystems. También podemos coger un sistema Linux sobre un PC e instalar un cortafuegos. Lo más normal, en ambos casos, es utilizar las características de cortafuegos que incorpora el propio kernel de Linux: ***iptables***.

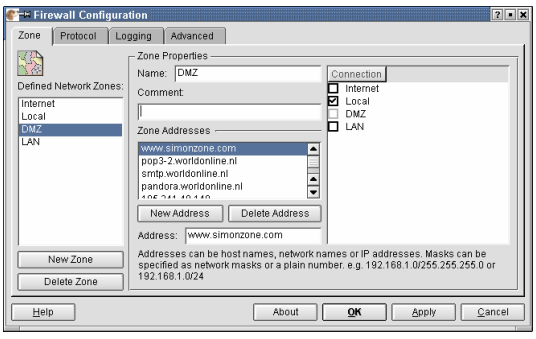
Un sistema cortafuegos puede ofrecer sus servicios a una red que se sitúa detrás de él, denominándose *cortafuegos de la red*, o el cortafuegos protege a la propia máquina en la que se ejecuta, denominándose *cortafuegos personal*.

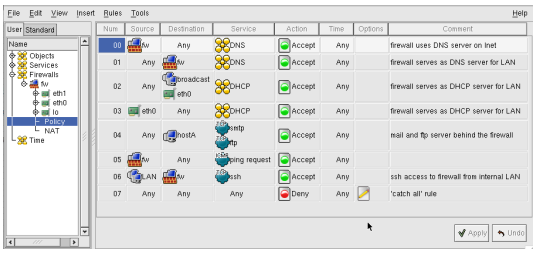
Para un cortafuegos de red, podemos utilizar distribuciones especializadas, que incorporan facilidades de administración por medio de navegador, integran proxys y utilidades adicionales. Podemos encontrar en este segmento a SmoothWall (`www.smoothwall.org`) o IPCop (`www.ipcop.org`). El proceso a seguir consiste en descargarse la imagen ISO de la distribución, grabar el CD e iniciar la instalación. El proceso es muy guiado y nos va haciendo las preguntas correspondientes para configurar el cortafuegos a nuestro gusto. La apariencia de ambas distribuciones la podéis observar en las siguientes figuras.



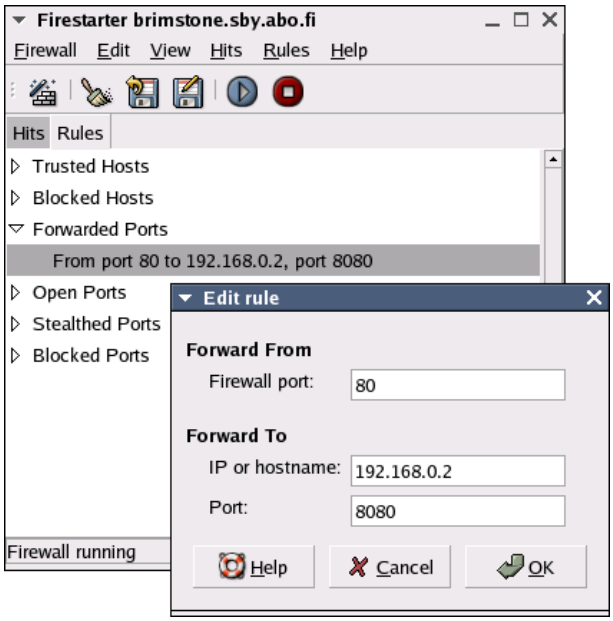


Para un cortafuegos personal, trabajamos con nuestra distribución favorita y se configura iptables para añadir seguridad. Sin embargo, el trabajar directamente con iptables puede ser muy engorroso y existen interfaces gráficos que nos permiten la configuración y el trabajo de forma más fácil. En este segmento encontramos a GuardDog (`http://www.simonzone.com/software/guarddog`), Firestarter (`http://firestarter.sourceforge.net`) o Firewall Builder (`http://www.firewallbuilder.org`), detalles de los cuales se muestran a continuación.

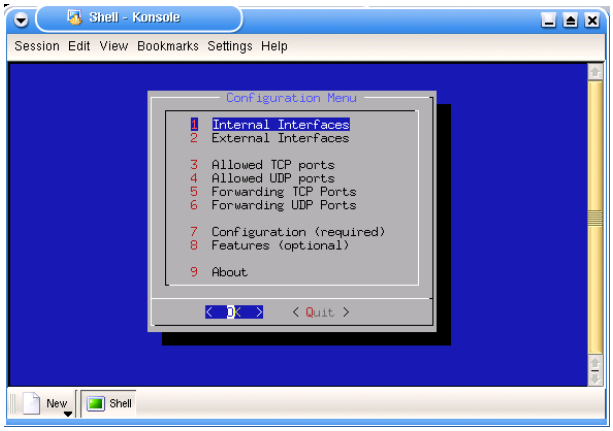




[6]En inglés, al concepto de entregar una caja cerrada y lista para enchufar en la red, se le llama *appliance*.



Incluso tenemos utilidades en terminal alfanumérico como `Firewall-jay` (`http:/firewall-jay.sourceforge.net`), simples pero potentes.



6.2.5. ¿Qué es iptables?

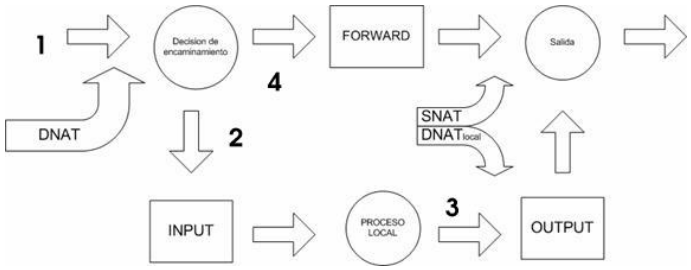
Como hemos comentado anteriormente, la principal herramienta de cortafuegos para Linux a partir de los kernels 2.4[7], es iptables. Iptables reemplaza al anterior *ipchains* de los kernels de la versión 2.2 y a *ipfwadm* de los kernels 2.0. La función de *iptables* es la de establecer, mantener e inspeccionar las reglas de filtrado de paquetes IP en el núcleo de Linux.

Iptables decide qué paquete de información puede pasar, según unos criterios que se almacenan en unas listas. Las listas se componen de reglas con un orden determinado, donde la última regla introducida será la última regla en aplicarse.

Cuando un paquete llega, se mira en qué lista debe aplicarse. En esa lista (iptables las llama tablas) se empieza por la primera regla. Si la regla no es aplicable al paquete, se pasa a la siguiente regla. Cuando una regla es aplicable (*match*) al paquete, se ejecuta la acción que haya sido definida en la regla (descartar el paquete, aceptarlo, enrutarlo, etc).

Veamos el camino que seguiría un paquete en el kernel utilizando iptables:

[7] Y sigue vigente para los actuales kernels 2.6



Cuando iptables recibe el paquete (1), se comprueba si el destino final es nuestra propia máquina o es otra, porque estemos funcionando como router/gateway o cortafuegos. Para los paquetes que van a la propia máquina se aplican las reglas INPUT (2) y para paquetes que van a otras redes o máquinas se aplican las reglas FORWARD (4). Las reglas de OUTPUT (3) se aplican cuando un paquete es enviado desde nuestra máquina al exterior.

INPUT,OUTPUT y FORWARD son los tres tipos de reglas de filtrado (FILTER). Antes de aplicar esas reglas es posible aplicar reglas de NAT[8] y de MANGLE[9].

La estructura de un comando iptables es la siguiente :

```
iptables -t [tabla] -[opciones] [regla] [criterio] -j [acción]
```

Veamos qué significa cada elemento:

-t [tabla] Esta parte del comando especifica cuál es la tabla en la que aplicamos la regla. Existen 3 tipos de tablas: FILTER, NAT y MANGLE, siendo *filter* la tabla por defecto si se omite esta parte del comando.

Filter es la tabla donde se añaden las reglas relacionadas con el filtrado de paquetes.

Nat se refiere a las conexiones que serán modificadas por el firewall, como por ejemplo, enmascarar conexiones, realizar redirecciones de puertos, etc.

Mangle es parecido a Nat, pero tiene la posibilidad de modificar más valores del paquete.

-[opciones] Las opciones básicas del comando son las siguientes :

A para añadir (*Append*) una regla.

L es para listar (*List*) las reglas.

F es para borrar (*Flush*) todas las reglas o en el caso de que INPUT, FORWARD o OUTPUT sean dados como argumento, se borrarán las reglas asociadas sólo a esa clase.

P establece la política (*Policy*) por defecto del firewall. Por defecto es aceptar todas las conexiones.

[regla] Reglas válidas son INPUT, FORWARD y OUTPUT.

[criterio] Aquí es donde se especificarán las características del paquete que casará con esta regla. Algunos ejemplos son:

-s : dirección de origen (source). Puede ser una dirección IP o una red. `-s 192.168.1.0/24`

-d : dirección de destino. `-d 84.56.73.3`

-p : tipo de protocolo (TCP,UDP,ICMP). `-p TCP`

–**sport** : puerto de origen

–**dport:** puerto de destino `--dport 23`

[8] Se usan para hacer redirecciones de puertos o cambios en las IPs de origen y destino

[9] Modifica paquetes, pero es más rica y potente que Nat. Con Mangle podemos modificar cualquier aspecto del paquete (flags, TTL, etc).

