Redes en Linux Como (Previamente Net-3 Como)

Autor actual: Joshua Drake, {Poet}, *poet@linuxports.com*
Autores originales: Terry Dawson (autor principal), *terry@perf.no.itg.telstra.com.au*; Alessandro Rubini, *rubini@linux.it* (mantenimiento)
Traducido por: Ricardo Javier Cárdenes Medina, *a1402@serdis.dis.ulpgc.es*
v1.5, 20 de agosto de 1997, traducción del 3 de septiembre de 1999

Este *Cómo* es la base para entender la evolución de las capacidades de Linux para tratar con redes informáticas. Es el punto de partida para aprender todo sobre el mantenimiento de redes TCP/IP, la configuración de los archivos relacionados con la red, y hay un amplio capítulo sobre configuración de dispositivos físicos. En suma, un documento muy exhaustivo que merece la pena leer.

Índice General

1 Introducción.

El Sistema Operativo Linux se enorgullece de contar con una implementación de servicios de red basada en el núcleo, escrita casi por completo partiendo de cero. Su rendimiento en los núcleos recientes lo convierte en una alternativa válida incluso a el mejor de sus iguales. Este documento intenta describir cómo instalar y configurar el software de red de Linux y sus herramientas asociadas.

Esta es la primera entrega desde que LinuxPorts comenzó a hacerse cargo del documento. Quisiera decir antes que nada que deseamos que durante los próximos meses encuentre de utilidad este documento, y que seamos capaces de proporcionar información precisa y puntual en lo que respecta al tratamiento de redes con Linux.

Este documento, al igual que los otros `Comos` de los que nos encargamos, se va a convertir en algo muy diferente. Dentro de poco pasará a ser el *Redes en Linux Como* en lugar de ser sólo el *Net-3(4) Como*. Va a cubrir temas como PPP, VPN (Redes Privadas Virtuales), y otros...

2 Historia del documento

El *NET-FAQ* original fue escrito por Matt Welsh y Terry Dawson para responder preguntas frecuentes sobre las redes para Linux, un tiempo antes de que comenzara formalmente el Proyecto de Documentación de Linux. Cubría las últimas versiones de desarrollo del Linux Networking Kernel. El *NET-2-HOWTO* sobreseyó el *NET-FAQ* y fue uno de los documentos HOWTO originales del LDP. Cubría lo que se llamó versión 2 y posteriormente versión 3 del Linux Kernel Networking software. Este documento a su vez lo sobresee y sólo se refiere a la versión 4 del Linux Networking Kernel: más específicamente a las versiones `2.0.x` y `2.2.x` del núcleo.

Las versiones anteriores de este documento llegaron a ser bastante grandes a causa de la enorme cantidad de material que caía dentro de su ámbito. Para ayudar a resolver este problema se han producido unos cuantos Como que tratan con temas específicos sobre redes. Este documento proporciona referencias a dichos Como cuando sean relevantes y cubre aquellas áreas de las que aún no se habla en otros documentos.

2.1 Comentarios y sugerencias

Estamos interesados en que la gente nos proporcione sugerencias (que nos informen de cambios, errores, etc.). Por favor, póngase en contacto con nosotros en: *poet@linuxports.com*. De nuevo, si encuentra algo erróneo o cualquier cosa que desearía que fuese añadida, por favor, contacte con nosotros.

3 Cómo usar este documento.

Este documento está organizado de una manera vertical. La primera sección incluye material informativo y se la puede saltar si no es de su interés; lo que sigue es una discusión genérica sobre temas referentes a las redes, y debe asegurarse de entender esto antes de proceder con partes más específicas. El resto, información «específica de la tecnología», está agrupada en tres secciones principales: información relativa a Ethernet e IP, tecnologías pertinentes a hardware para PC de amplia difusión y tecnologías usadas sólo rara vez.

La metodología que nosotros sugerimos a la hora de leer este documento es la siguiente:

Lea las secciones genéricas

Estas secciones se aplican a toda, o casi toda tecnología descrita más adelante, y por tanto es muy importante que lo comprenda.

Estudie su red

Debería saber cómo está, o estará diseñada su red, y exactamente qué tipo de hardware y tecnología estará implementando.

Lea la sección 6 (IP y Ethernet) si está conectado directamente a una LAN o a Internet

Esta sección describe la configuración básica para Ethernet ya las varias posibilidades que ofrece Linux para las redes IP, como servicio de cortafuegos, encaminamiento avanzado, etc.

Lea la siguiente sección si está interesado en redes locales de bajo coste o en conexiones punto a punto

Esta sección describe PLIP, PPP, SLIP y RDSI, las tecnologías de más amplia difusión en estaciones de trabajo personales.

Lea las secciones específicas a tecnologías relacionadas con sus necesidades

Si sus necesidades difieren de una solución con IP y hardware común, la sección del final cubre detalles específicos a protocolos diferentes al IP y a hardware de comunicaciones peculiar.

Configure

Debería intentar configurar su red y tomar nota cuidadosamente de cualquier problema que tenga.

Busque más ayuda si la necesita

Si experimenta problemas que este documento no le ayude a resolver entonces lea la sección relativa a dónde encontrar ayuda o dónde informar de los errores.

¡Diviértase!

Trabajar en red es divertido, disfrútelo.

3.1 Convenciones usadas en el documento

No se han utilizado convenciones especiales, pero debería estar al corriente de la manera en que aparecen las órdenes. Siguiendo el estilo clásico en la documentación de Unix, cualquier orden que deba escribir en su intérprete de órdenes, estará prefijada por el símbolo «prompt». Este Como muestra el prompt `usuario$` para las órdenes que no necesitan que las ejecute el root. He escogido usar `root#` en lugar de un simple `#` para evitar confusiones con los trozos dispersos de guiones (*shell script*), donde la almohadilla (#) se usa para definir comentarios entre líneas.

Cuando se muestran «Opciones de Compilación del Núcleo», se presentan en el formato usado por `menuconfig`. Deberían ser comprensibles incluso si usted (al igual que yo) no está acostumbrado a usar `menuconfig`. Si tiene dudas al respecto del anidamiento de las opciones, ejecutar el programa le será de ayuda.

Fíjese en que los enlaces a otros documentos COMO son locales para que pueda acceder a las copias de los documentos del LDP que haya en el servidor al que usted está accediendo, en caso de que esté leyendo la versión html de este documento. Si no dispone del paquete completo de documentos, cada COMO puede ser obtenido en *www.linuxdoc.org* (directorio `/pub/Linux/HOWTO`) y en sus incontables servidores réplica (*mirror sites*).

4 Información general sobre las redes en Linux.

4.1 Breve historia del desarrollo del *Linux Networking Kernel*.

Desarrollar una nueva implementación núcleo de la pila para el protocolo tcp/ip que rinda tan bien como las implementaciones existentes no es una tarea fácil. La decisión de no portar una de las implementaciones existentes se tomó en un tiempo en que había cierta incertidumbre al respecto de si las implementaciones existentes serían lastradas con copyrights restrictivos por el precedente sentado por U.S.L. y en que había un gran entusiasmo por hacerlo diferente y quizás incluso mejor de lo que se había hecho.

El voluntario original para liderar el desarrollo del núcleo del código de red fue Ross Biro, *biro@yggdrasil.com*. Ross produjo una implementación simple e incompleta pero aprovechable en su mayor parte de rutinas que fueron completadas con un controlador de Ethernet para la interfaz de red WD-8003. Esto era suficiente para tener a mucha gente probando y experimentando con el software y algunas personas incluso intentaron conectar máquinas con esta configuración a conexiones reales en Internet. La presión dentro de la comunidad de Linux conduciendo el desarrollo de la implementación de la red estaba creciendo y, al final, el coste de una combinación de cierta presión aplicada sobre Ross y sus propios compromisos personales le sobrecargaron y se descolgó del puesto de desarrollador jefe. Los esfuerzos de Ross por iniciar el proyecto y la aceptación de la responsabilidad de producir realmente algo útil en tan controvertidas circunstancias fue lo que catalizó todo el trabajo futuro y por lo tanto supone un componente esencial en el éxito del producto actual.

Orest Zborowski, *obz@Kodak.COM*, produjo la interfaz original de programación de sockets BSD para el núcleo de Linux. Fue un gran paso adelante ya que permitió portar a Linux muchas de las aplicaciones de red existentes sin modificaciones serias.

Por aquel momento, Laurence Culhane, `loz@holmes.demon.co.uk` desarrolló los primeros controladores de dispositivo para la implementación del protocolo SLIP en Linux. Esto permitió a mucha gente que no tenía acceso a redes Ethernet experimentar con el nuevo software de red. De nuevo, hubo gente que cogió este controlador y lo puso en servicio para conectarse a Internet. Esto dio a mucha gente una idea de las posibilidades que podrían hacerse realidad si Linux tuviera un soporte total de red y creciera el número de usuarios usando y experimentando de forma activa el software de red que existía.

Una de las personas que también había estado trabajando activamente en la tarea de construir la implementación de red era Fred vam Kempen, `waltje@uwalt.nl.mugnet.org`. Después de un periodo de incertidumbre tras la renuncia de Ross al cargo de desarrollador jefe, Fred ofreció su tiempo y esfuerzos y aceptó el papel esencialmente sin oposición. Fred tenía planes ambiciosos sobre la dirección que quería que siguiese el software de red de Linux y orientó el progreso hacia esos objetivos. Fred desarrolló varias versiones del código de red llamado el núcleo de código *NET-2* (el código *NET* era el de Ross) que mucha gente pudo usar de manera más útil. Fred puso formalmente algunas innovaciones en la agenda de desarrollo, como la interfaz dinámica de dispositivo, la implementación del protocolo *Amateur Radio AX.25* y una interfaz de red diseñada más modularmente. El código *NET-2* de Fred fue usado por un gran número de entusiastas, que crecía continuamente según se iba corriendo la voz de que el software funcionaba. El software de red en esos momentos aún estaba constituido por un gran número de parches a la versión estándar del núcleo y no estaba incluido en la versión normal. El *NET-FAQ* y el subsecuente *NET-2-HOWTO* describían el por entonces relativamente complejo procedimiento de ponerlo todo en marcha. Fred se concentraba en desarrollar innovaciones frente a las implementaciones estándar de red y eso estaba llevando tiempo. La comunidad de usuarios estaba impacientándose por algo que funcionase eficazmente y satisficiera al 80% de usuarios y, como con Ross, la presión sobre Fred como desarrollador en jefe creció.

Alan Cox, `iialan@www.uk.linux.org` propuso una solución al problema diseñada para resolver la situación. Propuso que él podría coger el código *NET-2* de Fred para quitar fallos, haciéndolo eficaz y estable para que satisficiera al impaciente usuario de base al tiempo que retiraba esa presión de Fred permitiéndole continuar con su trabajo. Alan se puso a ello, obteniendo buenos resultados y su primera versión del código de red de Linux fue llamada it/Net-2D(ebugged)/. El código funcionaba bien en muchas configuraciones típicas y el usuario de base estaba feliz. Alan puso claramente sus propias ideas y habilidad para contribuir al proyecto y como consecuencia comenzaron a aparecer muchas discusiones relativas a la dirección del código *NET-2*. Esto desarrolló dos tendencias distintas dentro de la comunidad de red de Linux, una que tenía la filosofía «hazlo funcionar primero, y luego hazlo mejor» y la otra «hazlo mejor primero». Linus terminó arbitrando y ofreciendo su apoyo a los esfuerzos de desarrollo de Alan e incluyó el código de Alan en la distribución estándar de las fuentes del núcleo. Esto puso a Fred en una posición difícil. Cualquier desarrollo continuado podría debilitar la gran base de usuarios usando y probando activamente el código y eso significaría que los progresos serían lentos y difíciles. Fred continuó trabajando un tiempo y a la larga acabó dejándolo y Alan se convirtió en el nuevo jefe del esfuerzo de desarrollo de red para Linux.

Donald Becker, `becker@cesdis.gsfc.nasa.gov` reveló pronto su talento para los aspectos de bajo nivel de las redes y produjo un amplio abanico de controladores de red. Casi todos los incluidos en los núcleos actuales fueron desarrollados por él. Hay más gente que ha hecho contribuciones significativas, pero el trabajo de Donald es prolífico y eso le garantiza una mención especial.

Alan continuó refinando el código *NET-2-Debugged* durante un tiempo mientras trabajaba en solucionar algunos de los problemas que quedaban sin mirar en la lista `PORHACER`. Por la época en que al código fuente `1.3.*` del núcleo les estaban creciendo los dientes, el código de red había migrado a la entrega *NET-3* que es en la que las versiones actuales están basadas. Alan trabajó en muchos aspectos diferentes del código de red y con la asistencia de mucha otra gente con talento de la comunidad de red de Linux hizo crecer el código en todas direcciones. Alan produjo dispositivos de red dinámicos y las primeras implementaciones de los estándares AX.25 e IPX. Alan ha continuado jugueteando con el código, estructurándolo y mejorándolo lentamente hasta el estado en que se encuentra hoy día.

Michael Callahan, *callahan@maths.ox.ac.uk* y Al Longyear *longyear@netcom.com* aportaron la implementación del protocolo PPP, lo cual fue crítico para incrementar el número de gente que usaba activamente Linux para trabajar en red.

Jonathon Naylor, *sn@cs.nott.ac.uk* ha contribuido mejorando significativamente el código de AX.25 de Alan, añadiendo los protocolos NetRom y Rose. La implementación de AX.25/NetRom/Rose es en sí misma bastante significativa, porque ningún otro sistema operativo puede enorgullecerse de trabajar de forma nativa con estos protocolos aparte de Linux.

Por supuesto, ha habido centenares de otras personas que han hecho contribuciones significativas al desarrollo del software de red de Linux. A algunas de estas personas las encontrará más adelante en la sección de tecnologías específicas; otras personas han contribuido con módulos, controladores, correcciones de errores, sugerencias, informes de pruebas y apoyo moral. En cada caso, cada uno puede decir que ha puesto su granito de arena y que ofreció lo que pudo. El código de red del núcleo de Linux es un excelente ejemplo de los resultados que se pueden obtener del anárquico estilo de desarrollo de Linux. Si aún no se ha sorprendido, lo hará pronto. El desarrollo no se ha detenido.

4.2 Recursos referentes al tratamiento de redes con Linux.

Hay varios sitios en los que puede encontrar información acerca de la implementación de red de Linux.

Tiene a su disposición un montón de asesores. Podrá encontrar un listado en la *LinuxPorts Consultants Database*, *http://www.linuxports.com*

Alan Cox, quien en estos momentos se encarga de mantener el código de red del núcleo de Linux tiene una página WWW con los últimos titulares y eventos de interés relativos al estado de desarrollo de las funcionalidades presentes y futuras en cuanto al código de red linux en: *www.uk.linux.org*.

Otro buen sitio es un libro escrito por Olaf Kirch titulado la `Guía para Administradores de Redes`. Es parte del Proyecto de Documentación de Linux *http://www.linuxdoc.org* y puede leerlo de forma interactiva en su versión HTML *http://metalab.unc.edu/LDP/LDP/nag/nag.html*, u obtenerlo en varios formatos mediante FTP (y en castellano) desde el archivo del proyecto LuCAS *ftp://lucas.hispalinux.es/pub/LuCAS/Manuales-LuCAS/GARL*. El libro de Olaf es bastante extenso y procura una buena iniciación de alto nivel a la configuración de redes en Linux.

Hay un grupo de noticias en la jerarquía de noticias de Linux dedicado a las redes y a materias relacionadas: *comp.os.linux.networking*

Hay una lista de correo a la que se puede suscribir, donde hacer preguntas relacionadas al trabajo en redes con Linux. Para suscribirse debe mandar un mensaje:

```
A: majordomo@vger.rutgers.edu
Asunto: cualquier cosa
Mensaje:

subscribe linux-net
```

En las diferentes redes de IRC suele haber canales `#linux` en los que suele haber gente que responde preguntas sobre linux y redes.

Por favor, cuando vaya a informar de cualquier problema, recuerde incluir todos los detalles que pueda relevantes al respecto. Debería informar, específicamente, de las versiones de los programas que esté usando, sobre todo la versión del núcleo, la versión de herramientas como `pppd` o `dip` y la naturaleza exacta del problema que está experimentando. Esto significa tomar nota de la sintaxis exacta de cualquier mensaje de error que reciba y de cualquier orden que esté ejecutando.

4.3 Dónde conseguir información sobre redes no específica de Linux.

Si está buscando información básica de aprendizaje sobre redes tcp/ip en general, entonces le recomiendo que eche un vistazo a los siguientes documentos:

TCP/IP Introduction

este documento está tanto en versión texto *ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc* como en versión PostScript *ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps*.

TCP/IP Administration

este documento está tanto en versión texto *ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc* como en versión PostScript *ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps*.

Si busca información algo más detallada sobre redes tcp/ip entonces le recomiendo:

> *Internetworking with TCP/IP, Volume 1: principles, protocols and architecture*, de Douglas E. Comer, ISBN 0-13-227836-7, Prentice Hall publications, Tercera edición, 1995.

Si quiere aprender a escribir aplicaciones de red en entornos compatibles Unix entonces también le recomiendo:

> *Unix Network Programming*, by W. Richard Stevens, ISBN 0-13-949876-1, Prentice Hall publications, 1990.

También debería probar el grupo de noticias *comp.protocols.tcp-ip*. Otra fuente importante de información técnica específica relativa a la Internet y al conjunto de protocolos tcp/ip son los RFC. RFC es un acrónimo de *Request For Comments* y es el medio estándar de enviar y documentar los protocolos estándar de Internet. Hay muchos sitios de donde tomar los RFC. Muchos de estos sitios son de FTP y otros proporcionan acceso por World Wide Web con un buscador asociado que le permite buscar palabras clave en la base de datos de RFC.

Un posible lugar donde encontrar lo RFC es la base de datos de Nexor *http://pubweb.nexor.co.uk/public/rfc/index/rfc.html*.

5 Información genérica sobre la configuración de redes.

Las siguientes subsecciones las necesitará para saber y comprender ciertas cosas antes de que intente configurar la red. Son principios fundamentales que se aplican independientemente de la naturaleza exacta de la red que desee organizar.

5.1 ¿Qué necesito para comenzar?

Antes de que empiece a construir o configurar la red necesita saber algunas cosas. Las más importantes son:

5.1.1 Código fuente del núcleo.

Antes que nada:

La mayoría de las distribuciones vienen por defecto con el soporte de red activado, por lo cual no habrá que recompilar el núcleo. Si tiene hardware común, debería irle bien. Por ejemplo, tarjetas de red 3COM, NE2000, o Intel. Sin embargo proporcionamos la siguiente información por si necesita actualizar el núcleo.

Como puede ser que el núcleo que está ejecutando no esté preparado para los tipos de red o tarjetas que desee usar, probablemente necesitará las fuentes del núcleo para recompilarlo con las opciones apropiadas.

Siempre puede obtener la última versión de CDROM.com *ftp://ftp.cdrom.com/pub/linux/sunsite/kernel.org/pub/linux/kernel*. Este no es el sitio oficial, pero tiene un GRAN ancho de banda y permite que MUCHOS usuarios conecten simultáneamente. El sitio oficial es `kernel.org`, pero use el anterior si tiene la posibilidad. Por favor, recuerde que `ftp.kernel.org` está seriamente sobrecargado. Use un servidor réplica.

Normalmente las fuentes del núcleo se instalarán en el directorio `/usr/src/linux`. Para más información sobre cómo aplicar parches y construir el núcleo debería leer el *Kernel Como*. Para más información sobre cómo configurar módulos del núcleo debería leer el *Modules mini-Howto*. Además, el fichero `README` que hay en las fuentes del núcleo y el directorio `Documentation` son muy ilustrativos para el lector intrépido.

A menos que se diga específicamente lo contrario, recomiendo que empiece con la versión estándar del núcleo (la que tenga un número par como segundo dígito del número de versión). Las versiones de desarrollo del núcleo (las que tienen el segundo dígito impar) podrían tener algunos cambios estructurales u otros que podrían causar problemas con otro software en su sistema. Si no está seguro de que pueda resolver ese tipo de problemas sumado al potencial de que haya otros errores de software, no los use.

Por otro lado, algunas de las capacidades aquí descritas han sido introducidas durante el desarrollo de los núcleos `2.1`, por lo que tendrá que tomar una decisión: puede mantenerse en `2.0` mientras espera por el `2.2` y por una distribución con cada herramienta actualizada, o puede coger un `2.1` y buscar los diversos programas necesarios para explotar las nuevas capacidades. En el momento de escribir este Como, en Agosto de 1998, el núcleo disponible es el `2.1.115` y se espera que el `2.2` aparezca pronto.

`Nota del Traductor`: En el momento en que acabó la traducción de este Como, en septiembre de 1999, las versiones disponibles del núcleo son la `2.2.12` (estable) y la `2.3.16` (desarrollo). Además, las principales distribuciones han puesto al día sus herramientas para tratar las capacidades de la serie `2.1` y superiores.

5.1.2 Herramientas de red actualizadas.

Las herramientas de red son los programas que usted usa para configurar los dispositivos de red de linux. Estas herramientas permiten asignar direcciones a dispositivos y configurar rutas, por ejemplo.

La mayoría de distribuciones modernas de Linux están dotadas con las herramientas de red, por lo que si ha instalado Linux a partir de una distribución y no ha instalado las herramientas de red, debería hacerlo.

Si no ha instalado a partir de una distribución entonces necesitará las fuentes para compilar las herramientas usted mismo. No es difícil.

Las herramientas de red las mantiene ahora Bernd Eckenfelds y están disponibles en:

ftp://ftp.inka.de/pub/comp/Linux/networking/NetTools/ y están replicadas en: *ftp://ftp.uk.linux.org/pub/linux/Networking/base/*.

También puede encontrar los últimos paquetes de RedHat en *ftp://ftp.cdrom.com/pub/linux/redhat/redhat-6.0/i386/base/net-tools-1.51-3.i386.rpm*.

Para Debian, los paquetes que traen las principales herramientas son los paquetes `hostname`, `netbase` y `netstd`, que encontrará en *ftp://ftp.debian.org/debian/dists/stable/main/binary-i386/base/*.

Asegúrese de que elige la versión que más se ajuste al núcleo que desee usar y siga las instrucciones del paquete para instalarlo.

Para instalar y configurar la versión actual, —en el momento de traducirse esto— esto necesitará hacer lo siguiente:

```
usuario% cd /usr/src
usuario% tar xvfz net-tools-x.xx.tar.gz
usuario% cd net-tools-x.xx
usuario% make config
usuario% make
root# make install
```

O si no, use los paquetes de su distribución. Por ejemplo, con RedHat:

```
root# rpm -U net-tools-x.xx-y.i386.rpm
```

y con Debian:

```
root# dpkg -i hostname_x.xx-y.yy.deb
root# dpkg -i netbase_x.xx-y.yy.deb
root# dpkg -i netstd_x.xx-y.yy.deb
```

En los anteriores ejemplos, `x.xx` se refiere a la versión de las herramientas, e `y.yy` a la revisión de los correspondientes paquetes.

Si además va a configurar un cortafuegos o a usar la característica de IP Masquerade, necesitará `ipfwadm`. La última versión la puede obtener en: *ftp:/ftp.xos.nl/pub/linux/ipfwadm*. De nuevo se encontrará con más de una versión disponible. Asegúrese de coger la versión que más se ajuste a su núcleo. Tenga en cuenta que las capacidades de cortafuegos de Linux cambiaron durante el desarrollo `2.1` y `ipfwadm` ha sido sustituida por `ipchains` en la versión `2.2` del núcleo. `ipfwadm` sólo vale para la versión `2.0` del núcleo. Se sabe de estas distribuciones que se ajustan a versiones `2.0` o anteriores del núcleo:

- Redhat 5.2 o anteriores
- Caldera antes de la versión 2.2
- Slackware antes de las versiones 4.x
- Debian antes de las versiones 2.x

Para instalar y configurar la versión actual en el momento de escribir esto necesita leer el *IPChains Howto*, disponible en el Proyecto de Documentación de Linux *http://www.linuxdoc.org*

Tenga en cuenta que si tiene una versión `2.2` (o `2.1` de las últimas) del núcleo, `ipfwadm` no es la herramienta correcta para configurar un cortafuegos. Esta versión del *NET-3-HOWTO* todavía no trata con la nueva configuración de cortafuegos. Si necesita información más detallada sobre `ipchains`, por favor, diríjase al documento mencionado anteriormente.

5.1.3 Aplicaciones de red.

Las aplicaciones de red son programas como `telnet` y `ftp` y sus respectivos programas servidores. David Holland estuvo manteniendo una distribución de las más comunes de la que ahora se ocupa *:net-bug@ftp.uk.linux.org*. Puede obtenerlas en: *ftp://ftp.uk.linux.org/pub/linux/Networking/base*.

5.1.4 Introducción a las direcciones IP.

Las direcciones del Protocolo Internet (IP) están compuestas por cuatro bytes. La convención es escribir estas direcciones en la denominada «notación decimal puntuada» (*dotted decimal notation*). De esta forma cada byte es convertido en un número decimal (0-255), despreciando los ceros a la izquierda a menos que el número en sí sea cero. Por convención, cada interfaz de una máquina o encaminador tiene una dirección IP. Es válido usar la misma IP para cada interfaz de una sola máquina en algunas circunstancias, pero normalmente cada interfaz tiene su propia dirección.

Las Redes basadas en *Internet Procotol* son secuencias contiguas de direcciones IP. Todas las direcciones dentro de una red tienen un número de dígitos de en común. A la porción de la red que es común a todas las direcciones llama la «porción de la red». Los dígitos restantes son llamados «porción de la máquina». 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. Consideremos el siguiente ejemplo:

```
---------------------   ---------------
Dirección Host          192.168.110.23
Máscara de red          255.255.255.0
Porción de red          192.168.110.
Porción de Host         .23
---------------------   ---------------
Dirección de Red        192.168.110.0
Dirección de Difusión   192.168.110.255
---------------------   ---------------
```

Cualquier dirección a la que se aplique una operación *and* de bits con su máscara de red, revelará la dirección de la red a la que pertenece. La dirección de red es por tanto siempre el menor número de dirección dentro de el rango de la red y siempre tiene la porción de máquina codificada toda con ceros.

La dirección «de difusión» (*broadcast*) es una especial a la que escucha cada máquina en la red además de a la suya propia. Esta dirección es a la que se envían los datagramas si se supone que todas las máquinas de la red lo deben recibir. Ciertos tipos de datos, como la información de encaminamiento y los mensajes de aviso son transmitidos a la dirección de difusión para que cada estación en la red pueda recibirlo simultáneamente. Hay dos estándares usados comúnmente al respecto de la dirección de difusión. El más ampliamente aceptado es el de usar la dirección más alta posible en la red. En el ejemplo anterior sería `192.168.110.255`. Por alguna razón, otras estaciones han adoptado la convención de usar las direcciones de red como direcciones de difusión. En la práctica no importa mucho cual use, pero asegúrese de que cada máquina en la red está configurada con la misma.

Por razones administrativas, durante el desarrollo inicial del protocolo IP se formaron, de forma arbitraria, algunos grupos de direcciones como redes, y estas redes se agruparon en las llamadas «clases». Estas clases proporcionan un cierto número de redes de tamaño estándar que pueden ser reservadas. Los rangos reservados son:

```
-----------------------------------------------------------
| Clase   | Máscara de    | Direcciones de red            |
| de red  | red           |                               |
-----------------------------------------------------------
|    A    | 255.0.0.0     | 0.0.0.0    - 127.255.255.255 |
|    B    | 255.255.0.0   | 128.0.0.0  - 191.255.255.255 |
|    C    | 255.255.255.0 | 192.0.0.0  - 223.255.255.255 |
|Multicast| 240.0.0.0     | 224.0.0.0  - 239.255.255.255 |
-----------------------------------------------------------
```

Las direcciones que deberá usar dependen de lo que vaya a hacer exactamente. Puede que tenga que realizar varias de las siguientes actividades para obtener las direcciones que necesite:

Instalar una máquina Linux en una red IP existente

Si desea instalar una máquina Linux en una red IP existente entonces debería contactar con los administradores de la red y preguntarles por la siguiente información:

- Dirección IP del Host
- Dirección IP de la red
- Dirección IP de broadcast
- Máscara de red IP
- Dirección del encaminador (router)
- Dirección del Servidor de Nombre de Dominio (DNS)

Debería configurar entonces el dispositivo de red Linux con esos detalles. No puede inventarlos y esperar que la configuración funcione.

Construir una nueva red propia que nunca conectará con Internet

Si está construyendo una red privada y no tiene intención de conectar nunca esa red a Internet entonces puede elegir las direcciones que quiera. De todas maneras, por razones de seguridad y consistencia, se han reservado algunas direcciones IP de red específicamente para este propósito. Están descritas en el RFC1597 y son las que siguen:

```
-----------------------------------------------------------
|      DIRECCIONES RESERVADAS PARA REDES PRIVADAS         |
-----------------------------------------------------------
| Clase   | Máscara de    | Direcciones de red            |
| de red  | red           |                               |
-----------------------------------------------------------
|    A    | 255.0.0.0     | 10.0.0.0    - 10.255.255.255  |
|    B    | 255.255.0.0   | 172.16.0.0  - 172.31.255.255  |
|    C    | 255.255.255.0 | 192.168.0.0 - 192.168.255.255 |
-----------------------------------------------------------
```

Primero debería decidir cuán grande quiere que sea su red para entonces elegir tantas direcciones como necesite.

5.2 ¿Dónde debería poner las órdenes de configuración?

Hay unas pocas opciones a elegir para el procedimiento de arranque del sistema Linux. Después de que carga el núcleo, siempre ejecuta un programa llamado `init`. El programa `init` lee entonces el fichero de configuración llamado `/etc/inittab` y comienza el proceso de arranque. Hay unos pocos `init` diferentes, aunque todo el mundo está ahora convergiendo al modelo *SystemV*, desarrollado por Miguel van Smoorenburg.

A pesar de que el programa `init` sea el mismo, la configuración del arranque del sistema está organizada de manera diferente en cada distribución.

Normalmente el fichero `/etc/inittab` contiene una entrada que dice algo como:

```
si::sysinit:/etc/init.d/boot
```

Esta línea especifica el nombre del fichero de guión de ejecución (script) que controla la secuencia de carga. Este fichero es algo así como el `AUTOEXEC.BAT` en MS-DOS.

El guión de arranque suele llamar a otros, y a menudo la red se configura dentro de alguno de éstos.

La siguiente tabla puede ser usada como guía para su sistema:

```
----------------------------------------------------------------------
Distrib. |Interfaz Configuración/Encaminado  |Iniciación del Servidor
----------------------------------------------------------------------
Debian   | /etc/init.d/network               | /etc/rc2.d/*
----------------------------------------------------------------------
Slackware| /etc/rc.d/rc.inet1                | /etc/rc.d/rc.inet2
----------------------------------------------------------------------
RedHat   | /etc/rc.d/init.d/network          | /etc/rc.d/rc3.d/*
----------------------------------------------------------------------
```

Fíjese en que Debian y Red Hat usan un directorio entero de guiones que «levantan» los servicios del sistema (y normalmente la información no se encuentra en esos archivos; por ejemplo, el sistema de Red Hat almacena toda la configuración del sistema en ficheros dentro de `/etc/sysconfig`, de donde es leída por los guiones de carga). Si quiere comprender los detalles del proceso de arranque del sistema, le sugiero que examine `/etc/inittab` y la documentación que acompaña a `init`. Linux Journal va a publicar (o lo ha hecho ya) un artículo tratando la iniciación del sistema, y este documento mantendrá una referencia a él tan pronto como esté disponible en la red.

La mayoría de distribuciones modernas incluyen algún programa que le permita configurar la mayoría de interfaces de red. Si tiene una de éstas entonces debería ver si hace lo que usted quiere antes de acudir a la configuración manual.

```
--------------------------------------------
Distrib.  | Programa de configuración de red
--------------------------------------------
RedHat    | /sbin/netcfg
Slackware | /sbin/netconfig
--------------------------------------------
```

5.3 Creación de las interfaces de red.

En muchos sistemas operativos Unix los dispositivos de red tienen correspondencias en el directorio `/dev`. Esto **no** pasa en Linux. Los dispositivos de red se crean de forma dinámica y por tanto no requieren de la presencia de ficheros de dispositivo.

En la mayoría de los casos los dispositivos de red son creados automáticamente por el controlador de dispositivos mientras se inicia y localiza el hardware. Por ejemplo, el controlador Ethernet crea interfaces `eth[0..n]` secuencialmente según va encontrado tarjetas Ethernet. La primera tarjeta que encuentra es `eth0`, la segunda `eth1`, etc.

Sin embargo, en algunos casos, de los que `slip` y `ppp` son ejemplos notables, los dispositivos de red son creados por la acción de algún programa de usuario. Se aplica la misma numeración secuencial de dispositivos, pero no se crean al arrancar. La razón es que al contrario que con los dispositivos Ethernet, el número de dispositivos `ppp` o `slip` puede variar durante la actividad de la máquina. Estos casos serán cubiertos con más detalle en secciones posteriores.

5.4 Configuración de una interfaz de red.

Cuando tenga todos los programas necesarios y su dirección e información de red, puede configurar la interfaz de red. Cuando hablamos de configurar una interfaz de red nos referimos al proceso de asignar direcciones apropiadas a un dispositivo y asignar valores adecuados a otros parámetros configurables. El programa usado más comúnmente para hacer esto es la orden `ifconfig` (**i**nter**f**ace **config**ure).

Lo normal será usar una orden similar a la siguiente:

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

En este caso estoy configurando la interfaz Ethernet `eth0`, con dirección IP `192.168.0.1` y máscara de red `255.255.255.0`. El '`up` del final de la orden le dice al interfaz que debería activarse, pero normalmente se puede omitir, ya que es el valor por defecto. Para desactivar una interfaz, simplemente tiene que ejecutar `ifconfig eth0 down`.

El núcleo asume ciertas cosas cuando configura interfaces. Por ejemplo, puede especificar la dirección de red y difusión de una interfaz, pero si usted no lo hace, como en mi ejemplo anterior, entonces el núcleo hará una suposición razonable de cuáles deberían ser, basándose en la máscara que se le proporciona; si tampoco indica la máscara, entonces partirá de la clase de la dirección IP configurada. En mi ejemplo, el núcleo asumirá que se va a configurar una red clase C en la interfaz y establece una dirección de red `192.168.0.0` y una dirección de difusión `192.168.0.255`. Hay otras muchas opciones para la orden `ifconfig`. Las mas importantes son:

up

esta opción activa una interfaz (y es la opción por defecto).

down

esta opción desactiva una interfaz.

[-]arp

esta opción activa o desactiva el uso del protocolo de resolución de dirección (**a**ddress **r**esolution **p**rotocol) sobre la interfaz

[-]allmulti

esta opción activa o desactiva la recepción de todos los paquetes multicast por hardware. El multicast por hardware permite que varios grupos de interfaces reciban paquetes remitidos a destinos especiales. Esto puede ser de importancia si está usando aplicaciones como videoconferencia, pero normalmente no se usa.

mtu N

este parámetro le permite especificar la MTU del dispositivo.

netmask <direc>

este parámetro le permite asignar la máscara de la red a la que pertenece el dispositivo.

irq <direc>

este parámetro sólo trabaja con ciertos tipos de hardware, y permite especificar la IRQ del dispositivo.

[-]broadcast [direc]

este parámetro le permite activar y asignar la acepción de datagramas destinados a la dirección de difusión, o desactivarla por completo.

[-]pointopoint [direc]

este parámetro le permite asignar la dirección de la máquina en el otro extremo de un enlace punto a punto, como en SLIP o PPP.

hw <tipo> <direc>

este parámetro le permite asignar la dirección del hardware de ciertos tipos de dispositivos de red. Esto no suele ser útil para Ethernet, pero lo es para otras redes como AX.25.

Puede usar la orden `ifconfig` sobre cualquier dispositivo de red. Algunos programas de usuario, como `pppd` y `dip` configuran automáticamente los dispositivos de red al crearlos, por lo que es innecesario el uso de `ifconfig` con ellos.

5.5 Configuración del sistema de resolución de nombres (*Name Resolver*)

El sistema de *resolución de nombres* es una parte de la biblioteca estándar de Linux. Su función principal es proporcionar un servicio para convertir las denominaciones «amistosas con el hombre» de las máquinas como `ftp.funet.fi` a direcciones IP «amigables para la máquina» como `128.214.248.6`.

5.5.1 ¿Qué hay en un nombre?

Posiblemente esté familiarizado con la apariencia de los nombres de máquina de Internet, pero puede que no entienda como se construyen, o como se «desconstruyen». Los nombres de dominio de Internet son jerárquicos por naturaleza; esto es, tienen una estructura en árbol. Un *dominio* es una familia, o grupo de nombres. Un *dominio* puede ser fragmentado en *subdominios*. Un «dominio de nivel superior» (*toplevel domain*) es un dominio que NO es un subdominio. Los Dominios de Nivel Superior están especificados en el *RFC-920*. Algunos ejemplos de los más comunes son:

COM

Organizaciones Comerciales

EDU

Organizaciones Educativas

GOV

Organizaciones Gubernamentales

MIL

Organizaciones Militares

ORG

Otras organizaciones

NET

Organizaciones relacionadas con InterNet

Designador de País

éstos son códigos de dos letras que representan a un país en particular.

Por razones históricas, la mayoría de los dominios pertenecientes a uno de los de nivel superior que no basados en un nombre de país, son de organizaciones dentro de los Estados Unidos, aunque los Estados Unidos tienen también su propio código de país `.us`. Esto ha dejado de ser cierto para los dominios `.com` y `.org`, que son usados de forma común por compañías de fuera de los Estados Unidos de América.

Cada uno de estos dominios de nivel superior tiene subdominios. Los dominios de nivel superior basados en el nombre de un país suelen estar divididos en subdominios basados en `com`, `edu`, `gov`, `mil` y `org`. Por ejemplo, encontrará cosas como `com.au` y `gov.au`, las organizaciones comerciales y gubernamentales en Australia.

El siguiente nivel de división suele representar el nombre de la organización. Los siguientes subdominios varían, a menudo basando el siguiente nivel en la estructura departamental de la organización a la que pertenecen, pero pueden estarlo en cualquier criterio considerado razonable y con significado claro para los administradores de la red de la organización.

La parte más a la izquierda de un nombre siempre es el nombre único asignado a la máquina, y es llamada *hostname*, la porción del nombre a la derecha del nombre de la máquina es el *domainname* (nombre de dominio), y el nombre completo es llamado *Fully Qualified Domain Name* (Nombre de Dominio Completamente Cualificado).

Si usamos el ordenador de Terry como ejemplo, el nombre completamente cualificado es `perf.no.itg.telstra.com.au`. Esto significa que el nombre de la máquina es `perf` y el nombre de dominio el `no.itg.telstra.com.au`. El nombre de dominio está basado en un dominio de nivel superior basado en su país, Australia, y como su dirección de correo pertenece a una organización comercial tenemos `.com` como siguiente nivel de dominio. El nombre de la compañía es (era) `telstra`, y la estructura interna de su nombre está basada en una estructura organizacional. En su caso, la máquina pertenece al Grupo de Tecnología de la Información (Information Technology Group), sección de Operaciones de Red (Network Operations).

Los nombres son, por norma, bastante más cortos; por ejemplo, mi PSI se llama `systemy.it` y mi organización sin ánimo de lucro se llama `linux.it`, sin subdominio `com` ni `org`, por lo que mi propia máquina se llama `morgana.systemy.it` y `rubini@linux.it` es una dirección de correo electrónico válida. Sepa que el dueño de un dominio tiene derecho tanto para registrar una máquina como para registrar subdominios; por ejemplo, el GUL (Grupo de Usuarios de Linux) al que pertenezco usa el dominio `pluto.linux.it`, porque los dueños de `linux.it` convinieron en abrir un subdominio para el GUL.

5.5.2 Qué información necesitará

Necesitará saber a qué dominio pertenecen sus máquinas. El software de resolución de nombres proporciona el servicio de traducción haciendo consultas a un *Servidor de Nombres de Dominio* (*Domain Name Server*), por lo que deberá saber la dirección IP del servidor de nombres (*nameserver*) local que vaya a usar.

Hay tres ficheros que es necesario editar. Los comentaré uno a uno.

5.5.3 `/etc/resolv.conf`

`/etc/resolv.conf` es el fichero de configuración principal del código de resolución de nombres. Su formato es bastante simple. Es un fichero de texto con una palabra clave por línea. Hay tres palabras clave de uso frecuente, que son:

domain

esta palabra clave especifica el nombre de dominio local.

search

ésta especifica una lista de dominios alternativos para completar el nombre de una máquina.

nameserver

ésta, que puede utilizarse varias veces, especifica una dirección IP de un servidor de nombres de dominio para consultarlo cuando se resuelvan nombres.

Su `/etc/resolv.conf` podría parecerse a éste:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

Este ejemplo especifica que el nombre de dominio por defecto para añadir a los nombres no cualificados (esto es, nombres de máquinas suministrados sin dominio) es `maths.wu.edu.au`, y que si no se encuentra la máquina en este dominio intentemos también el dominio `wu.edu.au` directamente. Se proporcionan dos entradas de servidor de nombres, cada uno de los cuales puede ser llamado por el código de resolución de nombres para traducir el nombre.

5.5.4 /etc/host.conf

El fichero `/etc/host.conf` es el lugar donde se configuran algunos elementos que gobiernan el comportamiento del código de resolución de nombres. El formato de este fichero está descrito en detalle en la página de manual `resolv+`. El ejemplo siguiente funcionará en casi todas las circunstancias:

```
order hosts,bind
multi on
```

Esta configuración le dice al resolutor de nombres que examine el fichero `/etc/hosts` antes de intentar consultar a un servidor de nombres, y que devuelva todas las direcciones válidas de una máquina que encuentre en el fichero `/etc/hosts`, en lugar de sólo el primero.

5.5.5 /etc/hosts

El fichero `/etc/hosts` es donde se pone el nombre y dirección IP de las máquinas locales. Si usted inserta un nombre en este fichero, entonces su ordenador no consultará a un servidor de dominio para obtener la dirección IP. La desventaja de este método es que usted mismo tendrá que poner el fichero al día si la dirección de alguna máquina cambia. En un sistema bien administrado, las únicas entradas que suelen aparecer son la interfaz de loopback (prueba en bucle) y el nombre de la máquina local.

```
# /etc/hosts
127.0.0.1      localhost loopback
192.168.0.1    this.host.name
```

Se puede especificar más de un nombre de máquina por línea, como se demuestra en la primera entrada, que es la forma estándar para la interfaz de prueba (loopback).

5.5.6 Ejecutar un servidor de nombres

Si quiere tener un servidor de nombres local, le resultará sencillo. Por favor, lea el DNS Como, *http://www.insflug.org/documentos/DNS-Como/* y la documentación incluida en su copia de `BIND` (*Berkeley Internet Name Domain*).

5.6 Configuración de la interfaz `loopback`

La interfaz *loopback*' es un tipo especial de interfaz que le permite hacer conexiones consigo mismo. Hay varias razones por las que podría querer esto. Por ejemplo, puede que desee probar algún tipo de programa sin interferir con alguien más en su red. Por convención, la dirección de red IP `127.0.0.1` ha sido asignada específicamente para el dispositivo de pruebas. Por lo que da igual lo que haga su máquina, que si abre una conexión de telnet a `127.0.0.1`, siempre llegará a la interfaz interna.

Configurar la interfaz `loopback` es simple y debería asegurarse de hacerlo.

```
root# ifconfig lo 127.0.0.1
root# route add -host 127.0.0.1 lo
```

Hablaremos más de la orden `route` en la siguiente sección.

5.7 Encaminamiento (*Routing*).

El encaminamiento es un tema amplio. Sería fácil llenar varios volúmenes hablando de él. La mayoría de ustedes tendrán requisitos de encaminamiento relativamente simples, pero algunos no. Cubriré solamente algunos de los fundamentos básicos. Si está interesado en información más detallada, entonces sugiero que se remita a las referencias propuestas al principio del documento.

Comencemos con una definición. ¿Qué es el encaminamiento IP? Esta es la que yo suelo aplicar:

> El Encaminamiento IP es el proceso por el que una máquina con múltiples conexiones de red decide por dónde dirigir un datagrama IP que haya recibido.

Sería útil ilustrar esto con un ejemplo. Imagine una oficina con el encaminamiento típico, que podría constar de un enlace PPP con Internet, varios segmentos Ethernet alimentando las estaciones de trabajo, y un enlace PPP hacia otra oficina. Cuando el encaminador recibe un datagrama en cualquiera de sus conexiones de red, el mecanismo que usa para determinar qué interfaz debería enviar el datagrama, es el encaminamiento. Las estaciones sencillas también necesitan encaminar. Todas las estaciones en Internet tienen dos dispositivos de red: uno es la interfaz de pruebas descrita anteriormente, y la otra es la que usa para comunicarse con el resto de la red, quizá una Ethernet, quizá una interfaz serie PPP o SLIP.