-i = in-interface : el interfaz[10] por el que se entra `-i eth0`

-o = –out-interface: el interfaz[11] por el que se sale `-o ppp0`

-j [accion] Aquí establecemos qué es lo que hay que hacer con el paquete. Las posibles acciones son :

ACCEPT: aceptar el paquete.

REJECT o DROP: desechar el paquete. La diferencia entre ellos reside en que DROP descartará el paquete silenciosamente y REJECT emitirá un paquete ICMP Port Unreachable, indicando que está cerrado el puerto.

REDIRECT redirigir el paquete a donde se indique en el criterio del comando.

LOG archiva el paquete para su posterior análisis.

Hay dos maneras de implementar un firewall, según la política por defecto que especifiquemos:

1. Política por defecto ACEPTAR: Se aceptan por defecto todos los paquetes. Sólo se denegará lo que se diga explícitamente. El equivalente sería la política de acceso a un bingo: pueden entrar todas las personas, excepto aquellas cuyo DNI aparezca en la lista de acceso prohibido.
2. Política por defecto DENEGAR: Todo está denegado, y sólo se permitirá pasar por el firewall aquello que se permita explícitamente. El equivalente sería el acceso a la cámara de cajas de seguridad de un banco. El acceso está prohibido a todo el mundo y se habilita una lista de personas autorizadas a entrar. Para un cortafuegos, se recomienda aplicar esta política por defecto.

Ejemplos de iptables

Veamos una regla que acepta conexiones al puerto 80 de nuestro equipo:

```
iptables -A INPUT -i eth0 -s 0.0.0.0/0 -p TCP --dport www -j ACCEPT
```

Comando `iptables`. Llamada al comando con los argumentos siguientes.

-A *append*, opción para añadir la regla

INPUT aplicable a los paquetes que entran a nuestra máquina

-i eth0: por el interfaz de red eth0

-s 0.0.0.0/0 dirección de origen del paquete. En este caso, cualquier dirección.

-p TCP tipo de protocolo. En este caso TCP.

–**dport** puerto de destino. Se especifica `www`, que mirando en `/etc/services` es el puerto número 80.

-j ACCEPT qué acción se realiza sobre el paquete. Se acepta.

Veamos otro ejemplo:

```
   iptables -A INPUT -p tcp -i eth0 -m state --state NEW,ESTABLISHED --dport 22 -j
ACCEPT
   iptables -A INPUT -p all -i eth0 -m state --state NEW,INVALID -j DROP
```

La primera regla deja pasar los paquetes con destino a nuestra máquina (INPUT), por protocolo `tcp`, que entren por el interfaz eth0 con destino al puerto 22. Como habéis visto, nos hemos saltado

[10] -i se usa con reglas INPUT y FORWARD
[11] -o se usa con reglas FORWARD y OUTPUT

algo. La opción `-m state` indica que tiene que cargar el módulo de inspección de estado. Es decir, va a escudriñar el interior del paquete, no sólo su filtrado: esto se conoce como inspección de estado. Ahora ya es aplicable la opción `--state NEW, ESTABLISHED`, que indica que pueden pasar los paquetes que abren una nueva conexión[12] (NEW) o pertenecen a una conexión ya establecida (ESTABLISHED).

La segunda regla descarta (`drop`) todas las conexiones entrantes (`INPUT`) de todos los protocolos (`-p all`), que intenten abrir una nueva conexión (`NEW`) o no pertenezcan a una conexión ya establecida (`INVALID`) desde la interfaz eth0.

Firewall Personal con Guadalinex 2004

En Guadalinex 2004 disponemos de un firewall que viene con el sistema. Se llama Firestarter. En realidad es un interface gráfico para iptables, pero que nos será muy útil para configurar nuestro cortafuegos, ya sea como cortafuegos personal[13] o como cortafuegos para proteger una red.

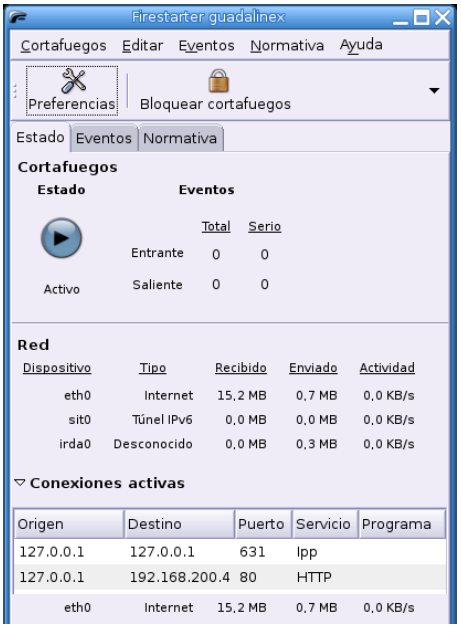
Lo primero que haremos es actualizar a una versión más reciente que la que viene por defecto. Para ello utilizamos `apt-get`

```
#apt-get install firestarter
```

Podemos lanzar la el cortafuegos desde **Aplicaciones→Configuración→firestarter** o ejecutando desde la línea de comandos[14]

```
#firestarter
```

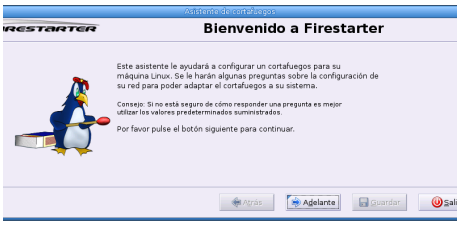
La ventana que nos aparece es la siguiente



Lo más destacable es que estamos viendo las conexiones activas en nuestra máquina. Con lo que no es solamente una ayuda para crear las reglas sino que también nos permite visualizar las conexiones activas en tiempo real.

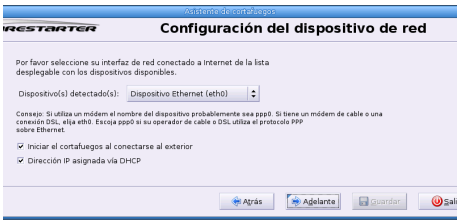
Veamos cómo podemos realizar la configuración. La forma más fácil es con el asistente, que podemos llamar desde la entrada **Cortafuegos** y **Ejecutar asistente**, desde el menú principal.

[12] ¿Recordáis el *three-way andshake* necesario para establecer una nueva conexión TCP?
[13] Solamente protegemos a nuestro propio sistema.
[14] Para ambos métodos, hay que tener privilegios de superusuario



Si pulsamos **Adelante**, pasamos a una ventana en la que podemos seleccionar el interfaz de nuestro sistema que se conecta a Internet. En este caso es `eth0` para conectarnos a un router ADSL. En esta opción estamos seleccionando el firewall personal que protege nuestra máquina del hostil mundo exterior.

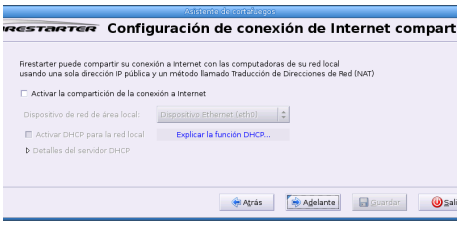
Además, seleccionamos que el cortafuegos se active al conectarnos a Internet.



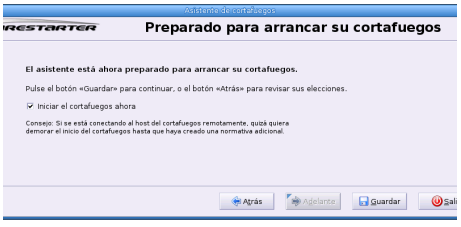
La siguiente pantalla nos permite decidir si el firewall de nuestro sistema además funcionará como pasarela para otros sistemas, convirtiéndose en el guardián de nuestra red.

Para ello, debemos contar con otra interfaz de red distinta de la anterior, que se conectará a la red protegida.

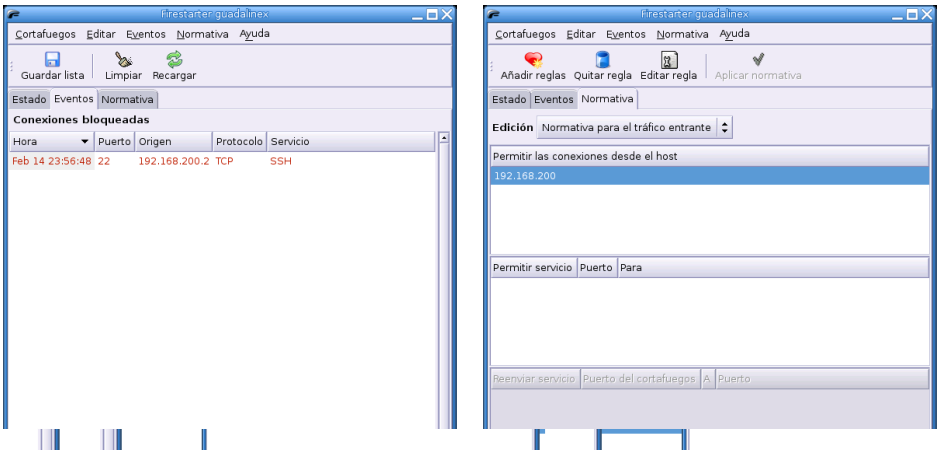
En este caso, como solamente queremos un firewall personal, no activamos la opción **Activar la compartición de la conexión a Internet**. Si quisiéramos que éste sea el cortafuegos de nuestra red, debemos especificar la interfaz conectada a la red de área local interna y si queremos activar el servidor de DHCP para dar direcciones automáticamente.



Pues ya hemos terminado una configuración básica del cotafuegos para uso personal. Podemos guardar las reglas e iniciar el cortafuegos.



La pestaña de **Estado** ya la conocíamos de la primera figura. Las otras dos pestañas son la de **Eventos**, que nos permite ver las conexiones que han sido bloqueadas por el cortafuegos y la de **Normativa**, que nos permite: añadir nuevas reglas y eliminar o modificar reglas, tanto de tráfico entrante como saliente.



Como véis, no es complicado y nos ayuda a mantener a raya a los chicos malos.

Firewall Personal con Fedora

Veamos las utilidades que Fedora pone a nuestro alcance para configurar un cortafuegos personal.

Para configurar en modo gráfico el firewall, ejecutaremos el programa `system-config-securitylevel` o con menús:

 **→Configuración del sistema→Nivel de seguridad**

La pantalla que se nos presenta es la siguiente[15]:

Las opciones de **Nivel de seguridad** serán *Habilitar cortafuegos* o *Deshabilitado*. En caso de que lo activemos, las conexiones a nuestro ordenador quedarán prohibidas. Deberemos abrir las conexiones que nos interesen, como WWW si tenemos un servidor web que queremos hacer accesible, o permitir conexiones SSH.

El último cuadro nos permite seleccionar los **Dispositivos fiables**. ¿Cuáles son éstos, los que no fallan nunca?. No, seguramente no es la traducción más acertada del término *trusted*, que podría ser más bien confiable y se refiere a los interfaces que, al estar en una red local protegida, consideramos que no nos van a venir ataques a través de ellos y permitimos relajar las conexiones que vengan por esa vía.

[15] En entregas posteriores veremos qué es esto de SELinux

Esta utilidad en realidad es un interfaz gráfico que escribe su configuración en el fichero `/etc/sysconfig/system-config-securitylevel`, como mostramos a continuación.

```
# /etc/init.d/networking restart
[root@linux images]# more /etc/sysconfig/system-config-securitylevel
#Configuration file for system-config-securitylevel
#Copyright (c) 2002 Red Hat, Inc. all rights reserved
--enabled
--trust=eth0
--port=http:tcp
--port=ssh:tcp
```

También existe una utilidad equivalente en modo texto, que ejecutamos mediante[16]

```
#system-config-securitylevel-tui
```

y cuya apariencia mostramos en la figura siguiente:

Si entramos en la opción de **Personalizar**, podemos seleccionar las opciones de apertura de nuestra máquina que deseemos, igual que antes en el modo gráfico.

Los valores de iptables se guardan en memoria cuando se ejecutan y necesitamos que se almacenen en algún sitio de donde poder recuperar la cofiguración al rearrancar el servicio. Mediante las utilidades anteriores, hemos configurado los valores que aparecen en el fichero `/etc/sysconfig/iptables`. Mostremos su valor actual:

```
[root@linux images]# more /etc/sysconfig/iptables
# Firewall configuration written by redhat-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -i eth0 -j ACCEPT
```

[16] También con
`#lokkit`

```
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 443 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

¿Podéis reconocer los valores de este fichero con la configuración de la figura anterior? Seguro que sí.

Si queremos almacenar los valores de iptables en un fichero, debemos ejecutar el comando `iptables-save`. Y análogamente, para recuperar desde ese fichero los valores y cargarlos en el kernel, ejecutamos `iptables-restore`.

Para arrancar el servicio iptables, lo hacemos mediante el comando
`#service iptables start`
y para pararlo, mediante
`#service iptables stop`.

Un ejemplo práctico de uso de iptables

Tenemos dos ordenadores conectados en red mediante un cable cruzado. El que tiene acceso a internet (`host0`) dispone de dos tarjetas de red, una que permite salir hacia fuera mediante DHCP (`eth0`) y otra (`eth1`) que se conecta con el otro ordenador de la casa (`host1`).

La configuración del interfaz de red en `host1` es:

IP: 192.168.0.2

MASCARA: 255.255.255.0

PUERTA DE ENLACE: 192.168.0.1

La configuraciòn de la red en `host0` es:

eth0: DHCP

eth1:

IP: 192.168.0.1
MASCARA: 255.255.255.0

Problema: Al hacer ping de un ordenador a otro todo va bien. Pero cuando se hace un ping desde `host1` a una máquina remota, por ejemplo `mileto.cica.es`, nos responde *host desconocido.*

Solución: Un script que nos puede solucionar el problema es

```
[paco@eco ~]#cat ipro
echo 1 > /proc/sys/net/ipv4/ip_forward
#borra las reglas
iptables --flush
iptables --table nat --flush
#Activamos el NAT con enmascaramiento
iptables --table nat --append POSTROUTING -s 192.168.0.0/24 --out-
interface eth0 -j MASQUERADE
iptables --append FORWARD -s 192.168.0.0/24 --in-interface eth1 -
j ACCEPT
#politica para la red local de todo permitido
iptables -A INPUT -s 192.168.0.0/24 -j ACCEPT
```

Si lo ejecutamos como root (`./ipro`) activaremos el filtrado de paquetes. Para que todo funcione hay que poner en `host1` las IP de los servidores de nombres[17] en el fichero `/etc/resolv.conf`

[17] Si no las ponemos saldremos hacia fuera pero no resolveremos nombres

Capítulo 7

Configuración de DHCP

> DHCP es útil para proporcionar de un modo rápido la configuración de red del cliente. ...
>
> Además, si un portátil o cualquier tipo de equipo móvil se configura para DHCP, podrá desplazarse entre distintas oficinas sin tener que volver a configurarlo, siempre y cuando cada oficina tenga un servidor DHCP que permita su conexión a la red. (*Manual de personalización de Red Hat Linux*)

7.1. Introducción

Si a nuestro cargo está el mantenimiento de una red local, el uso de IPs estáticas acarrea algunos inconvenientes de mantenimiento[1]. Estos problemas se acrecientan cuanto más grande es la red ya que la asignación de IPs se puede hacer más compleja. Para intentar solucionar este problema se desarrolló el Protocolo para Configuración Dinámica de Terminales (*Dynamic Host Configuration Protocol* o DHCP [RFC 2131][2]).

Pero, ¿qué es DHCP?

DHCP es un protocolo TCP/IP que proporciona una asignación dinámica y automatizada de direcciones IP además de otro tipo de información añadida, como puede ser la puerta de enlace predeterminada o las IPs de los servidores de nombres. Se basa en UDP y utiliza los puertos 67 y 68, el servidor escucha en el puerto 67 y el cliente en el 68.

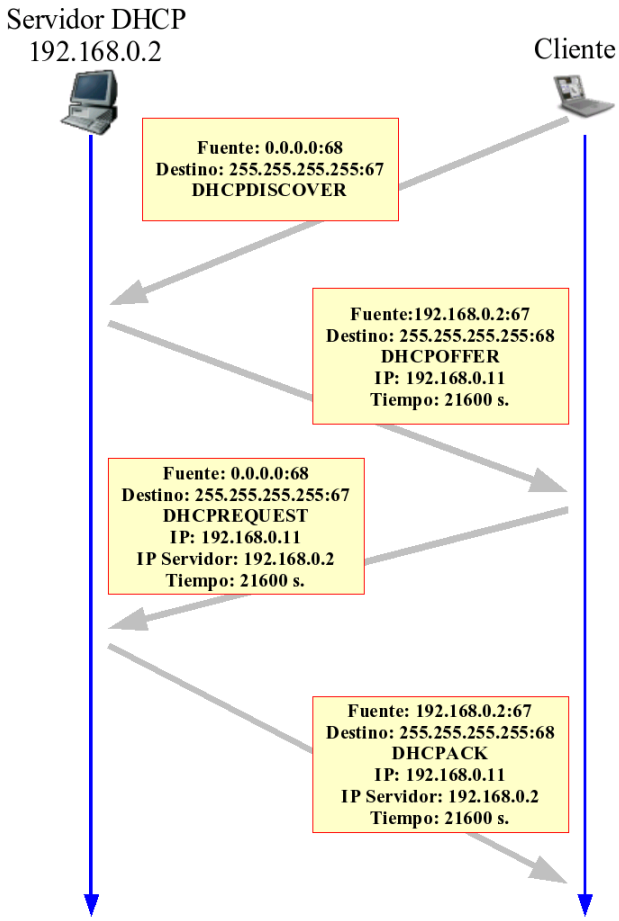
Cuando un ordenador al que no se le ha asignado IP fija se conecta a nuestra red, lo único de que dispone es de la dirección hardware del interfaz de red (*MAC address*). A partir de ese número, la máquina cliente realiza una llamada de difusión que tendrá respuesta por parte de nuestro servidor DHCP, indicándole al cliente, junto con la dirección, algunos parámetros adicionales.

Expliquemos esto un poco más[3]:

[1] Aunque personalmente, en un centro de enseñanaza, prefiero las IP fijas.

[2] Se basó en BOOTP (de *Boot Protocol*) mejorándolo.

[3] Entre paréntesis se incluyen los mensajes DHCP, para ampliar sobre su significado `http://dns.bdat.net/dhcp/x56.html`



1. El cliente difunde un mensaje de descubrimiento DHCP (DHCPDISCOVER), para eso envía un datagrama[4] UDP al puerto 67 con dirección de destino 255.255.255.255 y de origen 0.0.0.0 a la subred local.
2. Los servidores disponibles responden con un mensaje de ofrecimiento DHCP (DHCPOFFER), en ese mensaje se incluye la oferta de IP, el tiempo de concesión de esa IP, ...
3. Tras analizar las ofertas recibidas de los distintos servidores DHCP disponibles en la red, el cliente elige una de las ofertas en función de los parámetros incluidos en el mensaje y envía un mensaje de petición de DHCP (DHCPREQUEST) en que repite los parámetros de configuración.
4. El servidor guarda la asignación y responde con un mensaje (DHCPACK) de reconocimiento ACK en el que se confirman los datos de la configuración.

Como documentación complementaria, además de la ayuda que se instala con el programa, podemos ampliar sobre su uso en:

- Capítulo 18 de *Manual de personalización de Red Hat Linux*
  `http://europe.redhat.com/documentation/rhl9/rhl-cg-es-9/ch-dhcp.php3`
- *DHCPD mini-COMO para Linux*
  `http://es.tldp.org/COMO-INSFLUG/COMOs/DHCPd-Mini-Como/`

[4] Broadcast del cliente para localizar servidores

- `http://www.redes-linux.com/manuales.php`
- Internet Systems Consortium `http://www.isc.org/index.pl?/sw/dhcp/`
- `http://es.wikipedia.org/wiki/DHCP`

7.2. Instalación

En ambas distribuciones se instala por defecto el cliente `/sbin/dhclient`. En Fedora forma parte del paquete de igual nombre (`dhclient`), mientras que en Guadalinex 2004 el paquete que lo contiene es `dhcp3-client`.

Con Linux podemos disponer de servicios DHCP fácilmente, en ambas distribuciones sólo hay que instalar el paquete `dhcp`. La salida que se obtiene en Fedora es