Bien... Por tanto, ¿cómo funciona el encaminado? Cada máquina tiene una lista de reglas, llamada tabla de encaminamiento. Esta tabla contiene columnas que suelen contener al menos tres campos: el primero es una dirección de destino, el segundo es el nombre de la interfaz a la que se va a encaminar el datagrama, y el tercero es (opcionalmente) la dirección IP de otra máquina que cogerá el datagrama en su siguiente paso a través de la red. En Linux puede ver esta tabla usando la siguiente orden:

```
root# cat /proc/net/route
```

o con cualquiera de estas otras:

```
root# /sbin/route -n
root# /bin/netstat -r
```

El proceso de encaminado es relativamente simple: se recibe un datagrama que llega, se examina la dirección de destino (para quién es) y se compara con cada entrada en la lista. Se selecciona la entrada que más se parezca y se reenvía el datagrama a la interfaz especificada. Si el campo de pasarela (gateway) está descrito, el datagrama se reenvía a esa máquina mediante la interfaz especificada, y si no, se asume que la dirección de destino está en la red a la que se accede mediante la interfaz correspondiente.

Para manipular esta tabla se usa una orden especial. Esta instrucción toma sus argumentos en línea de órdenes y los convierte en llamadas al sistema del núcleo, que le piden que añada, borre o modifique entradas en la tabla de encaminamiento. Dicha orden se llama `route`.

Un ejemplo sencillo. Imagine que tiene una red Ethernet. Le han dicho que pertenece a una red clase C con dirección `192.168.1.0`. Le han dado la dirección IP `192.168.1.10` para su uso y también que `192.168.1.1` es un encaminador conectado a Internet.

El primer paso es configurar la interfaz como se describió anteriormente. Puede usar una orden como:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Ahora necesita añadir una entrada en la tabla de rutas para decirle al núcleo que los datagramas destinados a todas las máquinas con direcciones que se ajusten a `192.168.1.*`, deberán ser enviados al dispositivo Ethernet. Debería usar una orden similar a:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Fíjese en el uso del parámetro `-net` para especificar al programa `route` que esta entrada es una regla para una red completa. Otra opción es `-host`, que indica una vía específica a una dirección IP en particular.

Esta ruta le permitirá establecer conexiones IP con todas las estaciones de su segmento Ethernet. Pero ¿qué pasa con todas las máquinas con dirección IP que no están en su segmento Ethernet?

Sería bastante engorroso, por no decir imposible, tener que añadir caminos para cada red de destino posible, por lo que existe un truco que se usa para simplificar la tarea. A este truco se le llama camino *por defecto*. El camino *por defecto* se ajusta a todo destino posible, pero con prioridad mínima, por lo que si existe cualquier otra entrada que se ajuste a la dirección requerida, será la que se use en lugar del camino *por defecto*. La idea del camino *por defecto* es simplemente permitirle decir: «y todo lo demás debería ir por aquí». En el ejemplo que me he inventado podría usar una orden como:

```
root# route add default gw 192.168.1.1 eth0
```

El parámetro `gw` le dice a la orden `route` que lo que le sigue es la dirección IP, o nombre, de una pasarela u otra máquina encaminadora a la que se deberían enviar todos los datagramas que se ajusten a esta entrada para futuro encaminamiento.

Por lo tanto, la configuración completa debería parecerse a:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add default gw 192.168.1.1 eth0
```

Si echa una mirada a los ficheros `rc` de red de su máquina, encontrará que al menos uno de ellos se parece mucho a esto. Es una configuración muy común.

Veamos ahora una configuración de encaminamiento un poco más complicada. Imaginemos que estamos configurando el encaminador de antes, el que soportaba el enlace PPP a Internet y los segmentos LAN alimentando las estaciones de trabajo en la oficina. Imaginemos que el encaminador tiene tres segmentos Ethernet y un enlace PPP. Nuestra configuración de encaminamiento debería parecerse a esto:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
root# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
root# route add default ppp0
```

Cada una de las estaciones debería usar la forma simple que presentamos anteriormente. Sólo el encaminador necesita especificar cada uno de los caminos de red de forma separada, porque para las estaciones el mecanismo de encaminamiento `por defecto` enviará todos los paquetes al encaminador, dejándole el problema de repartirlos. Puede estar preguntándose por qué el encaminador por defecto presentado no especifica un `gw`. La razón es simple: los protocolos de enlace serie como PPP y SLIP sólo tienen dos terminales en su red, uno en cada extremo. Especificar el host al otro extremo como pasarela no tiene sentido y es redundante, ya que no hay otra elección, por lo que no se necesita especificar una pasarela para este tipo de conexión de red. Otros tipos como Ethernet, arcnet o token ring necesitan que se especifique la pasarela, ya que se puede acceder a un gran número de terminales al mismo tiempo.

5.7.1 Entonces ¿qué hace el programa `routed`?

La configuración de encaminamiento descrita anteriormente se ajusta a necesidades de red simples, donde los posibles caminos hacia los destinos son sencillos. Cuando se tienen necesidades de red más complejas, las cosas se vuelven un poco más complicadas. Afortunadamente la mayoría de ustedes no tendrá este problema.

El gran problema con el «encaminamiento manual» o «encaminamiento estático» que aquí se ha descrito, es que si una máquina o enlace falla en la red, entonces la única manera en que podrá dirigir sus datagramas por otra dirección, si es que existe, es interviniendo manualmente y ejecutando las órdenes apropiadas. Naturalmente esto es poco elegante, lento, nada práctico y peligroso. Se han desarrollado varias técnicas para ajustar automáticamente las tablas de encaminamiento en el caso de fallos en la red donde hubiera caminos alternativos. Todas estas técnicas se agrupan de manera no muy oficial bajo la definición «protocolos de encaminamiento dinámico».

Puede que haya oído de alguno de los protocolos más comunes. Es probable que RIP (*Routing Information Protocol*) y OSPF (*Open Shortest Path First Protocol*) sean los más comunes. El Routing Information Protocol es muy común en redes pequeñas, como las redes corporativas pequeñas y medianas o en las redes de edificios. El OSPF es más moderno y más capaz de gestionar grandes configuraciones de red, y está mejor preparado para entornos donde haya un gran número de caminos posibles a través de la red. Las implementaciones habituales de estos protocolos son: `routed` (RIP) y `gated` (RIP, OSPF y otros). El programa `routed` suele venir incluido en las distribuciones de Linux, y si no, estará incluido en el paquete *NetKit*, detallado más adelante.

Un ejemplo de dónde y cómo debería usar un protocolo de encaminamiento dinámico vendría a ser algo como lo siguiente:

```
         192.168.1.0 /                         192.168.2.0 /
            255.255.255.0                         255.255.255.0
             -                                     -
             |                                     |
             |   /-----\                 /-----\   |
             |   |     |ppp0   //    ppp0|     |   |
        eth0 |---|  A  |------//---------|  B  |---| eth0
             |   |     |     //          |     |   |
             |   \-----/                 \-----/   |
             |      \ ppp1             ppp1 /      |
             -       \                     /       -
                      \                   /
```

```
\                 /
 \               /
  \             /
   \           /
    \         /
     \       /
      \     /
   ppp0\   /ppp1
      /-----\
      |     |
      |  C  |
      |     |
      \-----/
         |eth0
         |
    |---------|
    192.168.3.0 /
       255.255.255.0
```

Tenemos tres encaminadores `A`, `B` y `C`. Cada uno da servicio a un segmento Ethernet con una red IP de clase C (máscara de red `255.255.255.0`). Cada encaminador tiene también un enlace PPP a cada uno de los otros encaminadores. La red forma un triángulo.

Debería estar claro que la tabla de encaminamiento del encaminador `A` podría ser algo como:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
root# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Esto funcionaría bien hasta que el enlace entre los encaminadores `A` y `B` falle. Si falla ese enlace, entonces con la entrada de encaminamiento mostrada anteriormente, las máquinas del segmento Ethernet de `A` no podrán alcanzar a las del segmento Ethernet en `B` porque sus datagramas serán dirigidos al enlace `ppp0` de `A` que está mal. Podrían seguir comunicándose todavía con las máquinas del segmento Ethernet de `C`, y las del segmento Ethernet de `C` con las del segmento Ethernet de `B`, porque el enlace entre `B` y `C` aún está intacto.

Pero espere un momento. Si `A` puede hablar con `C` y `C` puede aún hablar con `B`, ¿por qué no puede `A` encaminar sus datagramas para `B` a través de `C` y dejar que `C` se los mande a `B`? Este es exactamente el tipo de problema para el que fueron diseñados los protocolos de encaminamiento dinámico como RIP. Si cada uno de los encaminadores `A`, `B` y `C` está ejecutando el demonio de encaminamiento, entonces sus tablas deberían ser ajustadas automáticamente para reflejar el nuevo estado de la red si alguno de los enlaces falla. Configurar tal red es sencillo. En cada encaminador sólo necesita hacer dos cosas. En este caso, para el Encaminador `A`:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# /usr/sbin/routed
```

El demonio de encaminamiento `routed` encuentra automáticamente todos los puertos de red activos cuando comienza y busca mensajes en cada uno de los dispositivos de red para poder determinar y poner al día la tabla de encaminamiento en el host.

Ésta ha sido una explicación muy concisa del encaminamiento dinámico y de dónde podría usarlo. Si quiere más información, tendrá que acudir a las referencias listadas al principio del documento.

Los puntos importantes relacionados con el encaminamiento dinámico son:

1. Sólo necesita ejecutar un protocolo de encaminamiento dinámico cuando su máquina Linux tenga la posibilidad de elegir entre múltiples caminos para llegar a un destino.
2. El demonio de encaminamiento dinámico modificará automáticamente la tabla de encaminamiento para ajustarla a los cambios en la red.
3. RIP es adecuado para redes de tamaño pequeño y medio.

5.8 Configuración de los servidores de red y los servicios.

Los servidores de red y los servicios son aquellos programas que permiten a un usuario remoto hacer uso de su máquina Linux. Los programas servidores escuchan en los puertos de red. Los puertos de red son el medio de llegar a un servicio en particular en una máquina en particular, y es así como un servidor conoce la diferencia entre una conexión telnet y otra de FTP que le lleguen. El usuario remoto establece una conexión de red con la máquina, y el programa servidor, el demonio de red que esté escuchando en ese puerto, aceptará la conexión y se ejecutará. Hay dos modos de operación para los demonios de red. Ambos se usan por igual en la práctica. Las dos maneras son:

autónomo (standalone)

el programa demonio de red escucha en el puerto de red asignado y, cuando llega una conexión, se ocupa él mismo de dar el servicio de red.

esclavo del servidor `inetd`

el servidor inetd es un demonio de red especial que se especializa en controlar las conexiones entrantes. Tiene un fichero de configuración que le dice qué programa debe ser ejecutado cuando se reciba una conexión. Cualquier puerto de servicio puede ser configurado tanto para el protocolo tcp como para udp. Los puertos son descritos en otro fichero del que hablaremos dentro de poco.

Hay dos ficheros importantes que necesitamos configurar. Son el fichero `/etc/services` que asigna nombres a los números de puerto y el fichero `/etc/inetd.conf` que es el fichero de configuración del demonio de red `inetd`.

5.8.1 `/etc/services`

El fichero `/etc/services` es una base de datos sencilla, que asocia un nombre que nosotros podamos entender, con un puerto de servicio de la máquina. Su formato es bastante simple. Es un fichero de texto en el que cada línea representa una entrada a la base de datos. Cada entrada comprende tres campos separados por cualquier número de espacios en blanco (espacio o tabulador). Los campos son:

```
nombre      puerto/protocolo        sobrenombres     # comentario
```

nombre

una sola palabra que representa el servicio descrito.

puerto/protocolo

este campo está dividido en dos subcampos.

puerto

un número que especifica el número de puerto del servicio que estará disponible. La mayoría de los servicios comunes tienen asignados números de servicio, y están descritos en el *RFC-1340*.

protocolo

este subcampo debe tener como valor `tcp` o `udp`.

Es importante que tenga en cuenta que el servicio `18/tcp` es muy diferente del `18/udp` y que no hay razón técnica por la cual deban existir ambos. Normalmente prevalece el sentido común, y sólo verá una entrada `tcp` y otra `udp` para el mismo servicio si es que está disponible para ambos.

sobrenombres

(o alias) otros nombres que pueden usarse para referirse a esta entrada de servicio.

Cualquier texto que aparezca en una línea después de un caracter `#` es ignorado y se trata como un comentario.

Un ejemplo de fichero `/etc/services`. Todas las distribuciones modernas de Linux tienen un buen fichero `/etc/services`. Para el caso de que esté montando una máquina desde cero, aquí tiene una copia del fichero `/etc/services` proporcionado por una antigua la distribución Debian *http://www.debian.org*:

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1340, «Assigned Numbers» (July 1992).  Not all ports
# are included, only the more common ones.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - unassigned
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
```

```
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
bootps          67/tcp                          # BOOTP server
bootps          67/udp
bootpc          68/tcp                          # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
www             80/udp                          # HyperText Transfer Protocol
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5  # Kerberos v5
kerberos        88/udp          kerberos5 krb5  # Kerberos v5
supdup          95/tcp
# 100 - reserved
hostnames       101/tcp         hostname        # usually from sri-nic
iso-tsap        102/tcp         tsap            # part of ISODE.
csnet-ns        105/tcp         cso-ns          # also used by CSO name server
csnet-ns        105/udp         cso-ns
rtelnet         107/tcp                         # Remote Telnet
rtelnet         107/udp
pop-2           109/tcp         postoffice      # POP version 2
pop-2           109/udp
pop-3           110/tcp                         # POP version 3
pop-3           110/udp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
imap2           143/tcp                         # Interim Mail Access Proto v2
imap2           143/udp
snmp            161/udp                         # Simple Net Mgmt Proto
snmp-trap       162/udp         snmptrap        # Traps for SNMP
cmip-man        163/tcp                         # ISO mgmt over IP (CMOT)
cmip-man        163/udp
cmip-agent      164/tcp
cmip-agent      164/udp
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
```

```
nextstep        178/tcp         NeXTStep NextStep       # NeXTStep window
nextstep        178/udp         NeXTStep NextStep       # server
bgp             179/tcp                         # Border Gateway Proto.
bgp             179/udp
prospero        191/tcp                         # Cliff Neuman's Prospero
prospero        191/udp
irc             194/tcp                         # Internet Relay Chat
irc             194/udp
smux            199/tcp                         # SNMP Unix Multiplexer
smux            199/udp
at-rtmp         201/tcp                         # AppleTalk routing
at-rtmp         201/udp
at-nbp          202/tcp                         # AppleTalk name binding
at-nbp          202/udp
at-echo         204/tcp                         # AppleTalk echo
at-echo         204/udp
at-zis          206/tcp                         # AppleTalk zone information
at-zis          206/udp
z3950           210/tcp         wais            # NISO Z39.50 database
z3950           210/udp         wais
ipx             213/tcp                         # IPX
ipx             213/udp
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
ulistserv       372/tcp                         # UNIX Listserv
ulistserv       372/udp
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
route           520/udp         router routed   # RIP
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
klogin          543/tcp                         # Kerberized `rlogin' (v5)
kshell          544/tcp         krcmd           # Kerberized `rsh' (v5)
kerberos-adm    749/tcp                         # Kerberos `kadmin' (v5)
#
webster         765/tcp                         # Network dictionary
webster         765/udp
#
# From «Assigned Numbers»:
```

```
#
#> The Registered Ports are not controlled by the IANA and on most systems
#> can be used by ordinary user processes or programs executed by ordinary
#> users.
#
#> Ports are used in the TCP [45,106] to name the ends of logical
#> connections which carry long term conversations.  For the purpose of
#> providing services to unknown callers, a service contact port is
#> defined.  This list specifies the port used by the server process as its
#> contact port.  While the IANA can not control use of these ports it
#> does register or list use of these ports as a convienence to the
#> community.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np     1525/tcp                # Prospero non-privileged
prospero-np     1525/udp
rfe             5002/tcp                # Radio Free Ethernet
rfe             5002/udp                # Actually use UDP only
bbs             7000/tcp                # BBS service
#
#
# Kerberos (Project Athena/MIT) services
# Note that these are for Kerberos v4 and are unofficial.  Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
kerberos4       750/udp         kdc     # Kerberos (server) udp
kerberos4       750/tcp         kdc     # Kerberos (server) tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
krb_prop        754/tcp                 # Kerberos slave propagation
krbupdate       760/tcp         kreg    # Kerberos registration
kpasswd         761/tcp         kpwd    # Kerberos "passwd"
kpop            1109/tcp                # Pop with Kerberos
knetd           2053/tcp                # Kerberos de-multiplexor
zephyr-srv      2102/udp                # Zephyr server
zephyr-clt      2103/udp                # Zephyr serv-hm connection
zephyr-hm       2104/udp                # Zephyr hostmanager
eklogin         2105/tcp                # Kerberos encrypted rlogin
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv      871/tcp                 # SUP server
supfiledbg      1127/tcp                # SUP debugging
#
# Datagram Delivery Protocol services
#
rtmp            1/ddp                   # Routing Table Maintenance Protocol
nbp             2/ddp                   # Name Binding Protocol
echo            4/ddp                   # AppleTalk Echo Protocol
zip             6/ddp                   # Zone Information Protocol
#
# Debian GNU/Linux services
rmtcfg          1236/tcp                # Gracilis Packeten remote config server
```

```
xtel            1313/tcp                # french minitel
cfinger         2003/tcp                # GNU Finger
postgres        4321/tcp                # POSTGRES
mandelspawn     9359/udp        mandelbrot      # network mandelbrot

# Local services
```

En el día a día, este fichero se encuentra en proceso de continuo crecimiento según se van creando nuevos servicios. Si piensa que su copia es incompleta, le sugiero que haga una copia del `/etc/services` de una distribución reciente.

5.8.2 `/etc/inetd.conf`

`/etc/inetd.conf` es el fichero de configuración para el demonio servidor `inetd`. Su función es la de almacenar la información relativa a lo que `inetd` debe hacer cuando recibe una petición de conexión a un servicio en particular. Para cada servicio que desee que acepte conexiones deberá decirle a `inetd` qué demonio servidor de red ejecutar, y cómo ha de hacerlo.

Su formato también es relativamente sencillo. Es un fichero de texto en el que cada línea describe un servicio que desee proporcionar. Cualquier texto en una línea que siga a `#` es ignorado y se considera un comentario. Cada línea contiene siete campos separados por cualquier número de espacios en blanco (espacio o tabulador). El formato general es el siguiente:

```
servicio  tipo_socket  proto  flags  usuario  servidor  argumentos
```

servicio

es el servicio correspondiente a esta configuración, tomado del fichero `/etc/services`.

tipo_socket

describe el tipo de socket que esta entrada considerará relevante. Los valores permitidos son: `stream`, `dgram`, `raw`, `rdm` o `seqpacket`. Es un poco técnico por naturaleza, pero por regla general casi todos los servicios basados en `tcp` usan `stream`, y casi todos los basados en `udp` usan `dgram`. Sólo algunos demonios servidores muy particulares usarán otros valores.

proto

el protocolo considerado válido para este servicio. Debería corresponder con la entrada apropiada en el fichero `/etc/services` y suele ser `tcp` o `udp`. Los servidores basados en Sun RPC (*Remote Procedure Call*) usarán `rpc/tcp` o `rpc/udp`.

flags

sólo hay dos valores posibles. Este campo le dice a *inetd* si el programa servidor de red libera el socket después de comenzar la ejecución, y si por tanto `inetd` podrá ejecutar otro servidor para la siguiente petición de conexión, o si `inetd` deberá esperar y asumir que el demonio servidor que esté ejecutándose controlará las nuevas peticiones de conexión. Esto tiene su dificultad, pero por norma general todos los servidores `tcp` deberían tener esta entrada con el valor `nowait` y la mayoría de servidores `udp` deberían tener `wait`. De todas maneras hay algunas excepciones notables, por lo que debería leer la guía de ejemplo si no está seguro.

usuario

este campo indica qué cuenta de usuario de `/etc/passwd` será asignada como dueña del demonio de red cuando se ejecute. Esto es a menudo útil si quiere protegerse ante riesgos de seguridad. Puede asignar el usuario `nobody` a una entrada, por lo que si la seguridad del servidor de red es traspasada el posible daño queda minimizado. Habitualmente, sin embargo, este campo está asignado a `root`, porque muchos servidores requieren privilegios de administrador para funcionar correctamente.

servidor

este campo es el camino completo hasta el programa servidor a ejecutar para esta entrada.

argumentos

este campo comprende el resto de la línea de órdenes y es opcional. Es en donde se pone cualquier argumento de línea de órdenes que desee pasar al programa demonio servidor cuando es ejecutado.

Un ejemplo de `/etc/inetd.conf` Al igual que pasa con el `/etc/services`, todas las distribuciones modernas incluirán un buen fichero `/etc/inetd.conf` para trabajar con él. Aquí incluyo, como ejemplo, el fichero `/etc/inetd.conf` de la distribución Debian *http://www.debian.org*.

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Modified for Debian by Peter Tobias <tobias@et-inf.fho-emden.de>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Internal services
#
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
discard         stream  tcp     nowait  root    internal
discard         dgram   udp     wait    root    internal
daytime         stream  tcp     nowait  root    internal
daytime         dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# These are standard services.
#
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Mail, news and uucp services.
#
smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
```

```
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
# Pop et al
#
#pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# `cfinger' is for the GNU finger server available for Debian.  (NOTE: The
# current implementation of the `finger' daemon allows it to be run as `root'.)
#
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers."
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i -t 120
#
# Kerberos authenticated services (these probably need to be corrected)
#
#klogin         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k -x
#kshell         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services run ONLY on the Kerberos server (these probably need to be corrected)
#
#krbupdate      stream tcp      nowait  root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# RPC based services
#
#mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
#rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1        dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# End of inetd.conf.
ident           stream  tcp     nowait  nobody  /usr/sbin/identd        identd -i
```

5.9 Otros ficheros de configuración relacionados con la red

Hay varios ficheros misceláneos relacionados con la configuración de la red en Linux por los que podría estar interesado. Nunca debería tener que modificar estos ficheros, pero merece la pena describirlos para que sepa lo que contienen y para qué son.

5.9.1 /etc/protocols

El fichero `/etc/protocols` es una base de datos que correlaciona números de identificación de protocolos con sus nombres. Esto lo usan los programadores para especificar protocolos por su nombre en sus programas y también por programas como `tcpdump` para mostrar nombres en lugar de números en su salida. En general la sintaxis del fichero es:

```
nombredelprotocolo  número  sobrenombres
```

El fichero `/etc/protocols` proporcionado con la distribución Debian *http://www.debian.org* es como sigue:

```
# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).

ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially «IP»)
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf    73      RSPF            # Radio Shortest Path First.
vmtp    81      VMTP            # Versatile Message Transport
ospf    89      OSPFIGP         # Open Shortest Path First IGP
ipip    94      IPIP            # Yet Another IP encapsulation
encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.2 /etc/networks

El fichero `/etc/networks` tiene una función similar a la del fichero `/etc/hosts`. Proporciona una base de datos sencilla de nombres de red y direcciones de red. Su formato difiere en que sólo puede haber dos campos por línea y los campos están codificados así:

```
nombredelared direccióndered
```

Un ejemplo podría ser:

```
loopnet    127.0.0.0
localnet   192.168.0.0
amprnet    44.0.0.0
```

Cuando use órdenes como `route`, si un destino es una red y la red tiene una entrada en el fichero /etc/networks entonces route mostrará el nombre de la red en lugar de su dirección.

5.10 Seguridad en la red y control de acceso.

Déjeme empezar esta sección advirtiendo que la seguridad de su máquina y red ante ataques maliciosos es un arte complejo. No me considero un experto en este campo y aunque los mecanismos que voy a describir puedan ayudar, si quiere tomarse en serio la seguridad entonces le recomiendo que investigue un poco en el tema. Hay algunas buenas referencias en Internet relacionadas con la seguridad, incluido el *Security HOWTO* (Dispone de una traducción en *http://www.insflug.org/documentos/Seguridad-Como/*.

Una regla general importante es: **No ejecute servicios que no tenga intención de usar**'. Muchas distribuciones vienen configuradas con todo tipo de servicios que se inician automáticamente. Para asegurar un nivel mínimo de seguridad debería examinar el fichero `/etc/inetd.conf` y comentar (*poner un # al inicio de la línea*) de toda declaración de servicio que no vaya a usar. Buenos candidatos son: `shell`, `exec`, `uucp`, `ftp` y servicios de información como `finger`, `netstat` y `systat`.

Hay todo tipo de mecanismos de seguridad y control de acceso, describiré los más elementales.

5.10.1 /etc/ftpusers

El fichero `/etc/ftpusers` es un mecanismo sencillo que le permite denegar la entrada a ciertos usuarios mediante FTP. El fichero `/etc/ftpusers` lo lee el programa demonio de FTP (`ftpd`) cuando se recibe una conexión FTP. El fichero es una simple lista de aquellos usuarios que no tienen permitido el acceso. Puede parecerse a esto:

```
# /etc/ftpusers - users not allowed to login via ftp
root
uucp
bin
mail
```

5.10.2 /etc/securetty

El fichero `/etc/securetty` permite especificar qué dispositivos `tty` puede usar `root` para identificarse en el sistema. El fichero `/etc/securetty` es leído por el programa de acceso (normalmente `/etc/login`). Su formato es una lista de los nombres de dispositivos `tty` permitidos, en todos los demás `root` tiene prohibida la entrada:

```
# /etc/securetty - tty's on which root is allowed to login
tty1
tty2
tty3
tty4
```

5.10.3 El mecanismo de control de acceso *hosts* de `tcpd`

El programa `tcpd` que ha visto listado en el fichero `/etc/inetd.conf` proporciona mecanismos de control de registro y acceso a los servicios que haya de proteger.

Cuando es invocado por el programa `inetd` lee dos ficheros que contienen reglas de acceso y que permiten o deniegan acceso al servidor que está protegiendo.

Mirará en los ficheros de reglas hasta que encuentre la primera correspondencia. Si no se encuentran correspondencias asume que el acceso debería estar permitido para todo el mundo. La secuencia de archivos que revisa es: `/etc/hosts.allow`, `/etc/hosts.deny`. Describiré cada uno de estos en seguida. Para una descripción completa de este servicio debería referirse a las páginas del manual apropiadas (`hosts_access` (5) es un buen punto de partida).

`/etc/hosts.allow` El `/etc/hosts.allow` es un fichero de configuración del programa `/usr/sbin/tcpd`. El fichero `hosts.allow` contiene reglas que describen qué máquinas tienen permiso para acceder a un servicio en la suya.

El formato del fichero es bastante sencillo:

```
# /etc/hosts.allow
#
# <lista de servicios>: <lista de hosts> [: orden]
```

`lista de servicios`

: es una lista delimitada por comas de nombres de servidores a los que se aplica esta regla. Ejemplos de nombre de servicio son: `ftpd`, `telnetd` y `fingerd`.

`lista de hosts`

: es una lista de nombres de máquinas, delimitada por comas. Aquí también puede usar direcciones IP. De forma adicional, puede especificar nombres de máquinas o direcciones usando caracteres comodín para corresponder con grupos de máquinas. Por ejemplo: `gw.vk2ktj.ampr.org` para una máquina específica, `.uts.edu.au` para cualquier nombre de máquina que acabe en esa cadena, `44.` para cualquier dirección IP que comience con esos dígitos. Hay algunas palabras especiales para simplificar la configuración, algunas de las cuales son:

- `ALL`, que se corresponde con cualquier host.
- `LOCAL` se corresponde con cualquier nombre de host que no contenga un `.` o sea que esté en el mismo dominio que su máquina;
- `PARANOID` se corresponde con cualquier nombre que no se corresponda con esta dirección (name spoofing). Hay una última palabra que también es útil. La palabra
- `EXCEPT` permite proporcionar una lista con excepciones. Esto lo cubriremos en un capítulo posterior.

`orden`

: es un parámetro opcional. Este parámetro es el camino completo hasta una orden que debería ser ejecutada cada vez que se cumpla esta regla. Podría por ejemplo ejecutar una instrucción que intentase identificar quién está autenticado en el host que conecta, o generar un mensaje de correo u otro tipo de alerta a un administrador de sistema avisando de que alguien intenta conectar. Hay cierto número de expansiones que podríamos incluir, algunos ejemplos comunes son: `%h` se expande al nombre de la máquina que se conecta o a su dirección si no tiene un nombre, `%d` es el demonio que está siendo llamado.

Un ejemplo:

```
# /etc/hosts.allow
#
# Permitir correo a todo el mundo
in.smtpd: ALL
# Todo telnet y FTP sólo a hosts dentro de mi dominio y el host que tengo
# en caso
telnetd, ftpd: LOCAL, myhost.athome.org.au
# Permitir finger a cualquiera pero mantener un registro de quién es.
fingerd: ALL: (finger @%h | mail -s "finger desde %h" root)
```

`/etc/hosts.deny` El fichero `/etc/hosts.deny` es un fichero de configuración del programa `/usr/sbin/tcpd`. El fichero `hosts.deny` contiene reglas que describen qué máquinas tienen *prohibido* el acceso a un servicio en su máquina.

Un ejemplo simple podría parecerse a esto:

```
# /etc/hosts.deny
#
# Desautorizar a todos los host con nombre sospechoso
ALL: PARANOID
#
# Desautorizar a todos los host.
ALL: ALL
```

La entrada `PARANOID` es redundante porque la otra entrada abarca todo en cualquier caso. Ambas entradas serían razonables por defecto dependiendo de sus requisitos particulares.

La configuración más segura es tener `ALL: ALL` por defecto en `/etc/hosts.deny` para después dar permiso específicamente a aquellos servicios y hosts que se desee en `/etc/hosts.allow`.

5.10.4 `/etc/hosts.equiv`

El fichero `hosts.equiv` se usa para garantizar a ciertos hosts y usuarios derechos de acceso a cuentas en su máquina sin que tenga que proporcionar una clave. Esto es útil en un entorno seguro donde controle todas las máquinas, pero en otro caso es un peligro para la seguridad. Su máquina es sólo tan segura como la menos segura de aquellas en las que confíe. Para maximizar la seguridad, no use este mecanismo, y anime a sus usuarios para que tampoco usen ficheros `.rhost`.

5.10.5 Configure su demonio de ftp adecuadamente.

Muchos servidores estarán interesados en ejecutar un demonio de *FTP* anónimo para permitir a otras personas que subir y descargar ficheros sin necesidad de un *userid* específico. Si decide ofrecer este servicio asegúrese de que configura el demonio de *ftp* apropiadamente para acceso anónimo. La mayoría de las páginas de `ftpd(8)` describen cómo hacerlo. Debería asegurarse siempre de que sigue estas instrucciones. Una cosa importante es no usar una copia de su fichero `/etc/passwd` habitual en el directorio `/etc` de la cuenta anónima; asegúrese de que elimina todos los detalles sobre las cuentas excepto aquellos que deba tener, ya que en otro caso será vulnerable a las técnicas de adquisición de claves por fuerza bruta.

5.10.6 Cortafuegos para redes.

Un excelente medio de seguridad es no permitir que los datagramas lleguen siquiera a su máquina o servidores. Esto lo cubre en profundidad el *http://www.insflug.org/documentos/Cortafuegos-Como/* y, más concisamente, la última sección de este documento.

5.10.7 Otras sugerencias.

Hay otras sugerencias, que debería considerar, pero que realmente son cuestión de cada uno.

sendmail

a pesar de su popularidad, el demonio *sendmail* aparece con preocupante regularidad en los anuncios de alerta de seguridad. Usted decides, pero yo elijo no ejecutarlo.

NFS y otros servicios Sun RPC

tenga cuidado con estos. Hay todo tipo de posibles formas de explotar estos servicios. Es difícil de encontrar una opción a los servicios NFS, y si decide usarlos, asegúrese de que es cuidadoso con los permisos que da al configurarlo.

6 Información relacionada con IP y Ethernet

Esta sección cubre la información específica a Ethernet e IP. Estas subsecciones han sido agrupadas juntas debido a que pienso que son las más interesante dentro de la que previamente se llamó sección de «Tecnología Específica». Cualquiera que tenga una LAN se beneficiará de esta parte.

6.1 Ethernet

Los dispositivos Ethernet son `eth0`, `eth1`, `eth2`, etc. A la primera tarjeta detectada por el núcleo se le asigna `eth0` y al resto se les asigna secuencialmente en el orden en que sean detectadas.

Por defecto, el núcleo de Linux sólo intenta buscar una tarjeta de red, por lo que tendrá que pasarle algunos parámetros para forzar la detección de las demás.

Para aprender cómo hacer trabajar una tarjeta Ethernet bajo Linux debería acudir al Ethernet Howto, *Ethernet-HOWTO.html*.

Una vez que tenga el núcleo compilado para reconocer su tarjeta Ethernet, configurar la tarjeta es sencillo.

Normalmente debería hacer algo como:

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

La mayoría de los controladores Ethernet fueron desarrollados por Donald Becker, *becker@CESDIS.gsfc.nasa.gov*.

6.2 EQL - ecualizador de tráfico para líneas múltiples

El nombre del dispositivo EQL es `eql`. Con las fuentes estándar del núcleo sólo podrá tener un dispositivo EQL por máquina. EQL proporciona un medio para utilizar múltiples líneas punto a punto como un solo enlace PPP, slip o PLIP lógico para llevar tcp/ip. A menudo es más barato usar varias líneas de baja velocidad que tener una sola línea de alta velocidad.

Opciones de Compilación del Núcleo:

```
Network device support  --->
  [*] Network device support
  <*> EQL (serial line load balancing) support
```

Para poder usar este mecanismo la máquina en el otro extremo de la línea debe admitirlo también. Linux, Livingstone Portmasters y los nuevos servidores dial-in soportan servicios compatibles.

Para configurar EQL necesitará las herramientas eql que están disponibles en *ftp://metalab.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz*.

La configuración es bastante directa. Se comienza por la interfaz eql. La interfaz es exactamente como la de cualquier otro dispositivo de red. La dirección IP y mtu se configura usando `ifconfig`, con algo como:

```
root# ifconfig eql 192.168.10.1 mtu 1006
```

Lo siguiente que se necesita es iniciar manualmente cada una de las líneas que se vayan a usar. Esta puede ser cualquier combinación de dispositivos de red punto a punto. Cómo iniciar las conexiones depende del tipo de enlace que sea, por lo que ha de mirar las secciones apropiadas para más información.

Finalmente necesitará asociar el enlace serie con el dispositivo EQL, esto se llama «esclavizar» y se hace con la orden `eql_enslave` como se ve en el ejemplo:

```
root# eql_enslave eql sl0 28800
root# eql_enslave eql ppp0 14400
```

El parámetro *velocidad estimada* que se da a `eql_enslave` no hace nada directamente. Lo usa el controlador (*driver*) EQL para determinar qué parte de los datagramas va a recibir cada dispositivo, para así afinar el balance de las líneas jugando con este valor.

Para disociar una línea de un dispositivo EQL use `eql_emancipate` de esta manera:

```
#root eql_emancipate eql sl0
```

Puede añadir rutas de la misma manera que para cualquier otro enlace punto a punto, excepto en que las rutas deben referirse al dispositivo `eql` en lugar de a los verdaderos dispositivos en serie. Normalmente usará:

```
root# route add default eql
```

El controlador EQL fue desarrollado por Simon Janes, *simon@ncm.com*.

6.3 IP Accounting (en Linux 2.0)

Los servicios de auditoría IP del núcleo de Linux le permiten recolectar y analizar datos de uso de la red. Los datos recogidos comprenden el número de paquetes y de bytes acumulados desde que se puso a cero la última vez. Tiene a su disposición varias reglas para categorizar estas cifras para que se ajusten a los propósitos que les vaya a dar. Esta opción ha sido eliminada del núcleo a partir de la versión `2.1.102`, porque el viejo sistema de cortafuegos basado en `ipfwadm` fue reemplazado por `ipfwchains`.

Opciones de Compilación del Núcleo:

```
Networking options  --->
  [*] IP: accounting
```

Después de que haya compilado e instalado el núcleo necesitará hacer uso de la orden `ipfwadm` para configurar la auditoría de IP. Hay muchas maneras diferentes de obtener la información de auditoría para elegir. He creado un ejemplo sencillo de lo que podría ser útil, pero debería leer la página de manual de `ipfwadm` para obtener más información.

Escena: Tenemos una red Ethernet que accede a la Internet mediante un enlace PPP. En la Ethernet tiene una máquina que ofrece cierta variedad de servicios y está interesado en saber cuánto tráfico se genera por cada conexión ftp y http, así como el tráfico total tcp y udp.

Podrías usar un conjunto de órdenes que se pareciese a lo siguiente, que se presenta como guión de ejecución de órdenes (*shell script*):

```
#!/bin/sh
#
# Borrar las reglas de contabilidad
ipfwadm -A -f
#
# Establecer macros
localnet=44.136.8.96/29
any=0/0
# Agregar reglas para los segmentos de la red local
ipfwadm -A in  -a -P tcp -D $localnet ftp-data
ipfwadm -A out -a -P tcp -S $localnet ftp-data
ipfwadm -A in  -a -P tcp -D $localnet www
ipfwadm -A out -a -P tcp -S $localnet www
ipfwadm -A in  -a -P tcp -D $localnet
ipfwadm -A out -a -P tcp -S $localnet
ipfwadm -A in  -a -P udp -D $localnet
ipfwadm -A out -a -P udp -S $localnet
#
# Reglas por defecto
ipfwadm -A in  -a -P tcp -D $any ftp-data
ipfwadm -A out -a -P tcp -S $any ftp-data
ipfwadm -A in  -a -P tcp -D $any www
ipfwadm -A out -a -P tcp -S $any www
ipfwadm -A in  -a -P tcp -D $any
ipfwadm -A out -a -P tcp -S $any
ipfwadm -A in  -a -P udp -D $any
ipfwadm -A out -a -P udp -S $any
#
# Mostrar las reglas
ipfwadm -A -l -n
#
```

Los nombres `ftp-data` y `www` se refieren a líneas en `/etc/services`. La última orden da una lista de cada una de las reglas de auditoría y muestra los totales obtenidos.

Una cosa importante a tener en cuenta cuando analizamos IP accounting es que **los totales de todas las reglas que se han cumplido se han incrementado**, por tanto para obtener las diferencias tendrá que hacer los cálculos apropiados. Por ejemplo, si quiero saber cuantos datos no fueron de ftp, ni www restaré los totales individuales de la regla que se corresponde con todos los puertos.

```
root# ipfwadm -A -l -n
IP accounting rules
 pkts bytes dir prot source               destination          ports
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 20
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            20 -> *
   10  1166 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 80
   10   572 out tcp  44.136.8.96/29       0.0.0.0/0            80 -> *
  252 10943 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> *
  231 18831 out tcp  44.136.8.96/29       0.0.0.0/0             * -> *
    0     0 in  udp  0.0.0.0/0            44.136.8.96/29       * -> *
    0     0 out udp  44.136.8.96/29       0.0.0.0/0            * -> *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 20
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0            20 -> *
   10  1166 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> 80
   10   572 out tcp  0.0.0.0/0            0.0.0.0/0            80 -> *
  253 10983 in  tcp  0.0.0.0/0            0.0.0.0/0            * -> *
  231 18831 out tcp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 in  udp  0.0.0.0/0            0.0.0.0/0            * -> *
    0     0 out udp  0.0.0.0/0            0.0.0.0/0            * -> *
```

6.4 IP Accounting (en Linux 2.2)

El nuevo código para llevar la contabilidad se accede mediante «IP Firewall Chains». Consulte la página principal de IP chains *http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html* si desea más información. Entre otras cosas, necesitará usar `ipchains` en lugar de `ipfwadm` para configurar sus filtros. (De `Documentation/Changes` en las últimas fuentes del núcleo)

6.5 IP Aliasing

Hay algunas aplicaciones para las que es útil ser capaz de asignar varias direcciones IP a un sólo dispositivo de red. Los Proveedores de Servicios de Internet usan esto a menudo para ofertar a sus usuarios WWW y FTP «a medida». Puede acudir al *IP-Alias mini-HOWTO* si quiere obtener más información de la que aquí hay.

Opciones de Compilación del Núcleo:

```
Networking options  --->
   ....
   [*] Network aliasing
   ....
   <*> IP: aliasing support
```

Después de compilar e instalar su núcleo con la implementación de IP_Alias, la configuración es muy sencilla. Los alias se añaden a dispositivos de red virtuales asociados al verdadero dispositivo de red. Se aplica una convención sencilla para dar nombres a estos dispositivos, que es <`nombredisp`>`:`<`núm disp virtual`>, por ejemplo `eth0:0`, `ppp0:10`, etc. Tenga en cuenta que dispositivo nombre:número sólo se puede configurar *después* de haber activado la interfaz principal.

Por ejemplo, asumiremos que tiene una red Ethernet que soporta dos redes IP diferentes simultáneamente y quiere que tu máquina tenga acceso directo a ambas. Podríamos hacer algo como:

```
root# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

```
root# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
root# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Para borrar un alias sencillamente añadimos un '-' al final de su nombre y nos referimos a él de forma tan sencilla como:

```
root# ifconfig eth0:0- 0
```

También serán borradas automáticamente todas las rutas asociadas con ese alias.

6.6 IP Firewall (para Linux 2.0)

Los temas referentes a Cortafuegos IP y al trabajo con Cortafuegos está cubierto en mayor profundidad en el *http://www.insflug.org/documentos/Cortafuegos-Como/*. Tener un Cortafuegos IP le permite asegurar su máquina frente a accesos por la red no autorizados filtrando o permitiendo que entren datagramas de o hacia las direcciones IP que sean designadas. Hay tres clases diferentes de reglas, filtrado de entradas (*incoming filtering*), filtrado de salidas (*outgoing filtering*) y filtrado de reenvíos (*forwarding filtering*). Las reglas de entradas son aplicadas a los datagramas recibidos en un dispositivo de red. Las reglas de salidas se aplican a los datagramas que va a transmitir un dispositivo de red. Las reglas de reenvíos se aplican a datagramas que se reciben pero que no son para esta máquina, como por ejemplo datagramas que entran por una interfaz para ser encaminados por otra.

Opciones de Compilación del Núcleo:

```
Networking options  --->
  [*] Network firewalls
  ....
  [*] IP: forwarding/gatewaying
  ....
  [*] IP: firewalling
  [ ] IP: firewall packet logging
```

La configuración de las reglas del cortafuegos IP se realiza usando la orden `ipfwadm`. Como mencioné antes, yo no soy un experto en seguridad, por lo aunque le voy a presentar un ejemplo que se puede usar, usted debería investigar y desarrollar sus propias reglas si la seguridad es algo que le importe.

El uso más común de los cortafuegos IP es probablemente cuando está usando su máquina Linux como encaminador y pasarela (gateway) cortafuegos para proteger tu red local de accesos sin autorización desde fuera de la red.

La configuración siguiente está basada en una contribución de Arnt Gulbrandsen, *agulbra@troll.no*.