```
# apt-get update; apt-get install dhcp
Leyendo lista de paquetes... Hecho
Creando Árbol de dependencias... Hecho
Nota, seleccionando dhcp en lugar de dhcp
Se instalarán los siguientes paquetes NUEVOS:
  dhcp
0 actualizados, 1 se instalarán, 0 para eliminar y 123 no actualizados.
Necesito descargar 110kB de archivos.
Se utilizarán 311kB de espacio de disco adicional después de desempa-
quetar.
...
Please note that if you are installing the DHCP server for the first
time you need to configure it first. Please stop (/etc/init.d/dhcp
stop) the DHCP server daemon, edit /etc/dhcpd.conf to suit your needs
and particular configuration, and restart the DHCP server daemon
(/etc/init.d/dhcp start).
You also need to edit /etc/default/dhcp to specify the interfaces dhcpd
should listen to. By default it listens to eth0.
NOTE: dhcpd's messages are being sent to syslog. Look there for
diagnostics messages.
Starting DHCP server: dhcpd failed to start -
check syslog for diagnostics.
```

Debian

La salida no se corresponde con las líneas anteriores, y es que en Debian no se iniciará el servicio, ya que antes hemos de configurarlo. Se nos avisa de que antes de ponerlo en marcha hemos de adecuar el fichero `/etc/dhcp.conf` a nuestra red y que podemos restringir los interfces de red a la escucha en el fichero `/etc/default/dhcp`. Para solucionar ambos temas, sigamos leyendo.

7.3. Configuración

7.3.1. De la máquina Linux

El fichero de configuración es `/etc/dhcpd.conf` y el fichero en donde se almacenan las IP asignadas[5] `/var/lib/dhcp/dhcpd.leases`. Analicemos el primero a partir del fichero de ejemplo

[5] Al periodo de préstamo de la dirección IP se le llama alquiler (*lease*). Este tiempo siempre será finito, de esta forma el servicio DHCP puede detectar la retirada del cliente y hacer de nuevo uso de la IP asignada. Si el tiempo se agota, DHCP dispone de un mecanismo que permite la renovación de la IP asignada al cliente por otro periodo de tiempo.

instalado[6] `/usr/share/doc/dhcp-3.0.1/dhcpd.conf.sample` .

Podemos trabajar de dos formas distintas: en modo de actualización DNS *ad-hoc* (no recomendado) y en el modo de actualización en el que interaccionan DHCP-DNS es decir, modo `interim`, es el modo por defecto[7]

```
ddns-update-style interim;
```

Para ignorar las solicitudes de actualización del registro A[8] de los clientes (permite asociar a un nombre la dirección IP)

```
ignore client-updates;
```

Características comunes a la subred especificada

```
subnet 192.168.0.0 netmask 255.255.255.0 {
```

IP de los routers de acceso a Internet (si son varios se separan con comas) y máscara de subred

```
# --- gateway por defecto
     option routers 192.168.0.1;
     option subnet-mask 255.255.255.0;
```

Especifica el nombre del dominio NIS[9] del cliente, el nombre de dominio para la máquina cliente y las IPs de los servidores de nombres (si son varios se separan con comas)

```
     option nis-domain "domain.org";
     option domain-name "domain.org";
     option domain-name-servers 150.214.3.9;
```

Se trata de la forma en que sincronizamos nuestro tiempo en función del Meridiano de Greenwich. Al venir dado en segundos y negativo equivaldría a -5 horas[10].

```
     option time-offset -18000; # Tiempo de la Costa Este Americana
```

IP del servidor NTP[11]

```
#    option ntp-servers 192.168.0.5;
```

IP del servidor WINS[12]. Indica la estrategia que cada cliente de una red NetBIOS debe seguir para realizar el registro de nombres y la resolución[13]. Si no tenemos claro de qué va, es mejor no tocarlo (véanse las notas a pie de página 12 y 13).

```
#    option netbios-name-servers 192.168.0.6;
```

[6] Si bien se estudia para Fedora, es similar para Debian. Para esta distribución, la documentación está en `/usr/share/doc/dhcp`

[7] Para usar el primero tendríamos que quitar esa línea y en su lugar escribir
`ddns-update-style ad-hoc;`
Para saber más de esto: `$man dhcpd.conf`

[8] Veremos qué significa en la siguiente entrega cuando hablemos del DNS.

[9] Normalmente no es utilizado, ni debemos preocuparnos de él.

[10] Para saber exactamente cómo se calcula y qué es este valor, mirar la web
`http://www.cisco.com/warp/public/109/calculate_hexadecimal_dhcp.html`

[11] *Network Time Protocol:* protocolo de comunicaciones que permite sincronizar los relojes de un ordenador con un servidor central de tiempo

[12] WINS es el servicio que resuelve los nombres NetBIOS de las redes "Microsoft" a direcciones IP. Si usamos SAMBA podemos conseguir que nuestro Linux se encargue de este menester.

[13] En una red NetBIOS existen los tipos (la tabla está tomada del libro *Usando Samba,* para ampliar sobre este tema a él os remitimos):

Papel	**Valor**
b-node	Usa registro *broadcast* y sólo resolución.
p-node	Usa registro punto-a-punto y sólo resolución.
m-node	Usa *broadcast* para registro. Si tiene éxito, notifica al servidor NBNS el resultado. Usa broadcast para resolución; usa servidor NBNS si el broadcast no tiene éxito.
h-node (hybrid)	Usa servidor NBNS para registro y resolución; usa *broadcast* si el servidor NBNS no responde o no está operativo.

```
# --- Selects point-to-point node (default is hybrid). Don't change this unless
# -- you understand Netbios very well
#    option netbios-node-type 2;
```

Rango de direcciones IP que el servidor puede asignar en esta subred[14]

```
range dynamic-bootp 192.168.0.128 192.168.0.255;
```

La IP se concede por ese tiempo en segundos, una vez consumido se tiene que renegociar la IP asignada

```
default-lease-time 21600;
```

Duración máxima para disponer de una IP (en segundos)

```
max-lease-time 43200;
```

Le asignamos a la máquina `ns` una IP fija y un nombre, se trata del interfaz de red con la dirección MAC especificada.

```
    # we want the nameserver to appear at a fixed address
    host ns {
       next-server marvin.redhat.com;
       hardware ethernet 12:34:56:78:AB:CD;
       option host-name "ns";
       fixed-address 207.175.42.254;
    }
}
```

Ejemplo

Una vez instalado, vamos a configurarlo partiendo de un ejemplo concreto. Vamos a considerar la situación de que disponemos de un router que nos da acceso a Internet y un Linux que va a ser el servidor DHCP de nuestra clase de informática (con equipos Windows XP[15] y Linux). Es decir, partimos de que disponemos de una red de clase C con las características:

- Red: 192.168.0.0/24
- IP router: 192.168.0.1
- IP Linux: 192.168.0.10
- Máscara de red: 255.255.255.0
- Rango de direcciones libres: 192.168.0.100 - 192.168.0.200
- IP del servidor de nombres: 80.58.0.33

Creemos con un editor el fichero `/etc/dhcpd.conf`[16]

```
$cat /etc/dhcpd.conf
ddns-update-style interim;
ignore client-updates;
subnet 192.168.0.0 netmask 255.255.255.0 {
     option routers 192.168.0.1;
     option subnet-mask 255.255.255.0;
```

[14] El servicio DHCP se configura a partir de la idea de ámbito: un ámbito es un rango de direcciones IP. Podemos tener un ámbito distinto para cada subred. Así, un ámbito debe contener completamente el rango de direcciones de una subred. Dentro de un ámbito, podremos reservar direcciones que están asignadas ya de forma estática (por ejemplo el propio servidor, el router de acceso a internet) y otras podemos asignarlas de forma dinámica.

[15] Con la familia 9x es similar

[16] Si en el Linux hemos configurado SAMBA podemos añadir
`option netbios-name-servers 192.168.1.5;`

```
    option domain-name-servers 80.58.0.33;
    range dynamic-bootp 192.168.0.100 192.168.0.200;
    default-lease-time 21600;
    max-lease-time 43200;
}
```

y pongamos el servidor en marcha:

Fedora

```
# service dhcpd start
```

Lo normal es que en Fedora, optemos porque el servicio se active en el arranque.

```
# /usr/sbin/ntsysv
```

Debian:

Para activarlo

```
#/etc/init.d/dhcp start
```

Y para que se inicie en los distintos niveles de ejecución[17]:

```
#update-rc.d dhcp defaults
```

➥ Para practicar:

Si disponemos de varias tarjetas de red en nuestra máquina Linux y es ella la encargada de dar servicio a Internet (usando el proxy-caché squid o las posibilidades de enrutar de los núcleos de Linux):

eth0 Servicio ADSL, si está en monupuesto será la IP pública. En caso contrario, una de la subred del router ADSL:

eth1 192.168.1.10 Red "administrativa"

eth2 192.168.2.10 Red de "alumnos"

y deseamos disponer de un servidor DCHP que no permita conexiones por la interfaz `eth0`, optaremos por:

1. Limitar el servicio a los interfaces `eth1` y `eth2`, para eso tendremos que modificar el fichero `/etc/sysconfig/dhcpd`[18], con la línea:

   ```
   DHCPDARGS="eth1 eth2"
   ```

   De esta forma impedimos que el servidor "escuche" en la interfaz de red que sale a Internet.

2. Ajustar el fichero `/etc/dhcpd.conf` a la nueva situación

   ```
   ddns-update-style interim;
   ignore client-updates;
   option domain-name-servers 80.58.0.33;
   subnet 192.168.1.0 netmask 255.255.255.0 {
   ```

[17] En general este comando se ejecutará de forma automática
[18] Sólo para Fedora, en Debian editar y ajustar el fichero `/etc/init.d/dhcp`

```
    #si usamos iptables
    #option routers 192.168.1.10;
    option subnet-mask 255.255.255.0;
    range dynamic-bootp 192.168.1.11 192.168.0.200;
    default-lease-time 21600;
    max-lease-time 43200;
}

subnet 192.168.2.0 netmask 255.255.255.0 {
    #si usamos iptables
    #option routers 192.168.2.10;
    option subnet-mask 255.255.255.0;
    range dynamic-bootp 192.168.2.100 192.168.2.199;
    default-lease-time 21600;
    max-lease-time 43200;
}
```

Ajustando los rangos de IPs a servir dinámicamente a nuestros intereses. Si además deseamos que la máquina con dirección MAC `00:A0:0C:13:5D:2D` tenga la IP fija 192.168.1.50 añadiríamos para la primera subred:

```
host ipfija{
       #Para asignarle también el nombre
       option host-name "ipfija.midominio.org";
       hardware ethernet 12:34:56:78:AB:CD;
       fixed-address 192.168.1.50;
}
```

Para que los cambios sean efectivos en Fedora usaremos[19]:

```
#service dhcpd reload
```

o paramos y rearrancamos[20]

```
#service dhpcd restart
```

Si junto a esta configuración usamos el Linux como enrutador, podemos conseguir que las máquinas obtengan una IP dinámica y que todas puedan salir a Internet usando el Linux (véase iptables en en la página 97)■

7.3.2. Configuración de los clientes:

Linux

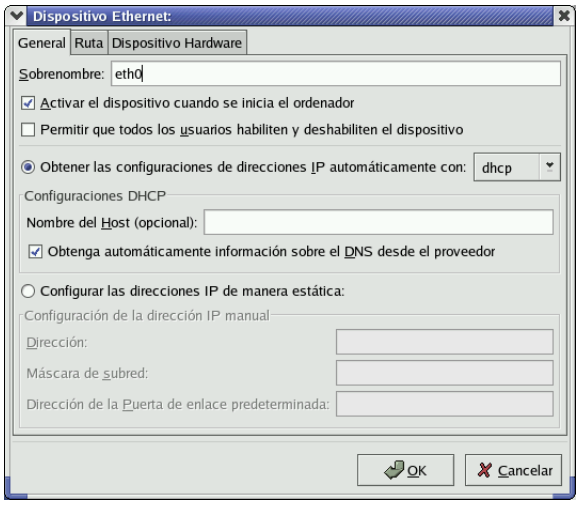
Fedora En modo gráfico no hay problema, ejecutaremos:

`#system-config-network`

y después de seleccionar **Dispositivos**→**Modificar** optaremos por

[19] `#/etc/init.d/dhcp restart`

[20] La opción restart suele ser conveniente, porque si está arrancado el servicio, primero lo para y luego lo arranca. En caso de no estar arrancado, la parada no hace nada y luego lo arranca.

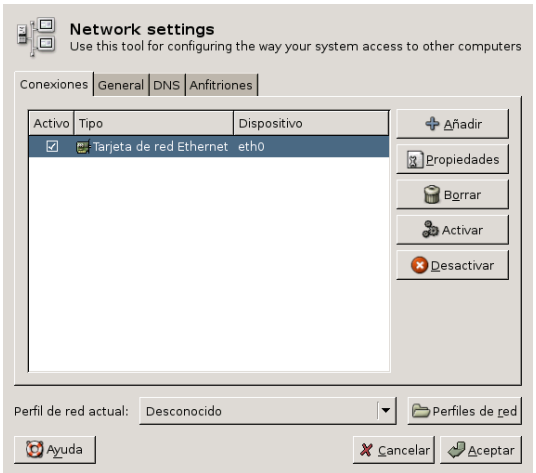


obtener las configuraciones de direcciones de IP automáticamemte con DHCP, obtener automáticamente información sobre DNS y, en su caso, rellenaremos el nombre de máquina que deseemos. Una vez guardados los cambios, habremos modificado el fichero como sigue (se puede hacer a mano, además la última línea sólo aparecerá si especificamos un nombre de host)

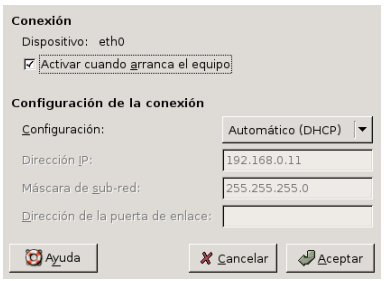
```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
USERCTL=no
PEERDNS=yes
TYPE=Ethernet
DHCP_HOSTNAME=cursolinux
```

Guadalinex 04

```
#network-admin
```



y tras optar por **Propiedades**, seleccionar que se inicie en el arranque usando DHCP



Los cambios realizados se traducen en el fichero

```
# cat /etc/network/interfaces
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet dhcp
        name Tarjeta de red Ethernet
```

➥ **Para practicar:**

Actualizar la IP de la máquina cliente ejecutando:

```
# /sbin/dhclient
```

Podemos comprobar la IP asignada con

```
# /sbin/ifconfig
```

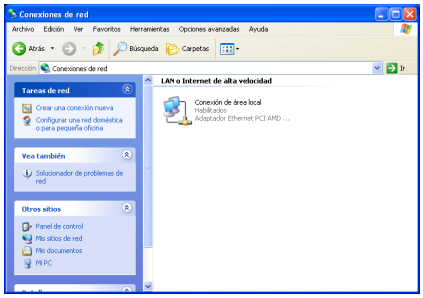
y la tabla de enrutamiento con:

```
# /bin/netstat -ar
```

■

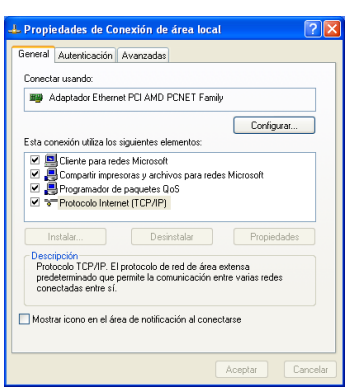
Windows

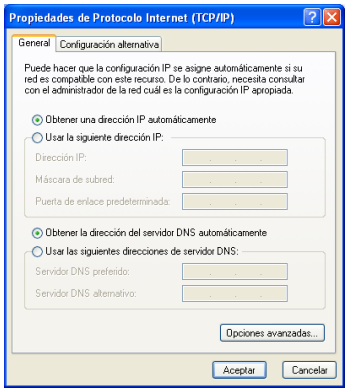
Para configurar la máquina Windows, optaremos por pulsar con el botón derecho[21] sobre **Conexiones de Red**



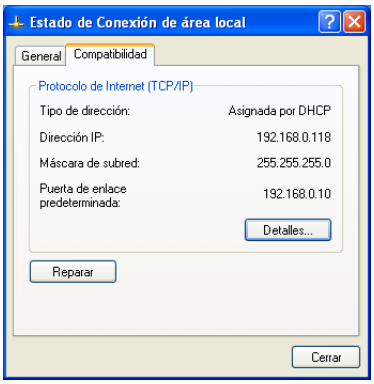
y en **Propiedades** optaremos por marcar las opciones de **Obtener la IP automáticamente** y pondremos como puerta de enlace la adecuada.

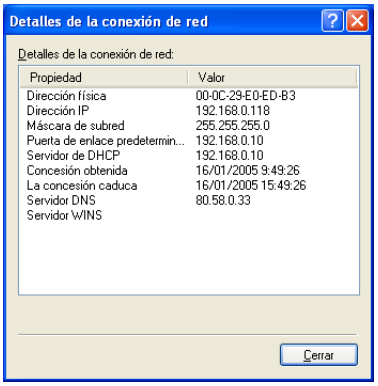
[21] Se parte de la idea de que es un Windows XP y que la red está configurada, al menos en cuanto a la cuestión *hardware*.





Podremos comprobar la configuración con el comando[22] `ipconfig` o en modo gráfico[23], por ejemplo, optando por **Estado** en la ventana emergente que aparece al pulsar con el botón derecho del ratón sobre **Conexiones de Red:**





[22] `winipcfg` en el 9x

[23] Los datos de la captura no se corresponden en su totalidad con los datos de ejemplo.

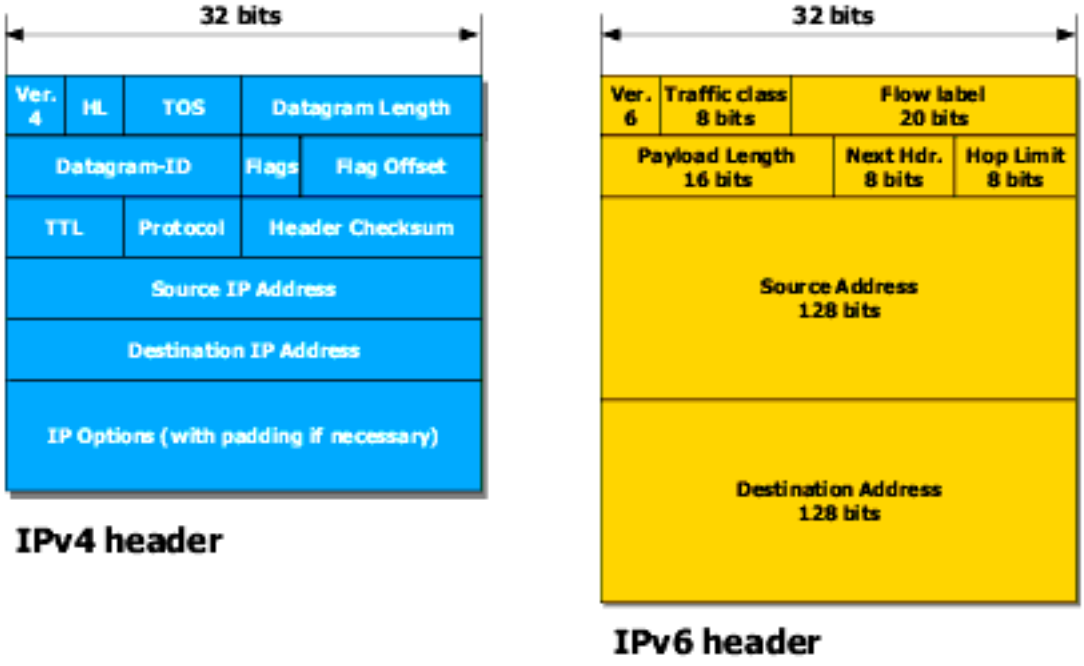
Apéndice A

IPv6

A.1. Introducción histórica

IPv6 es una evolución del protocolo IPv4, conocido simplemente como protocolo IP. El protocolo IP fue diseñado hace tiempo (el RFC consta desde enero de 1980) y desde su concepción han surgido nuevas necesidades y el rango de direcciones se ha demostrado insuficiente. El cambio más sustancial que se realiza en IPv6 se centra en el rediseño de las cabeceras, incluyendo un aumento en el tamaño de las direcciones de 32 a 128 bits, obteniendo así un direccionamiento muchísimo mayor.

Figura A.1: Comparación paquete IPv4 y paquete IPv6



Este aumento de direcciones está propiciado por la multitud de dispositivos que en el futuro pueden precisar de una dirección IP (frigoríficos, teléfonos móviles, ...).

La primera pregunta que surge es por qué se ha denominado IPv6 en lugar de IPv5, que

sería la continuación lógica de IPv4. La cabecera de un paquete IP tiene los 4 primeros dígitos reservados para indicar la versión del protocolo. Actualmente el 4 está reservado para IPv4 y el 5 está reservado para *Internet Stream Protocol v2*. Por consiguiente, el próximo número que queda libre para definir el protocolo es el 6, siendo éste el origen de IPv6.

La primera referencia que encontramos en el código fuente de linux fue en el kernel 2.1.8 en noviembre de 1996, siendo su autor Pedro Roque. Sin embargo, no fue hasta Octubre de 2000 cuando encontramos nuevas referencias a IPv6. Éstas estuvieron a cargo de un proyecto japonés, que se encargó de completar la especificación anterior, así como de actualizarla. Debido a su tamaño, en el kernel 2.4.* no se pudo añadir totalmente, por lo que hay algunos elementos que no están completamente definidos. En la versión del kernel 2.5.* se intentó introducir todas las extensiones disponibles, siendo ya en la 2.6.* donde aparece con toda su funcionalidad.

A modo de resumen, las características fundamentales de IPv6 serían:

- Mayor espacio de direcciones.
- "Plug & Play": Autoconfiguración de los interfaces de red.
- Seguridad intrínseca en el núcleo del protocolo (IPsec).
- Calidad de Servicio (QoS) y Clase de Servicio (CoS).
- *Multicast*: Envío de un mismo paquete a un grupo de receptores.
- *Anycast*: Envío de un paquete a un receptor dentro de un grupo.
- Paquetes IP eficientes y extensibles, sin que haya fragmentación en los *routers*, alineados a 64 bits (preparados para su procesado óptimo con los nuevos procesadores de 64 bits), y con una cabecera de longitud fija, más simple, que agiliza su procesado por parte del *router*.
- Posibilidad de paquetes con carga útil (datos) de más de 65.535 *bytes*.
- Encaminado (*routing*) más eficiente en el troncal (*backbone*) de la red, debido a una jerarquía de direccionamiento basada en la agregación.
- Renumeración y *multi-homing*, que facilita el cambio de proveedor de servicios.
- Características de movilidad.

A.1.1. ¿Cómo son las direcciones IPv6?

En IPv4 podemos asignar más de una IP a un interfaz de red si así lo requerimos (aliases, multicast). Manteniendo esta funcionalidad, IPv6 va más allá, permitiendo asignar tantas IP como queramos a una interfaz. El límite vendrá definido por la definición de la pila, con vista a evitar posibles ataques de denegación de servicio.

Las direcciones IPv6 son identificadores para interfaces y conjuntos de interfaces, clasificándose en tres tipos:

- *Unicast*: Identificador para una única interfaz. Un paquete enviado a una dirección unicast es entregado sólo a la interfaz identificada con dicha dirección. Es el equivalente a las direcciones IPv4 actuales.
- *Anycast*: Identificador para un conjunto de interfaces (normalmente pertenecen a diferentes nodos). Un paquete enviado a una dirección anycast es entregado en una (cualquiera) de las interfaces identificadas con dicha dirección (la más próxima, de acuerdo a las medidas de distancia del protocolo de encaminamiento). Permite que varias máquinas puedan ocuparse del mismo tráfico según una secuencia determinada (por el *routing*), si la primera "cae".

- *Multicast*: Identificador para un conjunto de interfaces (normalmente pertenecientes a diferentes nodos). Un paquete enviado a una dirección multicast es entregado a todas las interfaces identificadas por dicha dirección. La misión de este tipo de paquetes es evidente: aplicaciones de retransmisión múltiple (broadcast).

Vemos ahora más concretamente cómo es una dirección IPv6. Tal como se ha indicado anteriormente, las direciones IPv6 tienen una longitud de 128 bits, lo que en decimal equivale a un número de:

```
2^128-1: 340282366920938463463374607431768211455
```

Como puede observarse, estos números no son manejables. Una notación más simple sería en hexadecimal, partiendo de la representación binaria y agrupando los bits en grupos de 4:

```
2^128-1: 0xffffffffffffffffffffffffffffffff
```

Reducimos así la longitud de la dirección IPv6 a 32 caracteres. Aún así, esta representación es mejorable, ya que es fácil olvidar alguno de los caracteres. Se agrupan en bloques de 16 bits, separados por ':'

```
2^128-1: ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

De esta forma una dirección IPv6 podría ser la siguiente:

```
3ffe:ffff:0100:f101:0210:a4ff:fee3:9566
```

Simplificando aún más, los `0`'s iniciales de cada bloque se eliminan. De este modo, la dirección IPv6:

```
3ffe:ffff:0100:f101:0210:a4ff:fee3:9566
```

pasaría a ser

```
3ffe:ffff:100:f101:210:a4ff:fee3:9566
```

Otra simplificación más consiste en sustituir la secuencia de 16 bits compuesta por `0`'s por '`::`'. Esto se realizará una sola vez ya que en otro caso la representación no sería única.

```
3ffe:ffff:100:f101:0:0:0:1 -> 3ffe:ffff:100:f101::1
```

De esta forma, la mayor simplificación se produciría en la dirección `localhost` de IPv6:

```
0000:0000:0000:0000:0000:0000:0000:0001 -> ::1
```

A.1.2. Tipos de direcciones

Al igual que IPv4, las direcciones IPv6 pueden ser divididas en parte de red (64 bits superiores) y parte de host (64 bits inferiores), mediante el uso de máscaras de subred.