El ejemplo describe la configuración de las reglas de cortafuegos en la máquina cortafuegos/encaminadora ilustrada en este diagrama.

```
-                                  -
 \                                 | 172.16.37.0
  \                                |   /255.255.255.0
   \                 ---------     |
    |  172.16.174.30 | Linux |     |
NET =================|  f/w  |------|    ..37.19
    |    PPP         | router|     |  --------
   /                 ---------     |--| Mail |
```

```
  /                                |  | /DNS |
 /                                 |  --------
-                                  -
```

Las órdenes que siguen, deberían normalmente estar situadas en un fichero `rc` de manera que sean ejecutadas automáticamente cada vez que el sistema reinicie. Para mayor seguridad deberían ser llamadas después de que ser configuradas las interfaces de red, pero antes de que las interfaces hayan sido puestas «en marcha» para prevenir cualquier intento de acceder mientras la máquina cortafuegos está reiniciando.

```
#!/bin/sh

# Limpiar la tabla de reglas de 'Reenvíos'
# Cambiar la política por defecto a 'accept'
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# .. y lo mismo para 'Entradas'
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# Antes que nada, sellamos la interfaz PPP
# Me encantaría usar '-a deny' en lugar de '-a reject -y' pero entonces
# sería imposible originar conexiones desde ese interfaz.
# El -o causa que todos los datagramas rechazados sean registrados. Esto
# toma espacio de disco a cambio de tener el conocimiento de un ataque
# por error de configuración.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Eliminar ciertos tipos de paquetes que obviamente han sido generados
# de forma 'artificial': No puede venir nada de direcciones
# multicast/anycast/broadcast
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# y nunca deberíamos ver llegar por un cable nada de la red
# 'loopback'
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# aceptamos entradas de las conexiones SMTP y DNS, pero sólo
# hacia el Mail/Name Server
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS usa tanto UPD como TCP, por lo tanto los permitimos ambos
# para quien pregunte por nuestro servidor de nombres
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# pero no "respuestas" que lleguen a puertos peligrosos como el de NFS
# y extensiones de NFS de Larry McVoy. Si ejecutamos squid, añadir
# su puerto aquí también.
```

```
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
-D 172.16.37.0/24 2049 2050

# valen las respuestas a otros puertos de usuario
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535

# Rechazar conexiones de entrada a identd Usaremos 'reject' aquí para que
# se le diga al ordenador que intenta conectar que no continúe, si no lo
# hiciéramos, experimentaríamos retrasos mientras ident da un error de
# 'time out'
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Aceptamos algunas conexiones a servicios comunes desde las redes
# 192.168.64 y 192.168.65, que son amistades en las que confiamos.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
       -D 172.16.37.0/24 20:23

# aceptar y dejar pasar cualquier cosa que se origine dentro
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# denegar la mayoría del resto de conexiones TCP y registrarlas
# (añade 1:1023 si tiene problemas con el FTP)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# ... para UDP también
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Las buenas configuraciones de cortafuegos son un poco complicadillas. Este ejemplo debería ser un punto de partida razonable. La página de manual de `ipfwadm` ofrece más asistencia en lo que respecta al manejo de la herramienta. Si intenta configurar un cortafuegos, asegúrese de que hace suficientes preguntas para tener información de las fuentes que considera fiables y haga algunas pruebas de funcionamiento con su configuración desde el exterior de la red.

6.7 IP Firewall (para Linux 2.2)

Al nuevo código para controlar el cortafuegos se accede mediante *IP Firewall Chains*. Consulte la página principal de IP Chains *http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html* si desea más información. Entre otras cosas, necesitará usar `ipchains` en lugar de `ipfwadm` para configurar sus filtros. (De `Documentation/Changes` en las últimas fuentes del núcleo)

Estamos avisados de que esto está seguramente desfasado y estamos trabajando para tener esta sección más al día. Es posible que tengamos una versión nueva para agosto de 1999.

6.8 Encapsulación IPIP

¿Qué razón hay para encapsular datagramas IP dentro de datagramas IP? Le parecerá una tontería si nunca ha visto antes sus aplicaciones. Bien, aquí tenemos un par de ejemplos comunes en los que se usa: Mobile-IP e IP-Multicast. Posiblemente su uso más extendido o al menos el mejor conocido es Amateur Radio.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP: tunneling
```

Los dispositivos túneles de IP se denominan `tunl0`, `tunl1`, etc.

¿Pero por qué?. Bien, bien. Las reglas convencionales de encaminamiento de redes IP comprenden direcciones de red y máscaras de red. Esto hace que conjuntos de direcciones contiguas sean encaminadas mediante una sola regla de encaminamiento. Esto es muy conveniente, pero significa que sólo puede usar una dirección IP en particular cuando está conectado a alguna parte de la red a la que pertenece. En la mayoría de los casos esto vale, pero si suele desplazarse entonces no será capaz siempre de conectar desde el mismo sitio. La encapsulación IP/IP (IP tunneling) le permite saltarse esta restricción permitiendo que los datagramas que están destinados a su dirección IP sean encapsulados y dirigidos a otra dirección IP. Si sabe que va a estar durante un tiempo trabajando en otra red IP entonces podrá poner a punto una máquina de su red habitual para que acepte los datagramas que van dirigidos a su IP y que los reenvíe a la dirección que esté usando de manera temporal.

6.8.1 Una configuración de red con *túneles*

```
192.168.1/24                      192.168.2/24

    -                                  -
    |     ppp0 =            ppp0 =     |
    |  aaa.bbb.ccc.ddd  fff.ggg.hhh.iii  |
    |                                  |
    |   /-----\                /-----\   |
    |   |     |       //       |     |   |
    |---|  A  |------//---------|  B  |---|
    |   |     |     //         |     |   |
    |   \-----/                \-----/   |
    |                                  |
    -                                  -
```

El diagrama ilustra otra posible razón para usar encapsulación IPIP, las redes privadas virtuales. Este ejemplo presupone que tiene dos máquinas cada una con una sola conexión ppp a Internet. Cada máquina tiene una sola dirección IP. Tras cada una de estas máquinas hay algunas redes privadas de área local configuradas con direcciones de red IP reservadas. Supongamos que quiere permitir que cualquier máquina en la red `A` se conecte con cualquier máquina en la red `B`, como si estuvieran conectadas a Internet por una ruta de red. La encapsulación IP se lo permitirá. Tenga en cuenta que la encapsulación no resuelve el problema de cómo hacer que las máquinas las redes `A` y `B` se comuniquen con otras máquinas en Internet, ya que para seguimos necesitando trucos como el Enmascaramiento IP. La encapsulación la realiza normalmente una máquina que esté funcionando como encaminador.

El encaminador Linux `A` debería configurarse así:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Configuración Ethernet
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# Configuración ppp0 (iniciar enlace ppp, establecer ruta por defecto)
pppd
route add default ppp0
#
# Configuración del dispositivo túnel
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask $mask gw $remotegw tunl0
```

El encaminador Linux ‘`B`’ debería configurarse con:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Configuración Ethernet
ifconfig eth0 192.168.2.1 netmask $mask up
route add -net 192.168.2.0 netmask $mask eth0
#
# Configuración ppp0 (iniciar enlace PPP, establecer ruta por defecto)
pppd
route add default ppp0
#
# Configuración del dispositivo túnel
ifconfig tunl0 192.168.2.1 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

La orden:

```
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

se leería: *Envía cualquier datagrama destinado a* `192.168.1.0/24` *dentro de un datagrama IPIP con dirección de destino* `aaa.bbb.ccc.ddd`’.

Las configuraciones son recíprocas en cada extremo. El dispositivo de túnel usa el `gw` dado en la ruta como *destino* del datagrama IP en el que encerrará el datagrama que ha recibido para encaminar. Esa máquina debe saber cómo desencapsular datagramas IPIP, esto es, debe también estar configurada con un dispositivo túnel.

6.8.2 Configuración de la máquina cuyos paquetes serán encapsulados

No tiene por qué estar encaminando una red entera. Por ejemplo puede estar encapsulando una sola dirección IP. En este caso podría configurar el dispositivo `tunl` en la máquina ‘remota’ con su dirección IP y el extremo `A` usará la máquina encaminadora (y *Proxy Arp*) en lugar encaminar la red a través del dispositivo túnel.

Dibujemos de nuevo y modifiquemos apropiadamente nuestra configuración. Ahora tenemos una máquina `B` que quiere actuar y comportarse como si estuviera conectada a Internet y al mismo tiempo fuera parte de una red remota soportada por la máquina `A`:

```
 192.168.1/24

    -
    |      ppp0 =               ppp0 =
    |  aaa.bbb.ccc.ddd      fff.ggg.hhh.iii
    |
    |   /-----\                /-----\
    |   |     |       //       |     |
    |---|  A  |------//---------|  B  |
    |   |     |     //         |     |
    |   \-----/                \-----/
    |                      también: 192.168.1.12
    -
```

El encaminador Linux `A` debería configurarse así:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Configuración Ethernet
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# Configuración ppp0 (iniciar enlace ppp, establecer ruta por defecto)
pppd
route add default ppp0
#
# Configuración del dispositivo túnel
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw $remotegw tunl0
#
# Proxy ARP para la máquina remota
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

El encaminador Linux `B` debería configurarse así:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Configuración ppp0 (iniciar enlace ppp, establecer ruta por
# defecto)
pppd
route add default ppp0
#
# Configuración del dispositivo túnel
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

Esta clase de configuración es más típica de aplicaciones Mobile-IP, donde una máquina quiere moverse por Internet y mantener una única IP fija. Debería leer la sección Mobile-IP (IP Móvil) si quiere más información de cómo se hace esto en la práctica.

6.9 Enmascarado IP (*IP Masquerade*)

Mucha gente dispone de una sola cuenta por conexión telefónica para conectarse a Internet. Casi todos los que utilizan esa configuración se ve limitado a una sola dirección IP que le da el Proveedor de Servicios de Internet. Esto normalmente es bastante para permitir un sólo acceso completo a la red. El Enmascarado IP es un truco inteligente que permite que varias máquinas usen una sola dirección IP, haciendo que las otras máquinas se hagan pasar, y de ahí el término de enmascaramiento, por la máquina que realmente tiene la conexión. Hay un pequeño defecto y es que el enmascarado funciona casi siempre en un sólo sentido, y este es que las máquinas enmascaradas pueden hacer llamadas, pero no pueden aceptar o recibir llamadas de otras máquinas remotas. Esto significa que algunos servicios de red como *talk* no funcionarán y otros como *ftp* deberán ser configurados en modo pasivo (PASV) para que funcionen. Afortunadamente los servicios de red más comunes como *telnet*, WWW e *irc* funcionan bien.

Opciones de Compilación de Red:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Lo normal es que su máquina Linux esté usando una línea slip o PPP como si fuera una máquina aislada. Sin embargo, podemos tener además otros dispositivos de red configurados, quizá una ethernet, configurada con una de las direcciones de red privadas. Las máquinas que van a ser enmascaradas podrían estar en esta segunda red. Cada una de ellas debería tener asignada la dirección IP del puerto Ethernet de la máquina Linux como la pasarela (*gateway*) por defecto.

Una configuración típica podría ser algo parecido a:

```
-                                   -
 \                                  | 192.168.1.0
  \                                 |   /255.255.255.0
   \                 ---------      |
    |                | Linux | .1.1 |
NET =================| masq  |------|
    |    PPP/slip    | router|      |  --------
   /                 ---------      |--| host |
  /                                 |  |      |
 /                                  |  --------
-                                   -
```

Enmascarado con IPFWADM

Las órdenes más importantes para esta configuración son:

```
# Ruta de red hacia la Ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Ruta por defecto al resto de Internet
route add default ppp0
#
# Hace que todas las máquinas de la red 192.168.1/24 sean
# enmascarados
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Enmascarado con IPCHAINS

```
# Ruta de red hacia la Ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Ruta por defecto al resto de Internet
route add default ppp0
#
# Hace que todas las máquinas de la red 192.168.1/24 sean
# enmascarados
ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

Puede obtener más información sobre Enmascarado IP de Linux en la página de recursos de IP Masquerade *http://www.hwy401.com/achau/ipmasq*. También hay un documento *muy* detallado que habla sobre enmascaramiento, que es el *IP-Masquerade-Como* (que también enseña a configurar otros sistemas operativos para trabajar teniendo un Linux como servidor enmascarador).

6.10 Proxy IP transparente

El proxy transparente de IP es una característica que le permite redirigir servidores o servicios destinados a otra máquina a esos servicios en esta máquina. Normalmente esto es útil cuando tiene un Linux como encaminador que además proporciona un servidor proxy. Debería redirigir todas las conexiones destinadas a este servicio de forma remota al servidor local de proxy.

Opciones de Compilación del Núcleo:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    ....
    [*] IP: firewalling
    ....
    [*] IP: transparent proxy support (EXPERIMENTAL)
```

La configuración de la característica de proxy transparente se realiza mediante la orden `ipfwadm`.

Un ejemplo que podría ser útil es el siguiente:

```
root# ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

Este ejemplo hará que cualquier intento de conexión al puerto `telnet` (23) por parte de otra máquina sea redirigida al puerto 2323 de esta máquina. Si tiene un servicio activo en ese puerto, podría redirigir las conexiones telnet, hacer un registro de lo que pasa, o cualquier cosa que se ajuste a sus necesidades.

Un ejemplo más interesante es redirigir todo el tráfico `http` a través de un almacén caché local. Sin embargo, el protocolo usado por los servidores proxy es diferente al http nativo: mientras que un cliente conecta a `www.servidor.com:80` y pregunta por `/camino/página`, cuando conecta a la caché local busca `proxy.dominio.local:8080` y pregunta por `www.servidor.com/camino/página`.

Para filtrar una petición `http` a través del proxy local, necesitará adaptar el protocolo insertando un pequeño servidor llamado `transproxy` (lo puede encontrar en la web). Si quiere puede ejecutarlo en el puerto 8081, y ejecutar esta orden:

```
root# ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Entonces, el programa `transproxy` recibirá todas las conexiones que tengan que alcanzar servidores externos y las pasará al proxy local arreglando las diferencias de protocolo.

6.11 IPv6

¡Justo ahora que creía que empezaba a entender las redes IP las reglas cambian! IPv6 es la notación abreviada de la versión 6 del Internet Protocol. IPv6 fue desarrollada principalmente para responder a los temores de la comunidad de Internet al respecto de que pronto habrá escasez de direcciones IP para asignar. Las direcciones IPv6 son de 16 bytes (128 bits). IPv6 incorpora varios cambios más, en su mayor parte simplificaciones, que harán que las redes IPv6 sean más manejables que las IPv4.

Linux ya tiene una implementación de IPv6 que funciona, pero no está completa, a partir de la serie de núcleos `2.2.*`.

Si quiere experimentar con esta próxima generación de tecnología de Internet, o le hace falta, debería leerse la *IPv6-FAQ* que está disponible en *http://www.terra.net/ipv6*.

6.12 Mobile IP

El término *movilidad de IP* describe la habilidad de una máquina que es capaz de mover su conexión de red de un punto de Internet a otro sin cambiar su dirección IP o perder conectividad. Normalmente cuando una máquina con IP cambia su punto de conexión también debe cambiar su dirección IP. La Movilidad de IP soluciona este problema asignando una IP fija a la máquina móvil y usando encapsulación IP (*tunneling*) con encaminado automático para asegurar que los datagramas destinados a ella se encaminan a la verdadera dirección IP que esté usando en ese momento.

Está en camino un proyecto para proporcionar un juego completo de herramientas de movilidad de IP para Linux. El estado del proyecto y las herramientas los puede obtener de la página principal de Linux Mobile IP *http://anchor.cs.binghamton.edu/~mobileip*.

6.13 Multicast

IP Multicast permite encaminar datagramas IP hacia a un número arbitrario de máquinas con IP de forma simultánea. Este mecanismo se explota para proporcionar material de amplia distribución por Internet como transmisiones de imagen y sonido y otras aplicaciones noveles.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    [*] TCP/IP networking
    ....
    [*] IP: multicasting
```

Esto requiere un conjunto de herramientas y algo de configuración de la red. Una fuente de información sobre cómo instalar y configurar esto para Linux la encontramos en *http://www.teksouth.com/linux/multicast*.

6.14 NAT - Network Address Translation (Traducción de direcciones de red)

El servicio de Traducción de Direcciones de Redes IP es algo así como el hermano mayor estandarizado del servicio de Enmascarado IP de Linux. Está especificado con cierto detalle en el *RFC-1631* en su archivo RFC más próximo. NAT proporciona características que no posee el Enmascarado IP que lo hacen eminentemente más apropiado para su uso en los diseños de encaminamiento de cortafuegos corporativos y en instalaciones a mayor escala.

Michael Hasenstein, *Michael.Hasenstein@informatik.tu-chemnitz.de*, ha desarrollado una implementación alfa de NAT para el núcleo `2.0.29` de Linux. La documentación e implementación de Michaels está disponible en: la página Web de *Linux IP Network Address* *http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html*

Los núcleos más modernos de Linux `2.2.x` también incluyen algo de la funcionalidad NAT en el algoritmo de encaminamiento.

6.15 Traffic Shaper (Manipulación del ancho de banda)

El traffic shaper (regulador de caudal) es un controlador que crea nuevas interfaces de red, las cuales tienen una limitación en el caudal, definida por el usuario, y que actúan sobre una interfaz de red física para hacer la comunicación real, pudiendo ser utilizadas por el encaminador para el tráfico saliente.

El regulador fue introducido con Linux-2.1.15 y fue portado a Linux-2.0.36 (apareció en el `2.0.36-pre-patch-2` distribuido por Alan Cox, el autor del dispositivo regulador y mantenedor de Linux-2.0).

El regulador de caudal sólo puede ser compilado como módulo y para configurarlo se usa el programa *shapecfg* con órdenes como las siguientes:

```
shapecfg attach shaper0 eth1
shapecfg speed shaper0 64000
```

El dispositivo regulador sólo puede controlar el ancho de banda del tráfico de salida, ya que los paquetes son transmitidos a través suyo de acuerdo con las tablas de rutas; sin embargo, una funcionalidad de *rutas por dirección de origen* podría ayudar a limitar el ancho de banda global de máquinas específicas que estén usando un encaminador Linux.

Linux-2.2 implementa también ese tipo de encaminamiento, y si lo necesita en Linux-2.0, busque el parche de Mike McLagan, en *ftp://ftp.invlogic.com*. Lea `Documentation/networking/shaper.txt` si desea más información al respecto del *shaper*.

Si quiere probar un intento de regulación de caudal de paquetes entrantes, use `rshaper-1.01` (o más nuevo). Lo encontrará en *ftp://ftp.systemy.it/pub/develop*.

6.16 Encaminamiento con Linux-2.2

La última versión de Linux, la 2.2, ofrece mucha flexibilidad en lo que a política de encaminamiento se refiere. Desafortunadamente, deberá esperar a la siguiente versión de este Como, o acudir a las fuentes del núcleo.

7 Uso de hardware común en los PC

7.1 RDSI

La Red Digital de Servicios Integrados (RDSI) es una serie de estándares que especifican una red digital conmutada de propósito general. Una llamada RDSI crea un servicio de datos asíncrono punto a punto hacia el destino. RDSI suele distribuirse con un enlace de gran velocidad que se divide en varios canales discretos. Hay dos tipos de canales diferentes, los *Canales B* que son los que realmente transportan los datos del usuario y un sólo canal llamado *canal D* que se usa para enviar información de control a la RDSI para establecer llamadas y otras funciones. En Australia, por ejemplo, RDSI se distribuye con un enlace de 2Mbps que se divide en 30 canales discretos B de 64kbps con un canal D de 64kbps. Se pueden usar cualquier número de canales en cualquier momento y en cualquier combinación. Usted podría, por ejemplo, establecer 30 llamadas separadas a 30 destinos diferentes a 64kbps cada uno, o 15 llamadas a 15 destinos diferentes a 128kbps cada uno (dos canales por llamada), o sencillamente un pequeño número de llamadas y dejar el resto de los canales sin usar. Se puede usar un canal tanto para llamadas entrantes como salientes. La intención original de la RDSI fue permitir a las compañías de telecomunicaciones proporcionar un sólo servicio de datos que pudiera transmitir tanto teléfono (mediante voz digitalizada) como servicios de datos a su casa o negocio sin que tuviera que hacer ningún cambio especial de configuración.

Hay unas pocas maneras diferentes de conectar su ordenador a un servicio RDSI. Una manera es usar un dispositivo llamado *Adaptador Terminal* que se enchufa a la Unidad Terminadora de Red que su compañía local de telecomunicaciones tuvo que instalar al proporcionarle el servicio RDSI y que presenta unas cuantas interfaces serie. Uno de estos interfaces se usa para introducir instrucciones de establecimiento de llamadas y la configuración y los otros son los que realmente se conectan a los dispositivos de red que usarán los circuitos de datos cuando se hayan establecido. Linux trabajará con este tipo de configuración sin modificación alguna, simplemente tiene que tratar el puerto del Adaptador Terminal como trataría cualquier otro dispositivo serie. Otra manera, que es para la que está diseñado el soporte de RDSI del núcleo permite instalar una tarjeta RDSI en su máquina Linux y dejar entonces que sea el propio Linux quien maneje los protocolos y haga las llamadas.

Opciones de Compilación del Núcleo:

```
ISDN subsystem  --->
    <*> ISDN support
    [ ] Support synchronous PPP
    [ ] Support audio via ISDN
    < > ICN 2B and 4B support
    < > PCBIT-D support
    < > Teles/NICCY1016PC/Creatix support
```

La implementación del núcleo de Linux de la RDSI soporta varios tipos diferentes de tarjetas RDSI. Son las que vienen listadas en las opciones de configuración del núcleo.

- ICN 2B and 4B
- Octal PCBIT-D

- Tarjetas Teles ISDN y compatibles

Algunas de estas tarjetas requieren que se les carguen ciertos programas antes de funcionar. Hay una utilidad aparte que hace esto.

Hay disponibles más detalles de cómo configurar el soporte RDSI de Linux en el directorio `/usr/src/linux/Documentation/isdn/` y también un listado *PUF* (Pregutas de Uso Frecuente) dedicado a `isdn4linux` en *http://www.lrz-muenchen.de/~ui161ab/www/isdn*. (Puede pulsar sobre la bandera inglesa para la versión en inglés).

Una nota al respecto de PPP. El conjunto de protocolos PPP operará tanto sobre lineas serie síncronas como asíncronas. El demonio PPP (`pppd`) que se suele distribuir para Linux sólo soporta el modo asíncrono. Si desea ejecutar los protocolos PPP sobre el servicio RDSI, necesitará una versión especial modificada. En la documentación anteriormente mencionada encontrará detalles de cómo encontrarlo.

En *http://www.insflug.org/documentos/RDSI-Como/* tiene a su disposición un documento con detalles específicos de la configuración en España.

7.2 PLIP en Linux-2.0

Los nombres de los dispositivos PLIP son '`plip0`', '`plip1`' y '`plip2`'.

Opciones de Compilación del Núcleo:

```
Network device support  --->
    <*> PLIP (parallel port) support
```

plip (Parallel Line IP), se parece a SLIP en que se usa para proporcionar conexiones *punto a punto* entre dos máquinas, excepto en que está diseñado para usar los puertos paralelos de impresora de la máquina en lugar de los puertos serie (se incluye un diagrama de cableado en la sección de diagramas de cableado más adelante). Como es posible transmitir más de un bit a la vez con un puerto paralelo, es posible obtener mayores velocidades con la interfaz *plip* que con los dispositivos serie estándar. Además, puede aprovecharse incluso el más sencillo de los puertos paralelos, el puerto de impresora, en lugar de tener que comprar una UART 16550AFN, comparativamente más cara, para los puertos serie. PLIP usa mucha CPU si lo comparamos con un enlace serie y casi seguro que no será una buena opción si puede obtener algunas tarjetas Ethernet baratas, pero funcionará cuando no esté disponible nada más y además lo hará bastante bien. Se puede esperar una tasa de transferencia de alrededor de 20 kilobytes por segundo cuando el enlace está funcionando correctamente.