Direcciones con prefijos especiales

Dirección `localhost` Es la dirección para el interface de *loopback*, que en IPv4 era `127.0.0.1` y que en IPv6 es:

```
0000:0000:0000:0000:0000:0000:0000:0001
```

o en formato comprimido

```
::1
```

Los paquetes que tengan esta dirección como origen o destino nunca deben abandonar el *host*.

Direcciones sin especificar Es una dirección especial "cualquiera" que en IPv4 era `0.0.0.0` y que se utiliza en establecimiento de *sockets* y tablas de enrutamiento:

```
0000:0000:0000:0000:0000:0000:0000:0000
```

o en formato comprimido

```
::
```

Direcciones IPv6 con direcciones IPv4 embebidas Existen dos tipos de direcciones IPv6 que albergan direcciones IPv4 en su interior.

Las direcciones IPv4 mapeadas a IPv6 se utilizan en la creación de *sockets* por parte de demonios IPv6 que se conectarán a direcciones IPv4. Estas direcciones son definidas con un prefijo especial de longitud 96 bits, siendo `a.b.c.d` la dirección IPv4:

```
0:0:0:0:0:ffff:a.b.c.d/96
```

o en formato comprimido

```
::ffff:a.b.c.d/96
```

Así, la dirección IPv4 `192.168.0.32` sería `::ffff:192.168.0.32`

El otro tipo son las direcciones IPv4 compatibles con IPv6, utilizadas para el *tunneling* automático:

```
0:0:0:0:0:0:a.b.c.d/96
```

o en formato comprimido

```
::a.b.c.d/96
```

Prefijos que definen la red

Veamos a continuación los diferentes tipos de prefijos, que definirán tipos de direcciones.

Link local address Son direcciones especiales que son sólo válidas en el enlace de una interfaz. Utilizando esta dirección como destino, el paquete nunca pasará a través de un *router*.

Las direcciones locales de enlace han sido diseñadas para direccionar un único enlace para propósitos de auto-configuración (mediante identificadores de interfaz), descubrimiento del vecindario, o situaciones en las que no hay *routers*. Por tanto, los encaminadores no pueden retransmitir ningún paquete con direcciones fuente o destino que sean locales de enlace (su ámbito está limitado a la red local).

Tienen el siguiente formato:

```
fe8x: la única que se utiliza
fe9x:
feax:
febx:
```

donde `x` es un carácter hexadecimal, normalmente `0`.

Una dirección con este prefijo es encontrada en cada interfaz IPv6-*enabled* después de un estado de autoconfiguración.

Site local address Estas direcciones son similares a las direcciones de IPv4 utilizadas para redes privadas, aunque en este caso tienen la ventaja de que cualquiera que use este tipo de dirección puede usar los 16 bits para un máximo de 65.536 subredes. Es comparable a la subred `10.0.0.0/8` de IPv4.

Las direcciones locales de sitio permiten direccionar dentro de un *local site* u organización sin la necesidad de un prefijo global. Se configuran mediante un identificador de subred de 16 bits. Los encaminadores no deben retransmitir fuera del sitio ningún paquete cuya dirección fuente o destino sea *local site* (su ámbito está limitado a la red local o de la organización).

Otra ventaja añadida proviene de la facultad de asignar más de una direción IPv6 a un dispositivo IPv6, por lo que podría asignarse una IP global y una local.

Este tipo de direcciones tienen la siguiente estructura:

```
fecx: la más usada
fedx:
feex:
fefx:
```

donde `x` es un carácter hexadecimal, normalmente `0`.

A pesar que hay quien considera útiles este tipo de direcciones para su uso en laboratorios de pruebas, existe también la idea de que es mejor no hacer uso de ellas[1].

Tipo de dirección global unicast globales "agregables" Hay un tipo de dirección global definido (el primer diseño, llamado "basado en proveedor", se desechó hace varios años, quedando resquicios de la misma en algunos kernel obsoletos de linux). Su estructura es:

```
2xxx:
3xxx:
```

donde `x` son caracteres hexadecimales.

Direcciones Multicast Las direcciones multicast son utilizadas para servicios del mismo tipo. Siempre comienzan con `ffxy`, donde `xy` determinará un alcance distinto. El alcance multicast es un parámetro que indica la máxima distancia que puede viajar un paquete multicast desde la entidad que lo envió:

```
ffx1: node-local, los paquetes nunca abandonan el nodo.
ffx2: link-local, los paquete nunca son reenviados por rou-
ters, por lo que nunca abandonan el enlace especificado.
ffx5: site-local, los paquetes nunca abandonan el sitio.
ffx8: organization-local, los paquetes nunca abandonan la organización.
ffxe: global scope.
```

el resto de valores que pueden utilizarse están reservados.

Direcciones anycast Las direcciones anycast son un tipo especial de dirección utilizada para cubrir aspectos como el servidor DNS más cercano, el servidor DHCP más cercano, o grupos dinámicos similares.

Las direcciones se toman del espacio de direcciones unicast. El mecanismo anycast, desde el punto de vista del cliente, serán manejados por protocolos de *routing* dinámicos[2].

Un ejemplo de dirección anycast es la dirección anycast *subnet-router*. Asumiendo que un nodo tiene la siguiente dirección global IPv6:

```
3ffe:ffff:100:f101:210:a4ff:fee3:9566/64
```

[1] En inglés se denomina *deprecated*.

[2] Las direcciones anycast no pueden usarse como direcciones de origen, únicamente como direcciones destino.

La dirección anycast *subnet-router* se creará eliminando el subfijo (los 64 bits menos significativos) completamente:

```
3ffe:ffff:100:f101::/64
```

Tipos de direciones de nodos

Para aspectos de autoconfiguración y movilidad, se ha decidio usar los 64 bits más bajos como la parte de nodo de una dirección en la mayoría de los tipos de las direcciones. Aún así, cada subred puede albergar un gran número de direcciones.

Automáticas Con la autoconfiguración, la parte de host de la dirección es procesada convirtiendo la dirección MAC del interfaz (si está disponible), con el método EUI-64, a una dirección única IPv6. Si no hay dirección MAC disponible para este dispositivo (ocurre en dispositivos virtuales), algo distinto (dirección IPv4 o la dirección MAC del interfaz físico) es usado en su lugar.

Consideremos de nuevo el ejemplo anterior:

```
3ffe:ffff:100:f101:210:a4ff:fee3:9566
```

aquí, `210:a4ff:fee3:9566` es la parte de *host* y el resto es obtenido de la MAC `00:10:A4:E3:95:66` usando el IEEE-Tutorial EUI-64 (`http://standards.ieee.org/regauth/oui/tutorials/EUI64.html`), diseñado para identificadores EUI-48.

Manuales Para los servidores es probablemente más sencillo recordar una única dirección. Es posible asignar una dirección IPv6 adicional a un interfaz:

```
3ffe:ffff:100:f101::1
```

Para los sufijos como "`::1`" se requiere que el séptimo bit más significativo sea establecido a `0`. También se reservan otras combinaciones de bits para direcciones anycast.

A.1.3. ¿Estamos preparados para IPv6?

Antes de comenzara trabajar con IPv6 es preciso comprobar que el sistema sobre el que se trabajará está preparado para IPv6.

Soporte IPv6 en el kernel

La mayoría de las distribuciones modernas de Linux tiene el kernel preparado para soportar las funcionalidades IPv6, normalmente en forma de módulos. Estos módulos no siempre son cargados de forma automática.

La primera comprobación que se hará es comprobar que existe `/proc/net/if_inet6`. Su existencia nos indica que el módulo correspondiente está cargado. En caso de no encontrar esta entrada en `/proc` cargaremos el módulo de forma manual:

```
modprobe ipv6
```

En caso de éxito, podemos comprobar la correcta carga del módulo:

```
lsmod | grep ipv6
```

Es importante recordar que la descarga de este módulo del kernel no está soportado, pudiendo dar lugar a un error que deje nuestro sistema bloqueado.

Es posible cargar de forma automática el módulo IPv6, únicamente hay que añadir la línea siguiente a la configuración del cargador de módulos del kernel en `/etc/modules.conf`:

```
alias net-pf-10 ipv6
```

También es posible deshabilitar de forma automática la carga de dicho módulo, cambiando la línea anterior por:

```
alias net-pf-10 off
```

En el caso de que las comprobaciones anteriores hayan dado un resultado negativo, se pueden barajar las siguientes opciones:

- Actualizar la distribución a una que implemente una versión del kernel con soporte IPv6
- Compilar un nuevo kernel a partir de los fuentes

La primera opción es la recomendable, especialmente si no estamos habituados a las labores de recompilar el kernel.

Dispositivos de red con soporte IPv6

No todos los dispositivos de red están preparados para dar soporte a paquetes IPv6. Uno de los puntos que hay que tener en cuenta es que debido a la estructura de la capa de red que implementa el kernel, un paquete IPv6 no se reconoce por su número de cabecera IP (6 en lugar de 4). Se reconoce por el número de protocolo de la capa 2 del protocolo de transporte. Así, cualquier protocolo de transporte que no utilice este número de protocolo no puede despachar paquetes IPv6[3].

Utilidades de configuración dispositivos IPv6

net-tools El paquete `net-tools` incluye utilidades como `ifconfig` y `route` que ayudarán a configurar IPv6 en el interface.