Los controladores de dispositivo de PLIP compiten con el controlador de dispositivo paralelo por el hardware del puerto paralelo. Si desea usar ambos controladores deberá compilar los dos como módulos para asegurarse de que es capaz de elegir qué puerto quiere que use PLIP y qué puertos usará como dispositivos para la impresora. Lea el *Modules mini-Howto* para obtener más información sobre la configuración de los módulos en el núcleo.

Por favor tenga en cuenta que algunos portátiles usan circuitería que no funcionará con PLIP porque no permiten algunas combinaciones de señales en las que se basa PLIP, que las impresoras no usan.

El interfaz `plip` de Linux es compatible con el *Crynwyr Packet Driver PLIP* y esto significa que puede conectar su máquina Linux a una máquina DOS que esté ejecutando cualquier clase de programas tcp/ip mediante `plip`.

En la serie de núcleos 2.0.* los dispositivos PLIP están asignados a puertos e IRQ como sigue

```
dispositivo  i/o    IRQ
```

```
----------- -----   ---
plip0       0x3bc   5
plip1       0x378   7
plip2       0x278   2
```

Si sus puertos paralelos no se ajustan a alguna de las combinaciones de encima entonces tendrá que cambiar la IRQ de un puerto usando el parámetro `irq` de la orden `ifconfig`. Asegúrese de que activa las IRQ de los puertos de impresora en la ROM BIOS antes, si es que dispone de esa opción. Como alternativa, puede especificar las opciones `io=` e `irq=` como argumentos a `insmod`, si usa módulos. Por ejemplo:

```
root# insmod plip.o io=0x288 irq=5
```

La operación PLIP se controla con dos retardos, cuyos valores por defecto suelen ser correctos. Posiblemente necesite incrementarlos si tiene una máquina un tanto lenta, en cuyo caso los temporizadores a incrementar están en la **otra** máquina. Hay un programa llamado `plipconfig` que permite cambiar la configuración de los temporizadores sin recompilar el núcleo. Viene con muchas distribuciones Linux.

Para configurar una interfaz *plip*, necesitará invocar las siguientes órdenes (o **añadirlas** a su guión de iniciación del sistema):

```
root# /sbin/ifconfig plip1 pliplocal pointopoint plipremota
root# /sbin/route add plipremota plip1
```

Aquí, el puerto usado es el que está en la dirección E/S `0x378`; `pliplocal` y `plipremota` son los nombres o direcciones IP usadas para los extremos del cable PLIP. Personalmente, prefiero tenerlas en mi base de datos `/etc/hosts`:

```
# entradas plip
192.168.3.1   pliplocal
192.168.3.2   plipremota
```

El parámetro `pointopoint` tiene el mismo significado que para SLIP, en el que especifica la dirección de la máquina en el otro extremo del enlace.

En casi todos los aspectos, se puede tratar una interfaz `plip` como si fuese una interfaz `SLIP`, excepto que no se puede (ni se necesita) usar `dip` ni `slattach`.

Para obtener más información al respecto de PLIP, acuda al *PLIP mini-Howto*.

7.3 PLIP en Linux-2.2

Durante el desarrollo de las versiones 2.1 del núcleo, se cambió la implementación del puerto paralelo, hacia una configuración mejor.

Opciones de compilación del núcleo:

```
General setup  --->
    [*] Parallel port support
Network device support  --->
    <*> PLIP (parallel port) support
```

El nuevo código de PLIP se comporta como el anterior (se usan las mismas órdenes `ifconfig` y `route` que en la sección anterior, pero la iniciación del dispositivo es diferente debido a la avanzada implementación del manejo del puerto paralelo).

El «primer» dispositivo PLIP siempre es denominado `plip0`, siendo éste el que primero detecta el sistema, de manera similar a lo que ocurre con los dispositivos Ethernet. El verdadero puerto paralelo que se va a usar es uno de los que estén disponibles, tal como se muestra en `/proc/parport`. Por ejemplo, si sólo tiene un puerto paralelo, sólo tendrá un directorio llamado `/proc/parport/0`.

Si el núcleo no detecta la IRQ usada por el puerto, `insmod plip` fallará; en este caso simplemente tendrá que escribir el número correcto en `/proc/parport/0/irq` y reinvocar `insmod`.

Hay disponible información completa al respecto del puerto paralelo en el archivo `Documentation/parport.txt`, parte de las fuentes de su núcleo.

7.4 PPP

Los nombres de los dispositivos PPP son `ppp0`, `ppp1`, etc. Los dispositivos están numerados de forma secuencial de manera que el primer dispositivo configurado recibe el `0`.

Opciones de compilación del Núcleo:

```
Networking options  --->
    <*> PPP (point-to-point) support
```

La configuración del PPP está cubierta en detalle en el PPP-Como, *http://www.insflug.org/documentos/PPP-Como/*.

7.4.1 Mantener una conexión permanente a la red usando pppd.

Si es tan afortunado como para tener una conexión semi-permanente a la red y quiere hacer que su máquina establezca automáticamente la conexión PPP si se pierde, existe un truco sencillo para hacerlo.

Configure el PPP de tal manera que pueda ser iniciado por el usuario `root` ejecutando la orden:

```
# pppd
```

Asegúrese de que tiene configurada la opción `-detach` en el fichero `/etc/ppp/options`. Entonces, inserte la siguiente línea en el fichero `/etc/inittab`, bajo las definiciones de `getty`.

```
pd:23:respawn:/usr/sbin/pppd
```

Esto hará que el programa `init` ejecute y monitorice el programa `pppd` y que lo reinicie automáticamente si muere.

7.5 Cliente SLIP

Los nombres de los dispositivos SLIP son `sl0`, `sl1`, etc. siendo asignado el `0` al primer dispositivo configurado y el resto incrementando secuencialmente según van siendo configurados.

Opciones de compilación del núcleo:

```
Network device support  --->
    [*] Network device support
    <*> SLIP (serial line) support
    [ ]  CSLIP compressed headers
    [ ]  Keepalive and linefill
    [ ]  Six bit SLIP encapsulation
```

El SLIP (*Serial Line Internet Protocol*) le permite usar tcp/ip sobre una línea serie, sea una línea de teléfono con un módem, o una línea dedicada. Por supuesto para usar SLIP necesitará tener acceso a un *servidor SLIP* de algún tipo. Muchas universidades y empresas de todo el mundo proporcionan acceso mediante SLIP.

Slip usa los puertos serie de su máquina para transportar datagramas IP. Para hacerlo debe tomar el control del dispositivo serie. Los dispositivos SLIP se denominan `sl0`, `sl1`, etc. ¿Cómo se corresponden a sus dispositivos serie? El código de red usa lo que denominados una llamada *ioctl* (i/o control - control de e/s) para convertir los dispositivos serie en dispositivos SLIP. Se le proporcionan dos programas que pueden hacer esto, y se llaman `dip` y `slattach`.

7.5.1 dip

`dip` (Dialup IP) es un sofisticado programa capaz de programar la velocidad del dispositivo serie, ordenar a su módem que llame al otro extremo del enlace, identificarle automáticamente en el servidor remoto, examinar los mensajes que le envía el servidor y obtener información tal como su dirección IP y ejecutar las *ioctl* necesarias para conmutar su puerto serie al modo SLIP. `dip` tiene una gran capacidad para ejecutar guiones (scripts) y esto lo puede explotar para automatizar el procedimiento de autenticación.

Puede encontrarlo en: *ftp://metalab.unc.edu/pub/Linux/system/Network/serial/dip/dip337o-uri.tgz*.

Para instalarlo, intente lo siguiente:

```
usuario% cd /usr/src
usuario% gzip -dc dip337o-uri.tgz | tar xvf -
usuario% cd dip-3.3.7o
root# make install
```

El `Makefile` asume la existencia de un grupo denominado `uucp`, pero puede que usted quiera cambiarlo a `dip` o `SLIP` dependiendo de su configuración.

7.5.2 slattach

`slattach` es un programa muy sencillo comparado con `dip`, muy fácil de usar, pero no tiene la sofisticación de `dip`. No tiene capacidad para ejecutar guiones, todo lo que hace es configurar su dispositivo serie como dispositivo SLIP. Asume que tiene toda la información que necesita y la línea serie se establece antes de que lo invoque. `slattach` es ideal para usarlo donde necesite una conexión permanente al servidor, como un cable físico o una línea dedicada.

7.5.3 ¿Cuándo usar cada uno?

Debería usar `dip` cuando su enlace a la máquina servidora de SLIP es un módem, o algún otro enlace temporal. Debería usar `slattach` cuando disponga de una línea dedicada, quizá un cable, entre su máquina y el servidor, y no se necesita ejecutar ninguna acción especial para hacer que el enlace funcione. Mire la sección 7.5.7 (Conexión Permanente con Slip) para más información.

configurar SLIP se parece mucho a configurar una interfaz Ethernet (lea la sección 5.4 (Configuración de un dispositivo Ethernet) más atrás). Sin embargo, hay unas pocas diferencias clave.

Antes que nada, los enlaces SLIP son diferentes a las redes Ethernet en que sólo hay dos máquinas en la red, una en cada extremo del enlace. Al contrario que con Ethernet que está disponible para su uso nada más que termine de cablear, con SLIP, dependiendo del tipo de enlace que tenga, puede que tenga que iniciar la conexión de red de alguna manera especial.

Si está usando `dip` entonces esto no debería hacerlo en el momento de arrancar, sino algo más adelante, cuando esté preparado para usar el enlace. Es posible automatizar este procedimiento. Si está usando `slattach` entonces probablemente quieras añadir una sección a su fichero `rc.*` correspondiente. Esto lo describiremos pronto.

Hay dos tipos principales e servidores SLIP: servidores de direcciones IP dinámicas y servidores de direcciones IP estáticas. Casi cualquier servidor SLIP le pedirá al conectar una identificación y una contraseña. `dip` puede proporcionar estos datos automáticamente.

7.5.4 Servidor SLIP estático con línea por llamada y DIP.

Un servidor estático de SLIP es aquél en el que se le ha dado una dirección IP que es exclusivamente suya. Cada vez que conecte al servidor, tendrá que configurar su puerto SLIP con esa dirección. El servidor estático de SLIP contestará a la llamada del módem, posiblemente le pregunte su nombre de usuario y contraseña, y entonces encaminará cualquier datagrama destinado a su dirección mediante esa conexión. Si tiene un servidor estático, entonces puede que quiera añadir una entrada con su dirección IP y el nombre de su máquina (ya que sabe cuales serán) en `/etc/hosts`. También debería configurar algunos otros ficheros como: `rc.*`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` y `rc.local`. Recuerde que cuando configure `rc.*`, no necesita añadir ninguna orden especial para la conexión SLIP ya que es `dip` el que hace todo el trabajo duro al configurar la interfaz. Necesitará proporcionarle a `dip` la información apropiada de manera que configure la interfaz por usted después de ordenarle al módem que establezca la llamada y de darle de alta en el servidor SLIP.

Si esta es la manera de trabajar de su servidor SLIP entonces puede pasar a la sección 7.5.6 (Uso de Dip) para aprender a configurar `dip` de forma apropiada.

7.5.5 Servidor SLIP dinámico con línea por llamada y DIP.

Un servidor *dinámico* de SLIP es aquél que le asigna una dirección IP al azar, a partir de una reserva de direcciones, cada vez que se registra. Eso significa que no hay garantías de que tenga una dirección en particular cada vez, y esa dirección puede ser usada por otra persona después de que usted haya desconectado. El administrador de red que configuró el servidor SLIP habrá asignado un rango de direcciones para que el servidor SLIP las use, cuando el servidor recibe una nueva llamada entrante, localiza la primera dirección sin asignar, guía a quien llama a través del procedimiento de registro y entonces imprime un mensaje de bienvenida que contiene la dirección IP que se le ha asignado, para entonces proceder a usar esa dirección IP durante toda la llamada.

La configuración para este tipo de servidor es similar a la configuración para un servidor estático, excepto en que debe añadir un paso en el que obtiene la dirección IP que el servidor le ha asignado y configura su dispositivo SLIP con ella.

De nuevo, `dip` hace el trabajo duro y las nuevas versiones son suficientemente sofisticadas no sólo como para registrarle, sino también como para leer automáticamente la dirección IP impresa en el mensaje de bienvenida y almacenarla de manera que pueda configurar el dispositivo SLIP con ella.

Si esta es la manera de trabajar de su servidor SLIP, entonces puede pasar a la sección 7.5.6 (Uso de Dip) para aprender a configurar `dip` de forma apropiada.

7.5.6 Uso de Dip.

Como se explicó anteriormente, `dip` es un programa potente que puede simplificar y automatizar el proceso de llamada a el servidor SLIP, registro, inicio de la conexión y configuración de tus dispositivos SLIP con las órdenes `ifconfig` y `route` apropiadas.

Esencialmente, para usar `dip` escribiremos un «guión dip», que es básicamente una lista de órdenes que `dip` comprende y que le dicen cómo debe realizar cada una de las acciones que usted quiere que haga. Eche un vistazo al fichero `sample.dip` que viene con `dip` para hacerse una idea de cómo funciona. `dip` es un programa bastante potente, con muchas opciones. No las va a encontrar todas aquí, y tendrá que mirar la correspondiente página `man`, el `README` y los ficheros de ejemplo que vendrán con su versión de `dip`.

Fíjese en que el guión `sample.dip` asume que está accediendo a un servidor SLIP estático, por lo que su IP ya se conoce de antemano. Las nuevas versiones de `dip` incluyen, para los servidores SLIP dinámicos, una orden que puede usarse para leer y configurar automáticamente el dispositivo SLIP con la dirección IP que el servidor dinámico le asigna. El ejemplo siguiente es una versión modificada y traducida del `sample.dip` que viene con `dip337j-uri.tgz` y probablemente sea un buen punto de partida. Puede que quiera grabarlo como `/etc/guiondip` y editarlo para que se ajuste a su configuración.

```
#
# sample.dip    Programa de soporte de conexión para Dialup IP
#
#               Este fichero (debería mostrar) muestra el uso de DIP
#       Este fichero debería funcionar para los servidores dinámicos
#       de tipo Annex, si usted usa un servidor estático entonces utilice
#       el fichero sample.dip que viene en el paquete dip337-uri.tgz
#
#
# Versión:      @(#)sample.dip  1.40    20/07/93
#
# Autor:        Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Lo siguiente es configurar el nombre y dirección del otro extremo
# La máquina a la que llamo se llama 'xs4all.hacktic.nl'
# (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Asignar 255.255.255.0 como máscara de red de sl0
netmask 255.255.255.0
# Asignar el puerto serie y velocidad deseados
port cua02
speed 38400

# Reiniciar el módem y la línea terminal
# ¡Esto parece darle problemas a algunas personas!
reset

# ¡Nota! Valores "estándar" predefinidos de "errlevel"
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# Puede cambiarlos buscando "addchat()" en *.c...

# Nos preparamos para llamar.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto problema_con_modem
dial 555-1234567
if $errlvl != 1 goto problema_con_modem
```

```
# Estamos conectados. Registrarse en el sistema.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto problema_al_registrarse
send MILOGIN\n
wait ord: 20
if $errlvl != 0 goto clave_erronea
send MIPASSWD\n
loggedin:

# Ahora estamos registrados.
wait SOMEPROMPT 30
if $errlvl != 0 goto error_de_prompt

# Ordenamos al servidor que entre en modo SLIP
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto error_de_prompt

# Obtenemos nuestra dirección IP del servidor y la asignamos
#   Aquí asumiremos que tras ordenar al servidor SLIP que entre
#   en modo SLIP, nos imprime nuestra dirección IP.
get $local remote 30
if $errlvl != 0 goto prompt_error

# Configurar los parámetros operativos de SLIP
get $mtu 296
# Aseguramos que se ejecuta "route add -net default xs4all.hacktic.nl"
default

# ¡Decimos hola y arrancamos!
done:
print CONNECTED $local ---> $rmtip
mode CSLIP
goto salir

error_de_prompt:
print Se produjo un TIME-OUT esperando a que arrancara sliplogin
goto error

problema_al_registrarse:
print Hubo un problema esperando por el mensaje Login:...
goto error

clave_erronea:
print Hubo un problema esperando por el mensaje Password:...
goto error

problema_con_modem:
print Hubo un problema con el módem...
error:
print FALLÓ LA CONEXIÓN a $remote
quit
```

```
salir:
exit
```

El ejemplo anterior asume que estamos llamando a un servidor SLIP *dinámico*. Si está llamando a un servidor SLIP *estático*, entonces debería funcionarle el fichero `sample.dip` que viene con `dip-337-uri.tgz`.

Cuando se le da a `dip` la orden `get $local`, busca en el texto que viene del otro extremo una cadena que se parezca a una dirección IP, osea una cadena de números separados por caracteres «.». Esta modificación se hizo específicamente pensando en los servidores SLIP *dinámicos*, para que el proceso de leer la dirección IP pudiera ser automatizado.

El ejemplo anterior creará automáticamente una ruta por defecto a través del enlace SLIP. Si no es lo que deseaba, a lo mejor porque tiene una conexión Ethernet que debería ser la ruta por defecto, entonces elimine la orden `default` del guión. Después de que haya acabado de ejecutarse el guión, si ejecuta la orden `ifconfig`, verá que hay un dispositivo `sl0`. Este es el dispositivo SLIP. Si fuese necesario, podría modificar su configuración a mano, después de que la orden `dip` haya acabado, usando las órdenes `ifconfig` y `route`.

Por favor, observe que `dip` le permite elegir varios protocolos diferentes en la orden `mode`. El ejemplo más común es *cSLIP* que es SLIP con compresión. Por favor, tenga en cuenta que ambos extremos del enlace deben concordar, por lo que debe asegurarse de que elija lo que elija, sea la mismo que lo que tenga el servidor.

El ejemplo anterior es bastante robusto y debería copar con la mayoría de los errores. Para obtener más información, haga el favor de dirigirse a las páginas man de `dip`. Naturalmente podría, por ejemplo, codificar un guión para que haga cosas como llamar de nuevo al servidor si no consigue conectarse tras un periodo de tiempo determinado, o incluso hacer intentos con varios servidores si tiene acceso a más de uno.

7.5.7 Conexión SLIP permanente usando una línea dedicada y slattach

Si lo que tiene es un cable entre dos máquinas, o es tan afortunado como para tener una línea dedicada, o algún otro tipo de conexión en serie permanente entre su máquina y otra, entonces no necesita la complejidad de usar `dip` para establecer un enlace serie. `slattach` es una utilidad muy sencilla de usar, que le permite suficiente funcionalidad como para configurar una conexión.

Como la conexión será permanente, querrá añadir algunas órdenes al fichero `rc` pertinente. En esencia, todo lo que necesita hacer para tener una conexión permanente es asegurarse de que configura el dispositivo serie a la velocidad correcta y que lo pasa a modo SLIP. `slattach` permite hacer esto con una sola orden. Añade lo siguiente al fichero `rc` pertinente:

```
#
# Activar una conexión SLIP estática por línea dedicada
#
#  configuramos /dev/cua0 para 19.2kbps y cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# Fin de SLIP estático
```

Nota del traductor: A fecha de la traducción, la serie estable del núcleo era la `2.2`. En la serie `2.2`, los dispositivos `/dev/cuaX` han sido completamente eliminados, unificándolos a los `/dev/ttySX`.

Donde:

IPA.IPA.IPA.IPA

representa su dirección IP.

IPR.IPR.IPR.IPR

representa la dirección IP del extremo remoto.

`slattach` asigna el primer dispositivo SLIP sin asignar al dispositivo serie especificado. `slattach` comienza por `sl0`. Por lo tanto la primera orden `slattach` asocia el dispositivo SLIP `sl0` al dispositivo serie especificado y `sl1` la siguiente vez, etc.

`slattach` le permite configurar varios protocolos diferentes con el argumento `-p`. En su caso podrá usar tanto `SLIP` como `cSLIP` dependiendo de si quiere o no usar compresión. Nota: ambos extremos deben estar de acuerdo sobre si usar compresión o no.

7.6 Servidor SLIP.

Si tiene una máquina, quizá conectada a una red, a la que le gustaría que otra máquina pudiese llamar y proporcionar servicios de red, entonces necesita configurarla como servidor. Si quiere usar SLIP como protocolo de línea serie, entonces tiene tres opciones para configurar su máquina Linux como servidor SLIP. Yo preferiría usar la primera presentada, `sliplogin`, ya que parece la más sencilla de configurar y entender, pero presentaré un sumario de cada una, para que pueda tomar su propia decisión.

7.6.1 Servidor slip usando `sliplogin`.

`sliplogin` es un programa, que puede asignar a los usuarios de SLIP en lugar de un «shell» normal, que convierte una línea terminal en una línea SLIP. Esto le permite configurar una máquina Linux tanto como *servidor de direcciones estáticas*, obteniendo los usuarios la misma dirección cada vez que llaman, o como *servidor de direcciones dinámicas*, donde los usuarios obtienen una dirección que no tiene que ser necesariamente la misma que la última vez que llamaron.

El que llama se registrará igual que en un proceso estándar de `login`, introduciendo su nombre de usuario y contraseña, pero en lugar de presentarle un intérprete de órdenes normal tras el registro, se ejecuta `sliplogin`, el cual busca en su fichero de configuración (`/etc/slip.hosts`) una entrada con el nombre de usuario que corresponda a la persona que llamó. Si lo encuentra, entonces configura la línea como «clean» y de 8 bit, y hace una llamada *ioctl* para convertir la disciplina de la línea a SLIP. Cuando se completa este proceso, toma lugar la última parte de la configuración, en la cual `sliplogin` invoca un guión de intérprete de órdenes que configura la interfaz SLIP con la dirección ip y máscara de red relevantes, y asigna las rutas apropiadas. Este guión suele llamarse `/etc/slip.login` pero, de forma parecida a `getty`, si ciertos usuarios requieren una iniciación especial, entonces puede crear guiones de configuración llamados `/etc/slip.login.nombreusuario` que serán ejecutados específicamente para ellos en lugar del que hay por defecto.

También hay tres o cuatro ficheros que necesitará configurar para tener `sliplogin` en funcionamiento. Detallaré cómo y dónde obtener los programas y cómo se configura cada uno. Los ficheros son:

- `/etc/passwd`, para las cuentas de los usuarios que llaman.
- `/etc/slip.hosts`, para almacenar la información única a cada usuario que llama.
- `/etc/slip.login`, que lleva la configuración del encaminamiento que necesite hacerse para el usuario.
- `/etc/slip.tty`, que será necesario sólo si está configurando el servidor para asignar direcciones dinámicas y contiene una tabla de direcciones a asignar.
- `/etc/slip.logout`, que contiene las órdenes necesarias para dejar las cosas a punto una vez el usuario cuelgue o cierre la conexión.

Dónde obtener `sliplogin` Puede que ya tenga el paquete `sliplogin` instalado como parte de su distribución; si no es así entonces podrá obtenerlo de: *ftp://metalab.unc.edu/pub/linux/system/Network/serial/sliplogin-2.1.1.tar.gz*. El fichero tar contiene tanto las fuentes, como los programas ya compilados y una página *man*.

para asegurar que sólo son capaces de ejecutar `sliplogin` los usuarios autorizados, debería añadir una entrada al fichero `/etc/group` similar a la siguiente:

```
 ..
slip::13:radio,fred
 ..
```

Cuando instale el paquete `sliplogin`, el `Makefile` cambiará el grupo al que pertenece el programa `sliplogin` por `slip`, y esto significa que sólo los usuarios que pertenezcan a ese grupo serán capaces de ejecutarlo. El ejemplo anterior autorizará ejecutar `sliplogin` sólo a los usuarios `radio` y `fred`.

Para instalar los ejecutables en el directorio `/sbin` y la página de *man* en la sección 8, haga lo siguiente:

```
# cd /usr/src
# gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..edite el Makefile si no usa shadow password..>
# make install
```

Si quiere recompilar los ejecutables tras la instalación, añada un `make clean` antes de `make install`. Si quiere instalar los ejecutables en cualquier otro lado, necesitará editar la regla `install` en el `Makefile`.

Por favor, lea los ficheros `README` que vienen con el paquete si quiere obtener más información.

Configuración de `/etc/passwd` **para máquinas que usan Slip.** Lo normal es que cree algunos *login* especiales en el fichero `/etc/passwd` para la gente que llama con Slip. Una convención comúnmente aceptada es usar el nombre de la máquina que llaman prefijándole una letra «S» mayúscula. Por tanto, por ejemplo, si la máquina que llama se denomina `radio`, entonces podrías crear una entrada en `/etc/passwd` que se pareciese a:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

Realmente no importa cómo se llame la cuenta, siempre y cuando tenga algún significado para usted.

Nota: el que llama no necesita ningún directorio personal, ya que no se le presentará ningún intérprete de órdenes, por tanto `/tmp` es una buena elección. Observe también que se usa `sliplogin` en lugar de un intérprete de órdenes normal.

Configuración de `/etc/slip.hosts` El fichero `/etc/slip.hosts` es el fichero en el que *sliplogin* busca las entradas que se correspondan al nombre de login que de las que obtiene los detalles de configuración para esa persona. Es aquí donde se especifica la dirección ip y la máscara de red que se le asignará al que llama y que serán configuradas para su uso. Las entradas para dos máquinas, una con configuración estática para `radio` y otra, con configuración dinámica, para la máquina `albert`, deberían parecerse a lo siguiente:

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Las entradas del fichero `/etc/slip.hosts` son

1. el nombre de login del que llama.
2. la dirección ip de la máquina servidor, osea esta máquina.
3. la dirección IP que se asignará al que llama. Si este campo se rellena con `DYNAMIC` entonces se asignará una dirección ip basada en la información contenida en el fichero `/etc/slip.tty` del que se habló antes. **Nota:** deberá usar al menos la versión `1.3` de sliplogin para que esto funcione.
4. la máscara de red asignada a la máquina que llama en notación decimal punteada, por ejemplo `255.255.255.0` para una máscara de red para una clase C.
5. las opciones del modo slip, que le permiten activar o desactivar la compresión y otras características de slip. Los valores permitidos aquí son `normal` y `compressed`.
6. un parámetro de *timeout* que especifica cuánto tiempo puede estar ociosa la línea (sin recibir datagramas) antes de que se desconecte la línea. Un valor negativo desactivará esta característica.
7. argumentos opcionales.

Nota: Para los campos 2 y 3 puede usar tanto el nombre de la máquina como su dirección IP en notación decimal. Si usted usa nombres entonces tendrán que ser resueltos, esto es, su máquina deberá ser capaz de localizar la dirección ip de esos nombres o en caso contrario el guión fallará cuando sea ejecutado. Puede probarlo intentando hacer telnet hacia esos nombres, y si obtiene el mensaje `Trying nnn.nnn.nnn...` entonces es que su máquina ha sido capaz de encontrar una dirección ip para ese nombre. Si obtiene el mensaje `Unknown host`, entonces no lo consiguió. En este caso, use una dirección ip en notación decimal o ponga a punto la configuración de su resolutor de nombres. (Ver sección 5.5 (Resolución de Nombres)).

Los modos slip más comunes son:

normal

para habilitar el SLIP normal sin compresión.

compressed

para habilitar la compresión de cabeceras de van Jacobsen (cSLIP)

Por supuesto, son mutuamente exclusivos. Puede usar uno o el otro, pero no ambos. Para obtener más información sobre las otras opciones disponibles, ve a las páginas de `man`.

Configuración del fichero `/etc/slip.login`. Después de que sliplogin haya examinado el `/etc/slip.hosts` y encontrado una entrada que concuerde, intentará ejecutar el fichero `/etc/slip.login` para configurar la interfaz SLIP con su dirección ip y su máscara.

El fichero de ejemplo `/etc/slip.login` proporcionado con el paquete `sliplogin` se parece a esto:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# fichero de login genérico para una línea SLIP. sliplogin
# lo invoca con los parámetros:
#     $1       $2       $3    $4, $5, $6 ...
# unidadSLIP veloctty   pid   los argumentos de la entrada en slip.host
```

```
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Podrá comprobar que este guión se limita a usar las órdenes `ifconfig` y `route` para configurar el dispositivo SLIP con su dirección ip, dirección ip remota y máscara de red y crea una ruta hasta la dirección remota a través del dispositivo SLIP. Exactamente lo mismo que haría si estuviera usando la orden `slattach`.

Advierta también el uso de *Proxy ARP* para asegurar que otras máquinas en la misma Ethernet que la máquina servidora sabrán cómo llegar a la máquina que llama. El campo `<hw_addr>` (dirección hardware) debería ser la dirección hardware de la tarjeta Ethernet de la máquina. Si su máquina servidora no está en una red Ethernet, entonces puede eliminar completamente esa línea.

Configuración del fichero `/etc/slip.logout`. Cuando la llamada se corta, querrá asegurarse de que se restaura el estado normal del dispositivo serie para que otras personas sean capaces más adelante de registrarse correctamente. Esto se lleva a cabo mediante el uso del fichero `/etc/slip.logout`. Es de formato bastante sencillo y se le llama con los mismos argumentos que al fichero `/etc/slip.login`.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Todo lo que hace es desactivar (down) la interfaz, lo que borrará la ruta que se creó anteriormente. También usa la orden `arp` para borrar cualquier proxy que se estableciese. De nuevo, no necesita la orden `arp` en el guión si su máquina servidora no tiene un puerto ethernet.

Configuración del fichero `/etc/slip.tty`. Si está usando asignación dinámica de direcciones ip (tiene configurada cualquier máquina con la palabra clave `DYNAMIC` en el fichero `/etc/slip.hosts`, entonces deberá configurar el fichero `etc/slip.tty` para que tenga una lista de las direcciones que se asignarán a qué puerto. Sólo necesita este fichero si desea que su servidor asigne de manera dinámica las direcciones a los usuarios.

El fichero es una tabla que lista los dispositivos `tty` que darán soporte a las conexiones SLIP y las direcciones ip que deberían asignarse a los usuarios que llaman a ese puerto.

Su formato es como sigue:

```
# slip.tty    tty -> correspondencias IP para SLIP dinámico
# formato: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Lo que dice esta tabla es que a la gente que llame al puerto `/dev/ttyS0` y que tengan la palabra `DYNAMIC` en su campo de dirección remota en el fichero `/etc/slip.hosts` les será asignada una dirección de `192.168.0.100`.

De esta manera sólo necesita asignar una dirección por puerto para todos los usuarios que no requieran una dirección dedicada para ellos. Esto le ayuda a mantener bajo mínimos los números de dirección para evitar su escasez.

7.6.2 Servidor Slip usando `dip`.

Déjeme empezar diciendo que parte de la información que sigue viene de las páginas de manual de `dip`, donde se documenta brevemente la manera de usar Linux como servidor SLIP. por favor, preste atención, puesto que lo siguiente está basado en el paquete `dip337o-uri.tgz` y probablemente no ses aplicable a otrs versiones de `dip`.

`dip` tiene un modo de operación de entrada, en el cual asigna automáticamente una entrada para el usuario que lo invoque y configura la línea serie como enlace SLIP de acuerdo con la información que encuentre en el fichero `/etc/dipshosts`. Este modo de operación de entrada se activa invocando a `dip` como `diplogin`. Es así por tanto la manera de usar `dip` como servidor SLIP, creando cuentas especiales donde se usa `diplogin` como *login shell*.

Lo primero que necesita hacer es un enlace simbólico como sigue:

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Después necesitará añadir entradas tanto al fichero `/etc/passwd` como al `/etc/diphosts`. Las entradas que necesita hacer tienen el formato que sigue:

Para configurar Linux como servidor SLIP con `dip`, se necesita crear algunas cuentas SLIP especiales para los usuarios, en las que se usa `dip` (en modo entrada) como *login shell*. Se sugiere la convención de comenzar los nombres de las cuentas SLIP con una `S` mayúscula, como por ejemplo `Sfredm`.

Una entrada en `/etc/passwd` para un usuario SLIP se parece a:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^         ^^        ^^   ^^   ^^   ^^   ^^
|          |         |    |    |    |    \__ diplogin como login shell
|          |         |    |    |    \_______ Directorio ‘home’
|          |         |    |    \____________ Nombre completo del usuario
|          |         |    \_________________ ID de grupo del usuario
|          |         \_____________________ ID del usuario
|          \_______________________________ Contraseña cifrada
\__________________________________________ Nombre de Login del Usuario SLIP
```

Después de que el usuario se registre, el programa `login`, si encuentra y autentica al usuario, ejecutará la orden `diplogin`. `dip`, cuando es invocado como `diplogin`, sabe que debe asumir automáticamente que va a ser usado como intérprete de órdenes de *login*. Cuando comienza como `diplogin`, la primera cosa que hace es usar la función `getuid()` para tomar el userid de quien fuera que lo invocó. Entonces busca en el fichero `/etc/diphosts` la primera entrada que se corresponda bien con el userid, bien con el dispositivo `tty` por el que entró la llamada y se configura apropiadamente. Por razones de sentido común, para que se pueda dar a un usuario una entrada en el fichero `diphosts`, o bien para que se le dé la configuración por defecto, es posible montar el servidor de tal manera que se pueda tener una mezcla de usuarios a los que se asigne la dirección de forma estática o dinámica.

`dip` añadirá automáticamente una entrada *Proxy-ARP* si se le invoca en modo entrada, por lo que no tendrá que preocuparse de hacerlo manualmente.

Configuración del fichero /etc/diphosts `/etc/diphosts` lo usa `dip` para buscar configuraciones ya establecidas para máquinas remotas. Estas máquinas remotas pueden ser usuarios que llaman a la máquina Linux, o pueden ser máquinas a las que usted llamas desde su máquina.

El formato general para `/etc/diphosts` es como sigue:

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

Los campos son:

1. `nombre de login`: el que devuelve `getpwuid(getuid())` o el nombre de la tty.
2. `sin uso`: para compatibilidad con `passwd`
3. `Dirección Remota`: dirección IP de la máquina que llama, puede ser tanto la numérica como el nombre.
4. `Dirección Local`: dirección IP de esta máquina, en número o por nombre.
5. `Máscara de red`: en notación decimal puntuada.
6. `Campo de comentario`: ponga aquí lo que quiera.
7. `protocolo`: Slip, cSLIP, etc.
8. `MTU`: número decimal.

Un ejemplo de entrada en `/etc/net/diphosts` para un usuario SLIP remoto podría ser:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

lo que especifica un enlace SLIP con dirección remota `145.71.34.1` y MTU de `296`, o:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

que especifica un enlace capaz de usar cSLIP con dirección remota `145.71.34.1` y MTU de `1006`.

Por tanto, todos los usuarios que desee que tengan asignado un acceso de llamada con IP estática deberían tener una entrada en `/etc/diphosts`. Si quiere que los usuarios que llaman a un puerto en particular y, que los detalles sean asignados dinámicamente, deberá tener entonces una entrada para el dispositivo `tty` y no una basada en el usuario. Debería configurar al menos una entrada para cada dispositivo `tty` que usen los usuarios para asegurar que hay disponible para ellos una configuración adecuada independientemente al módem al que llamen.

Cuando un usuario se registra recibirá las preguntas normales sobre su cuenta y contraseña en las que debería introducir su userid o contraseña de SLIP-login. Si se verifican, entonces el usuario no verá mensajes especiales y sólo tendrá que cambiar el modo SLIP en su extremo. El usuario debería ser capaz de conectar y quedar configurado con los parámetros relevantes del fichero `diphosts`.

7.6.3 Servidor SLIP usando el paquete dSLIP.

Matt Dillon, *dillon@apollo.west.oic.com* ha escrito un paquete que no sólo hace llamadas SLIP entrantes sino también llamadas salientes. El paquete de Matt es una combinación de pequeños programas y guiones que coordinan las conexiones por usted. Necesitará tener instalado `tcsh` ya que al menos uno de los guiones lo necesita. Matt proporciona una copia ejecutable de la utilidad `expect` ya que la necesita otro de los guiones. Además necesitará tener algo de experiencia con `expect` para hacer que este paquete funcione a su gusto, pero no deje que eso le desanime.

Matt ha escrito unas instrucciones de instalación muy buenas en el fichero `README`, por lo que no pienso repetirlas.

Puede obtener el paquete dSLIP de su sitio de origen:

ftp://apollo.west.oic.com/pub/linux/dillon_src/dSLIP203.tgz

o desde:

ftp://metalab.unc.edu/pub/Linux/system/Network/serial/dSLIP203.tgz

Léase el fichero `README` y cree las entradas en `/etc/passwd` y `/etc/group` antes de hacer `make install`.

8 Otras tecnologías de red

Las siguientes subsecciones son específicas a ciertas tecnologías de red. La información que contienen no son válidas necesariamente para cualquier otro tipo de tecnología. Los conceptos están ordenados alfabéticamente.

8.1 ARCNet

Los nombres de dispositivo ARCNet son `arc0e`, `arc1e`, `arc2e` etc. o `arc0s`, `arc1s`, `arc2s` etc. A la primera tarjeta detectada por el núcleo se le asigna `arc0e` o `arc0s` y el resto es asignado secuencialmente en el orden en que se detecte. La letra del final identifica si ha seleccionado el formato de paquete de encapsulación Ethernet o el formato de paquete especificado en el *RFC 1051*.

Opciones de Compilación del Núcleo:

```
Network device support  --->
    [*] Network device support
    <*> ARCnet support
    [ ]   Enable arc0e (ARCnet "Ether-Encap" packet format)
    [ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

Una vez que haya compilado el núcleo apropiadamente para admitir su tarjeta Ethernet, la configuración de la tarjeta es sencilla.

Normalmente se usa algo como:

```
# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

Lea, por favor, los ficheros `/usr/src/linux/Documentation/networking/arcnet.txt` y `/usr/src/linux/Documentation/networking/arcnet-hardware.txt` si desea obtener más información.

La implementación de ARCNet fue desarrollada por Avery Pennarun, *apenwarr@foxnet.net*.

8.2 Appletalk (AF_APPLETALK)

La implementación de Appletalk no utiliza nombres especiales para sus dispositivos ya que usa otros ya existentes.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    <*> Appletalk DDP
```

Trabajar con Appletalk permite a una máquina Linux interconectar con redes Apple. Una utilidad importante que se saca de esto es poder compartir recursos tales como impresoras y discos entre una máquina Linux y ordenadores Apple. Se necesita un programa adicional, que se llama *netatalk*. Wesley Craig *netatalk@umich.edu* representa a un equipo llamado el *Research Systems Unix Group*, de la Universidad de Michigan y han creado el paquete *netatalk*, que proporciona programas que implementan la pila del protocolo Appletalk y algunas utilidades. El paquete *netatalk* viene en su distribución de Linux, y si no, lo puede encontrar en su servidor de origen, en la Universidad de Michigan *ftp://terminator.rs.itd.umich.edu/unix/netatalk/*.

Para compilar e instalar el paquete haga algo como esto:

```
user% tar xvfz .../netatalk-1.4b2.tar.Z
user% make
root# make install
```

Puede que quiera editar el `Makefile` antes de ejecutar `make`. Principalmente para cambiar la variable `DESTDIR`, que define el lugar donde serán instalados los ficheros. El directorio por defecto `/usr/local/atalk` suele ser una buena elección.

8.2.1 Configuración del software Appletalk.

La primera cosa que tiene que hacer para que todo funcione es asegurarse de que están presentes las entradas apropiadas en el fichero `/etc/services`. La entradas que necesita son:

```
rtmp  1/ddp   # Routing Table Maintenance Protocol
nbp   2/ddp   # Name Binding Protocol
echo  4/ddp   # AppleTalk Echo Protocol
zip   6/ddp   # Zone Information Protocol
```

El siguiente paso es crear los ficheros de configuración de Appletalk en el directorio `/usr/local/atalk/etc` (o donde haya instalado el paquete).

El primer fichero a crear es el `/usr/local/atalk/etc/atalkd.conf`. En principio, este fichero sólo necesita una línea que da el nombre del dispositivo a través del cual se accede a la red donde están tus máquinas Apple.

```
eth0
```

El demonio Appletalk añadirá más detalles después de ejecutarse.

8.2.2 Exportación de un sistema de ficheros Linux vía Appletalk.

Existe la posibilidad de exportar sistemas de ficheros desde una máquina Linux a la red para que las máquinas Apple en la red puedan compartirlos.

Para hacerlo tiene que configurar el fichero `/usr/local/atalk/etc/AppleVolumes.system`. Hay otro fichero de configuración llamado `/usr/local/atalk/etc/AppleVolumes.default`, que tiene exactamente el mismo formato y describe qué sistemas de archivos recibirán los usuarios que conecten con privilegios de invitado.

Puede encontrar todos los detalles de configuración de estos ficheros y qué significa cada opción en la página de manual del `afpd`.

Un ejemplo sencillo podría parecerse a esto:

```
/tmp Scratch
/home/ftp/pub "Espacio Público"
```

Que exportaría su sistema de ficheros `/tmp` como el Volumen AppleShare `Scratch` y el directorio público de FTP como el Volumen AppleShare `Espacio Público`. Los nombres de volumen no son obligatorios, el demonio elegirá un nombre por defecto, pero no le va a morder si lo especifica.

8.2.3 Compartir la impresora Linux a través de Appletalk.

Puede compartir una impresora linux con máquinas Apple de manera bastante sencilla. Necesita ejecutar el programa `papd` que es el *Printer Access Protocol Daemon* de Appletalk. Cuando ejecute este programa, aceptará peticiones de las máquinas Apple y meterá la tarea de impresión en la cola del demonio de la impresora local para ser impreso. Debe editar el fichero `/usr/local/atalk/etc/papd.conf` para configurar el demonio. La sintaxis de este fichero es la misma que la del fichero `/etc/printcap`. El nombre que le dé a la definición se registrado con el protocolo de nombres de Appletalk, NBP.

Un ejemplo de configuración podría ser como éste:

```
TricWriter:\
   :pr=lp:op=cg:
```

Que pondría a disposición de la red Appletalk la impresora `TricWriter` y todos los trabajos aceptados serían impresos en la impresora linux `lp` (definida en el fichero `/etc/printcap`) usando `lpd`. La entrada `op=cg` dice que el usuario de Linux `cg` es el operador de la impresora.

8.2.4 Ejecución de AppleTalk.

Muy bien, ahora debería estar preparado para probar esta configuración básica. Hay un fichero `rc.atalk` proporcionado con el paquete `netatalk` que debería funcionar bien, por lo que todo lo que tiene que hacer es:

```
root# /usr/local/atalk/etc/rc.atalk
```

y todo debería comenzar y ejecutarse bien. No debería ver mensajes de error y el software enviará mensajes a la consola indicando cada etapa según comienza.

8.2.5 Comprobación de AppleTalk.

Para comprobar que el software está funcionando adecuadamente, vaya a una de las máquinas Apple, abra el menú Apple, seleccione el *Chooser*, pulse sobre *AppleShare*, y debería aparecer la máquina Linux.

8.2.6 Problemas con AppleTalk.

- Puede que necesite ejecutar el software Appletalk antes de configurar la red IP. Si tiene problemas ejecutando los programas Appletalk, o si después de ejecutarlo tiene problemas con la red IP, entonces intente ejecutar los programas Appletalk antes de ejecutar el fichero `rc` correspondiente.
- El `afpd` (*Apple Filing Protocol Daemon*) desordena mucho el disco duro. Bajo el punto de montaje crea un par de directorios llamados `.AppleDesktop` y `Network Trash Folder`. Además, por cada directorio al que acceda creará un `.AppleDouble` bajo él de manera que pueda almacenar *resource forks*, etc. Por tanto, piénselo dos veces antes de exportar `/`, o se pasará un rato borrando después.
- El programa `afpd` espera que las claves que vengan de los Mac estén sin cifrar. Esto podría ser un problema de seguridad; por tanto, sea cuidadoso cuando ejecute este demonio en una máquina conectada a Internet, o sufrirá las consecuencias si algún indeseable decide hacerle alguna maldad.
- Las herramientas de diagnóstico existentes como `netstat` e `ifconfig` no soportan Appletalk. La información «en bruto» está disponible en el directorio `/proc/net` si tuviese necesidad de ella.

8.2.7 Si necesitase más información...

Eche un vistazo a la página del *Linux Netatalk-Howto* de Anders Brownworth si quiere una descripción más detallada de cómo configurar Appletalk para Linux en *http://thehamptons.com/anders/netatalk*.

8.3 ATM

Werner Almesberger, *werner.almesberger@lrc.di.epfl.ch* está dirigiendo un proyecto para proporcionar una implementación del *Asynchronous Transfer Mode* en Linux. Puede obtener información actualizada sobre el estado del proyecto en: *http://lrcwww.epfl.ch/linux-atm*.

8.4 AX25 (AF_AX25)

Los nombres de los dispositivos AX.25 son `sl0`, `sl1`, etc. en los núcleos `2.0.*` y `ax0`, `ax1`, etc. a partir de los núcleos `2.1.*`.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

Los protocolos AX25, Netrom y Rose están cubiertos por el *AX25 Howto*. Estos protocolos son usados por los Operadores de Amateur Radio de todo el mundo para experimentar con packet radio.

La mayoría del trabajo de implementación de estos protocolos lo ha hecho Jonathon Naylor, *sn@cs.nott.ac.uk*.

8.5 DECNet

Actualmente se está trabajando en la implementación de DECNet. Debería aparecer en algún núcleo `2.1.*` tardío.

8.6 FDDI

Los nombres de dispositivo FDDI son `fddi0`, `fddi1`, `fddi2`, etc. A la primera tarjeta detectada por el núcleo se le asigna `fddi0` y al resto se le asigna secuencialmente en el orden en que son detectadas.

Lawrence V. Stefani, *larry_stefani@us.newbridge.com*, ha desarrollado un controlador para las tarjetas FDDI EISA y PCI de *Digital Equipment Corporation*.

Opciones de Compilación del Núcleo:

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

Cuando tenga el núcleo compilado para trabajar con el controlador FDDI e instalado, la configuración de la interfaz FDDI es casi idéntica al de una interfaz Ethernet. Simplemente ha de especificar la interfaz FDDI apropiada en las órdenes `ifconfig` y `route`.

8.7 Retransmisión de Tramas (`Frame Relay`)

Los nombres de dispositivo Frame Relay son `dlci00`, `dlci01`, etc para los dispositivos de encapsulación DLCI y `sdla0`, `sdla1`, etc para los FRAD.

El Frame Relay (Retransmisión de tramas) es una tecnología de red diseñada para ajustarse al tráfico de comunicación de datos que es de naturaleza «explosiva» o intermitente. La conexión a una red Frame Relay se realiza usando un *Frame Relay Access Device* (FRAD). El Linux Frame Relay implementa IP sobre Frame Relay según se describe en el *RFC-1490*.

Opciones de Compilación del Núcleo:

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

Mike McLagan, *mike.mclagan@linux.org*, desarrolló el soporte de Frame Relay y las herramientas de configuración.

Actualmente los únicos FRAD soportados son los `S502A`, `S502E` y `S508` de *Sangoma Technologies* *http://www.sangoma.com* Para configurar los dispositivos FRAD y DLCI después de haber recompilado el núcleo necesitará las herramientas de configuración Frame Relay. Están disponibles en *ftp://ftp.invlogic.com/pub/linux/fr/frad-0.15.tgz*. Compilar e instalar las herramientas es algo muy sencillo, pero la carencia de un fichero `Makefile` para todo lo convierte en un proceso básicamente manual:

```
user% tar xvfz .../frad-0.15.tgz
user% cd frad-0.15
user% for i in common dlci frad; make -C $i clean; make -C $i; done
root# mkdir /etc/frad
root# install -m 644 -o root -g root bin/*.sfm /etc/frad
root# install -m 700 -o root -g root frad/fradcfg /sbin
root# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Tenga en cuenta que estas órdenes usan sintaxis de `sh`. Si utiliza un intérprete de órdenes tipo `csh` (como `tcsh`), el bucle `for` será diferente.