Guadalinex

```
apt-get install net-tools
```

Fedora

```
rpm -Uvh net-tools-1.60-37.i386.rpm
```

iproute Este conjunto de herramientas permiten configurar la red a través de la utilidad ip. Proporciona más funcionalidades que el paquete `net-tools` pero no está tan documentado como éste.

Guadalinex

```
#apt-get install iproute
```

Fedora

```
#rpm -Uvh iproute-2.6.9-3.i386.rpm
```

Utilidades de chequeo IPv6

Una vez configurado el interfaz de red para tener soporte IPv6 es necesario tener acceso a las utilidades que nos permitan comprobar que nuestros paquetes IPv6 llegan a su destino.

[3]El paquete se transmitirá sobre el enlace, pero en la parte de recepción la entrega no funcionará.

ping IPv6 Normalmente este programa está incluido en el paquete iputils.

Guadalinex

```
#apt-get install iputils-ping
```

Fedora

```
#rpm -Uvh iputils-20020927-16.i386.rpm
```

Se utiliza para enviar ICMPv6 *echo request* que sirvan de comprobación.

```
root@guadalinex:~# ping6 -c 1 ::1
PING ::1(::1) 56 data bytes 64 bytes from ::1: icmp_seq=1 ttl=64 ti-
me=12.2 ms
--- ::1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, ti-
me 1ms rtt min/avg/max/mdev = 12.281/12.281/12.281/0.000 ms
```

Puede que si usuarios que no tengan permisos de root intentan ejecutarlo se encuentren con problemas. Esto puede ser debido a:

- `ping6` no está en la ruta del usuario
- `ping6` no se ejecuta adecuadamente debido a los permisos, será necesario ejecutar

  ```
  chmod u+s /usr/sbin/ping6
  ```

Usando direcciones de enlace local para un ping IPv6, el kernel no sabe a través de qué dispositivo (físico o virtual) debe enviar el paquete[4].

```
root@guadalinex:~# ping6 fe80::212:34ff:fe12:3456 connect: Invalid ar-
gument
```

En este caso sería necesario especificar el interface que vamos a utilizar:

```
root@guadalinex:~# ping6 -I eth0 -c 1 fe80::2e0:18ff:fe90:9205
PING  fe80::212:23ff:fe12:3456(fe80::212:23ff:fe12:3456) from
¬ fe80::212:34ff:fe12:3478 eth0: 56 data bytes
64 bytes from fe80::212:23ff:fe12:3456: icmp_seq=0 hops=64 ti-
me=445 usec --- fe80::2e0:18ff:fe90:9205 ping statistics --- 1 pac-
kets transmitted, 1 packets received, 0% packet loss round-
trip ¬ min/avg/max/mdev = 0.445/0.445/0.445/0.000 ms
```

En el caso de realizar ping sobre direcciones multicast IPv6, podemos encontrar los nodos con direcciones IPv6 activos haciendo `ping6` sobre la dirección multicast del enlace local *all-node*.

```
root@guadalinex:~# ping6 -
I eth0 ff02::1 PING ff02::1(ff02::1) from fe80:::2ab:cdff:feef:0123 eth0: 56 da-
ta bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.104 ms
64 bytes from fe80::212:34ff:fe12:3450: icmp_seq=1 ttl=64 ti-
me=0.549 ms (DUP!)
```

A diferencia de IPv4, donde las respuestas a un ping de la dirección de broadcast pueden ser deshabilitadas, el comportamiento en IPv6 es que no puede deshabilitarse excepto IPv6 *firewalling*.

[4]Cada dispositivo tiene una dirección de enlace local

traceroute/tracepath IPv6 Estas utilidades suelen estar incluidas en el paquete iputils.

Guadalinex

```
#apt-get install iputils-tracepath
```

Fedora

```
#rpm -Uvh iputils-20020927-16.i386.rpm
```

El comportamiento de `traceroute6` será similar al traceroute de IPv4, pero referido a direcciones IPv6. En el caso de `tracepath`, es igual que `traceroute`, pero va descubriendo las MTU en el camino al nodo destino.

tcpdump IPv6 Para obtener los datos que viajan por la red utilizaremos la misma herramienta que en IPv4, `tcpdump`. Esta herramienta tiene en las versiones actuales soporte para IPv6. Anteriormente se vio el uso de esta herramienta, así como de `ethereal`, por lo que nos remitimos a lo ya contado.

Clientes y servidores en IPv6

Algunos de los clientes que tienen soporte para IPv6[5] son:

- DNS
- Telnet
- OpenSSH
- Navegadores web[6]

Hay que tener en cuenta que si nuestro navegador web soporta IPv6, pero el acceso a internet lo realiza a través de un proxy IPv4 no accederemos a servidores `http` IPv6.

En el caso de programas servidores con soporte IPv6 habría que tener las mismas consideraciones que para los clientes, junto con una configuración adecuada. Los programas servidores BIND, xinetd, Apache2 son algunos de los que tienen soporte IPv6.

[5]Es posible que por defecto algunos no soporten IPv6, sería necesario recompilarlos con dicho soporte.

[6]Podemos comprobar fácilmente si nuestro navegador es IPv6, basta acceder a `http://www.kame.net`. Si tiene soporte IPv6 la tortuga se moverá, en otro caso permanecerá estática.

Bibliografía

[1] Linux IPv6 HOWTO. Peter Bieringer. http://www.bieringer.de/linux/IPv6/

[2] Ethereal User's guide. http://www.ethereal.com/

[3] Guía de Administración de Redes con Linux. Olaf Kirch y Terry Dawson. Proyecto LuCAS, traducción al español. http://es.tldp.org/Manuales-LuCAS/GARL2/garl2

[4] Introduction to Linux A Hands on Guide. Machtelt Garrels. http://tille.soti.org/training/tldp

[5] Introducción a la Administración de una Red Local basada en Internet. Charles L. Hedrick. http://es.tldp.org/Manuales-LuCAS/IAR/intro-admon-redes-v1.1.html

[6] Los HOWTO dedicados a las redes: http://www.tldp.org/HOWTO/HOWTO-INDEX/networking.html#NETGENERAL

[7] Rute User's Tutorial and Exposition. Paul Sheer. http://www.icon.co.za/~psheer/book/rute.html.gz

[8] Modelo de referencia OSI. Eduardo T. Sánchez Badillo http://www.geocities.com/txmetsb/el_modelo_de_referencia_osi.htm

[9] Tutorial y descripción técnica de TCP/IP. http://ditec.um.es/laso/docs/tut-tcpip

[10] Telecomunicaciones. http://www.eveliux.com/fundatel/menu_telecom.html

[11] Sitio web oficial de IPtables. http://www.netfilter.org/

[12] TCP/IP Illustrated, Volume 1 The Protocols. W. Richard Stevens. http://av.stanford.edu/books/tcpip/

[13] Enabling Technologies for E-Commerce. http://penguin.dcs.bbk.ac.uk/academic/technology/

[14] Protocolos TCP/IP. Juan Salvador Miravet Bonet. http://www4.uji.es/~al019803/Tcpip.htm

[15] IPTABLES: Manual práctico. Pello Xabier Altadill Izura. http://es.tldp.org/Manuales-LuCAS/doc-iptables-firewall

[16] YoLinux Tutorial - Linux Networking. http://www.yolinux.com/TUTORIALS/LinuxTutorialNetworking.html

[17] Iptables. Guía rápida. http://www.beeeeee.net/nautopia/linux/iptables.htm

[18] Guía de Administración de Redes con Linux. Olaf Kirch y Terry Dawson. Proyecto LuCAS, traducción al español. http://es.tldp.org/Manuales-LuCAS/GARL2/garl2

[19] Introduction to Linux A Hands on Guide. Machtelt Garrels. http://tille.soti.org/training/tldp

[20] Introducción a la Administración de una Red Local basada en Internet. Charles L. Hedrick. http://es.tldp.org/Manuales-LuCAS/IAR/intro-admon-redes-v1.1.html

[21] Los HOWTO dedicados a las redes: http://www.tldp.org/HOWTO/HOWTO-INDEX/networking.html#NETGENERAL

[22] Rute User's Tutorial and Exposition. Paul Sheer. http://www.icon.co.za/~psheer/book/rute.html.gz

[23] Modelo de referencia OSI. Eduardo T. Sánchez Badillo http://www.geocities.com/txmetsb/el_modelo_de_referencia_osi.htm

[24] Tutorial y descripción técnica de TCP/IP. http://ditec.um.es/laso/docs/tut-tcpip

[25] Telecomunicaciones. http://www.eveliux.com/fundatel/menu_telecom.html

[26] Sitio web oficial de IPtables. http://www.netfilter.org/

[27] TCP/IP Illustrated, Volume 1 The Protocols. W. Richard Stevens. http://av.stanford.edu/books/tcpip/

[28] Enabling Technologies for E-Commerce. http://penguin.dcs.bbk.ac.uk/academic/technology/

[29] Protocolos TCP/IP. Juan Salvador Miravet Bonet. http://www4.uji.es/~al019803/Tcpip.htm

[30] IPTABLES: Manual práctico. Pello Xabier Altadill Izura. http://es.tldp.org/Manuales-LuCAS/doc-iptables-firewall

[31] YoLinux Tutorial - Linux Networking. http://www.yolinux.com/TUTORIALS/LinuxTutorialNetworking.html

[32] Iptables. Guía rápida. http://www.beeeeee.net/nautopia/linux/iptables.htm