Después de instalar las herramientas necesitará crear un fichero `/etc/frad/router.conf`. Puede usar esta plantilla, que es una versión modificada de uno de los ficheros de ejemplo:

```
# /etc/frad/router.conf
# Esta es una plantilla de configuración para retransmisión de tramas.
# Se incluyen todas las etiquetas. Los valores por defecto están basados
# en el código proporcionado con los controladores DOS para la tarjeta
# Sangoma S502A.
#
# Un '#' en cualquier parte de una línea constituye un comentario
# Los espacios en blanco son ignorados (puede indentar con tabuladores
# también)
# Las entradas [] y claves desconocidas son ignoradas
#

[Devices]
Count=1                 # Número de dispositivos a configurar
Dev_1=sdla0             # el nombre del dispositivo
#Dev_2=sdla1            # el nombre del dispositivo

# Lo especificado aquí es aplicado a todos los dispositivos y puede ser
# cambiado para cada tarjeta en particular.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
#
# MTU=1500              # Máxima longitud del IFrame, por defecto 4096
# T391=10               # valor T391    5 - 30, por defecto 10
# T392=15               # valor T392    5 - 30, por defecto 15
# N391=6                # valor N391    1 - 255, por defecto 6
# N392=3                # valor N392    1 - 10, por defecto 3
# N393=4                # valor N393    1 - 10, por defecto 4

# Lo especificado aquí da los valores por defecto para todas las tarjetas
# CIRfwd=16             # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511

#
#
# Configuración específica para el dispositivo
#
#

#
# El primer dispositivo es un Sangoma S502E
#
```

```
[sdla0]
Type=Sangoma            # Tipo del dispositivo a configurar, actualmente
                        # sólo se reconoce SANGOMA
#
# Estas claves son específicas al tipo "Sangoma"
#
# El tipo de tarjeta Sangoma - S502A, S502E, S508
Board=S502E
#
# El nombre del firmware de prueba para la tarjeta Sangoma
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# El nombre del firmware FR
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360                # Puerto de esta tarjeta
Mem=C8                  # Dirección de la memoria, A0-EE, depende
IRQ=5                   # Número de la IRQ, no especificar para la S502A
DLCIs=1                 # Número de DLCI asociados al dispositivo
DLCI_1=16               # Número del DLCI nº 1, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Lo especificado aquí se aplica a este dispositivo nada más y
# prevalece sobre los valores por defecto
#
# Access=CPE            # CPE o NODE, por defecto CPE
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal        # External o Internal, por defecto Internal
# Baud=128              # Tasa en baudios del CSU/DSU asociado
# MTU=2048              # Longitud máxima del IFrame, por defecto 4096
# T391=10               # valor T391    5 - 30, por defecto 10
# T392=15               # valor T392    5 - 30, por defecto 15
# N391=6                # valor N391    1 - 255, por defecto 6
# N392=3                # valor N392    1 - 10, por defecto 3
# N393=4                # valor N393    1 - 10, por defecto 4

#
# El segundo dispositivo es otra tarjeta
#
# [sdla1]
# Type=FancyCard        # Type of the device to configure.
# Board=                # Type of Sangoma board
# Key=Value             # values specific to this type of device

#
# DLCI Default configuration parameters
# These may be overridden in the DLCI specific configurations
#
CIRfwd=64               # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
```

```
# Bc_bak=16            # Bc backward   1 - 512
# Be_bak=0             # Be backward   0 - 511

#
# DLCI Configuration
# These are all optional. The naming convention is
# [DLCI_D<devicenum>_<DLCI_Num>]
#

[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=16
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0
```

Cuando haya terminado el fichero `/etc/frad/router.conf`, el único paso que queda es configurar el dispositivo en sí. Esto es sólo un poco más complejo que la configuración de un dispositivo de red normal. Debe recordar activar el dispositivo FRAD antes que los dispositivos de encapsulación DLCI. Es mejor que ponga estas órdenes en un guión, ya que son muchos:

```
#!/bin/sh
# Configurar los parámetros del frad y los DLCI
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Activar el dispositivo FRAD
ifconfig sdla0 up
#
# Configurar los dispositivos de encapsulación DLCI
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add -net 192.168.10.0 netmask 255.255.255.0 dlci00
#
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add -net 192.168.11.0 netmask 255.255.255.0 dlci00
#
```

```
route add default dev dlci00
#
```

8.8 IPX (AF_IPX)

El protocolo IPX se usa comúnmente en entornos de redes de área local Novell Netware(tm). Linux incluye una implementación de este protocolo y puede ser configurado para actuar como punto final en una red, o como encaminador de IPX.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    [*] The IPX protocol
    [ ] Full internal IPX network
```

Los protocolos IPX y el NCPFS están cubiertos en gran profundidad en el *IPX Howto*.

8.9 NetRom (AF_NETROM)

Los nombres de los dispositivos NetRom son nr0, nr1, etc.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Los protocolos AX25, Netrom y Rose están cubiertos en el *AX25 Howto*. Estos protocolos los usan Operadores de Amateur Radio de todo el mundo en la experimentación de *packet radio*.

La mayoría del trabajo de la implementación de estos protocolos lo ha Jonathon Naylor, *jsn@cs.nott.ac.uk*.

8.10 Protocolo Rose (AF_ROSE)

Los nombres de los dispositivos Rose son rs0, rs1, etc, en los núcleos 2.1.*. Rose está disponible a partir de los núcleos 2.1.*.

Opciones de Compilación del Núcleo:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    <*> Amateur Radio X.25 PLP (Rose)
```

Los protocolos AX25, Netrom y Rose están cubiertos por el *AX25 Howto*. Estos protocolos los usan Operadores de Amateur Radio de todo el mundo en la experimentación de *packet radio*.

La mayoría del trabajo de la implementación de estos protocolos lo ha hecho Jonathon Naylor, *jsn@cs.nott.ac.uk*.

8.11 Soporte SAMBA - NetBEUI, NetBios.

SAMBA es una implementación del protocolo *Session Management Block*. Samba permite que los sistemas de Microsoft (y otros) monten y usen sus discos e impresoras.

SAMBA y su configuración vienen cubiertos en detalle en el *http://www.insflug.org/documentos/Samba-Como/*.

8.12 Soporte de STRIP (*Starmode Radio IP*)

Los nombres de los dispositivos STRIP son `st0`, `st1`, etc.

Opciones de Compilación del Núcleo:

```
Network device support  --->
    [*] Network device support
    ....
    [*] Radio network interfaces
    < > STRIP (Metricom starmode radio IP)
```

STRIP es un protocolo diseñado específicamente para un rango de radio-módems Metricon para un proyecto de investigación conducido por la Universidad de Stanford llamado MosquitoNet Project *http://mosquitonet.Stanford.EDU/mosquitonet.html*. Es muy interesante leerse esto, incluso si no está interesado directamente en el proyecto.

Las radios Metricon se conectan a un puerto serie, emplean tecnología de espectro amplio y suelen ser capaces de alcanzar los 100kbps. Hay información disponible acerca de las radios Metricon en el servidor Web de Metricon *http://www.metricom.com*.

En estos momentos las herramientas y utilidades estándar de red no implementan el controlador STRIP, por lo que deberá obtener algunas herramientas preparadas desde el servidor WWW de MosquitoNet. Los detalles sobre los programas que necesitará están disponibles en la página de MosquitoNet sobre STRIP *http://mosquitonet.Stanford.EDU/strip.html*.

En resumen de la configuración, use un programa `slattach` modificado para establecer STRIP como la disciplina de línea de un dispositivo serie tty y entonces configurar el dispositivo `st[0-9]` resultante igual que haría con uno Ethernet con una excepción importante: por razones técnicas, STRIP no soporta el protocolo ARP, por lo que deberá configurar manualmente las entradas ARP de cada una de las máquinas en la subred. Esto no debería ser demasiado oneroso.

8.13 Anillo con testigo (*Token Ring*)

Los nombres de los dispositivos de anillo con testigo son `tr0`, `tr1`, etc. El anillo con testigo es un protocolo de red LAN estándar de IBM que evita colisiones proporcionando un mecanismo que permite que sólo una de las estaciones en la red tenga en un momento determinado derecho a transmitir. Una estación mantiene un «testigo» (*token*) durante un tiempo determinado y ésa es la única con permiso para transmitir. Cuando ha transmitido los datos le pasa el testigo a la siguiente estación. El testigo traza un bucle entre todas las estaciones activas, de ahí el nombre de «Anillo con Testigo».

Opciones de Compilación del Núcleo:

```
Network device support  --->
    [*] Network device support
    ....
    [*] Token Ring driver support
    < > IBM Tropic chipset based adaptor support
```

La configuración de un anillo con testigo es idéntica a la de una Ethernet con la excepción del nombre de los dispositivos a configurar.

8.14 X.25

El X.25 es un protocolo de circuitos basados en conmutación de paquetes definido por el `C.C.I.T.T.` (una organización de estándares reconocida por compañías de Telecomunicaciones en la mayor parte del mundo). En estos momentos está en funcionamiento una implementación de X.25 y LAPB y los núcleos `2.1.*` más recientes incluyen el trabajo que está en progreso.

Jonathon Naylor *jsn@cs.nott.ac.uk* está liderando el desarrollo y se ha establecido una lista de correo para discutir materias relativas al X.25 en Linux. Para suscribirse, envíe un mensaje a: *majordomo@vger.rutgers.edu* con el texto `subscribe linux-x25` en el cuerpo del mensaje.

Se pueden obtener las primeras versiones de las herramientas de configuración desde el sitio de FTP de Jonathon en *ftp://ftp.cs.nott.ac.uk/jsn*.

8.15 Tarjeta WaveLan

Los nombres de los dispositivos Wavelan son `eth0`, `eth1`, etc.

Opciones de Compilación del Núcleo:

```
Network device support  --->
    [*] Network device support
    ....
    [*] Radio network interfaces
    ....
    <*> WaveLAN support
```

La tarjeta WaveLAN es una tarjeta de red LAN inalámbrica de amplio espectro. La tarjeta se parece bastante en la práctica a una tarjeta Ethernet y se configura de la misma manera.

Puede obtener información acerca de la tarjeta Wavelan de Wavelan.com *http://www.wavelan.com*.

9 Cables y Cableado

Aquellos de ustedes que se manejen con un soldador puede que quieran construir sus propios cables para interconectar dos máquinas Linux. Los siguientes diagramas de cableado deberían serles de ayuda.

9.1 Cable serie Módem NULO (NULL Modem)

No todos los cables Módem NULO son iguales. Muchos cables módem nulo hacen poco más que un pequeño truco para que el ordenador crea que están presentes todas las señales apropiadas, cruzando los cables de transmisión y recepción. Esto vale, pero significa que deberá usar programas de control de flujo software (XON/XOFF), que es menos eficiente que el control de flujo hardware. El siguiente cable proporciona la mejor señalización posible entre las máquinas y permite usar control de flujo por hardware (RTS/CTS).

```
Nombre Patilla Patilla                           Patilla
 Datos Tx       2  -----------------------------  3
 Datos Rx       3  -----------------------------  2
 RTS            4  -----------------------------  5
 CTS            5  -----------------------------  4
 Tierra         7  -----------------------------  7
 DTR            20 -\---------------------------  8
```

```
DSR           6  -/
RLSD/DCD      8  ---------------------------/-  20
                                            \-  6
```

9.2 Cable de puerto paralelo (cable PLIP)

Si pretende usar el protocolo PLIP entre dos máquinas, entonces este cable funcionará independientemente del tipo de puertos paralelos que tenga instalados.

```
Nombre Patilla  Patilla        Patilla
 STROBE         1*
 D0->ERROR      2  ----------- 15
 D1->SLCT       3  ----------- 13
 D2->PAPOUT     4  ----------- 12
 D3->ACK        5  ----------- 10
 D4->BUSY       6  ----------- 11
 D5             7*
 D6             8*
 D7             9*
 ACK->D3        10 ----------- 5
 BUSY->D4       11 ----------- 6
 PAPOUT->D2     12 ----------- 4
 SLCT->D1       13 ----------- 3
 FEED           14*
 ERROR->D0      15 ----------- 2
 INIT           16*
 SLCTIN         17*
 GROUND         25 ----------- 25
```

Notas:

- No conecte las patillas marcadas con un asterisco, *.
- Las tomas de tierra extra son 18,19,20,21,22,23 y 24.
- Si el cable que está usando tiene apantallamiento metálico, debería estar conectado a la carcasa DB-25 en **sólo uno** de los extremos.

Cuidado: Un cable PLIP mal hecho puede destruir la tarjeta controladora. Sea muy cuidadoso y examine dos veces cada conexión para asegurarse de que no va a hacer más trabajo ni a llevarse más infartos de lo necesario.

Aunque puede que sea capaz de tener cables PLIP para grandes distancias, debería evitarlo. Las especificaciones del cable permiten una longitud de alrededor de 1 metro. Por favor, tenga mucho cuidado cuando tienda cables PLIP largos, ya que las fuentes de campos electromagnéticos fuertes, como los rayos, cables de corriente y emisoras de radio pueden interferir en los controladores, y a veces dañarlos. Si realmente quiere conectar dos ordenadores a larga distancia, debería intentar obtener un par de tarjetas Ethernet para thin-net (red de cable fino) y tender cable coaxial.

9.3 Cableado Ethernet 10base2 (coaxial fino)

10base2 es un estándar de cableado Ethernet que especifica el uso de cables coaxiales de 52 ohmios con un diámetro de alrededor de 5 milímetros. Hay un par de reglas importantes a recordar cuando conectemos máquinas con cableado 10base2. La primera es que debe usar terminadores en *ambos extremos* del cable. Un terminador es una resistencia de 52 ohmios que ayuda a asegurar que la señal es absorbida y no reflejada cuando alcanza el final del cable. Sin un terminador a cada extremo del cable, podría pasar que la Ethernet sea ineficiente o que no funcione. Normalmente debería usar conectores en «T» para interconectar las máquinas por lo que terminará con algo que se parezca a:

```
|==========T=============T=============T==========T==========|
           |             |             |          |
           |             |             |          |
         -----         -----         -----      -----
         |   |         |   |         |   |      |   |
         -----         -----         -----      -----
```

Donde el | a cada extremo representa un terminador, el `======` representa cables coaxiales con conectores BNC a cada extremo y las `T` representan conectores «en T». Debería hacer que la longitud del cable entre la `T` y el PC lo más corto posible ya que, idealmente, la «T» debería estar directamente enchufada a la tarjeta Ethernet.

9.4 Cable Ethernet de Par Trenzado

Si tiene sólo dos tarjetas Ethernet de par trenzado y desea conectarlas, entonces no necesita un concentrador. Puede cablear las dos tarjetas directamente una a otra. Hay un diagrama que muestra cómo hacerlo incluido en el *Ethernet Howto*

10 Glosario de Términos usados en este documento

La siguiente lista contiene algunos de los términos más importantes usados en este documento.

ARP

Es un acrónimo para del *Protocolo de Resolución de Direcciones* (Address Resolution Protocol) y es la manera en que asocia una máquina de red una dirección IP con una dirección hardware.

ATM

Es un acrónimo de *Modo Asíncrono de Transferencia* (Asynchronous Transfer Mode). Una red ATM empaqueta los datos en bloques de tamaño estándar que puede transportar eficientemente de un punto a otro. ATM es una tecnología de red basada en circuitos de paquetes conmutados.

cliente

Suele ser la parte de un programa que se encuentra en el lado del usuario. Hay ciertas excepciones, por ejemplo, en el sistema de ventanas X11 es el servidor el que está con el usuario, y el cliente puede estar ejecutándose en una máquina remota. El cliente es el programa o terminal de un sistema que está recibiendo el servicio proporcionado por el servidor. En el caso de sistema *de igual a igual* (peer to peer), como `slip` o `ppp`, se dice que el cliente es el extremo que inicia la conexión y el extremo remoto, que ha sido llamado, es el servidor.

datagrama

Un datagrama es un paquete discreto de datos y cabeceras que contiene direcciones, que es la unidad básica de transmisión a través de una red IP. Puede que lo haya oído llamar «paquete».

Dirección hardware

Es un número que identifica de forma unívoca una máquina en una red física en la capa de acceso al medio. Algunos ejemplos son las *Direcciones Ethernet* y las *Direcciones AX.25*.

Dirección IP

Es un número que identifica unívocamente una máquina TCP/IP en la red. La dirección es de 4 bytes de longitud y normalmente se la representa en *notación decimal puntuada*, donde cada byte es representado en decimal con puntos «.» entre ellos.

DLCI

DLCI viene de *Data Link Connection Identifier* y se usa para identificar una conexión virtual punto a punto en particular a través de una red de Retransmisión de Tramas. Los DLCI los suele asignar el proveedor de red de Retransmisión de Tramas.

ISP

Véase PSI.

MSS

El Tamaño Máximo de Segmento (*Maximum Segment Size*) es la mayor cantidad de datos que pueden ser transmitidos a la vez. Si quiere prevenir la fragmentación, el MSS debería ser igual al MTU de la cabecera IP.

MTU

La Máxima Unidad de Transmisión (*Maximum Transmission Unit*) es un parámetro que determina el mayor datagrama que puede ser transmitido por una interfaz IP sin necesidad de dividirlo en unidades más pequeñas. La MTU debería ser más grande que el mayor datagrama que desee transmitir sin fragmentar. Fíjese que esto sólo previene la fragmentación de manera local, algunos otros enlaces por el camino pueden tener una MTU más pequeña y el datagrama será fragmentado allí. Los valores normales son 1500 bytes para una interfaz Ethernet, o 576 bytes para una interfaz SLIP.

PSI

Es un acrónimo de Proveedor de Servicios de Internet (ISP - Internet Service Provider). Son organizaciones y compañías que proporcionan conectividad con Internet.

RDSI

Es un acrónimo de *Red Digital de Servicios Integrados* (ISDN en inglés). La RDSI proporciona medios estandarizados por los cuales las compañías de Telecomunicaciones pueden llevar información tanto de voz como de datos según las premisas de los abonados. Técnicamente la RDSI es una red de circuitos de paquetes conmutados.

Retransmisión de tramas

La retransmisión de tramas (o Frame Relay) es una tecnología de red idealmente preparada para transportar tráfico que es de naturaleza esporádica o a ráfagas. Los costes de red se reducen compartiendo varios abonados a la retransmisión de tramas la misma capacidad de red y dejándoles a ellos el deseo de usar la red en momentos ligeramente diferentes.

ruta

La *ruta* es el camino que siguen los datagramas a través de la red para alcanzar su destino.

servidor

Normalmente es la parte del programa o terminal de un sistema remoto con respecto al usuario. El servidor proporciona algún servicio a uno o varios clientes. Los ejemplos de servidores incluyen *ftp*, *Network File System* o *Domain Name Server*. En el caso de sistemas de *igual a igual* (peer to peer) tales como `slip` o `ppp`. Se dice que es el servidor el extremo del enlace que recibe la llamada y el otro extremo es el cliente.

ventana

La *ventana* es la mayor cantidad de datos que el extremo receptor puede aceptar en un momento dado.

11 ¿Linux para un PSI?

Si está interesado en usar Linux para propósitos de PSI entonces le recomiendo que eche un vistazo a la página principal de Linux ISP *http://www.anime.net/linuxisp* si quiere obtener una buena lista de enlaces a la información que pudiera necesitar.

12 Reconocimientos

Me gustaría dar las gracias a las siguientes personas por su contribución a este documento (no están por orden): Terry Dawson, Axel Boldt, Arnt Gulbrandsen, Gary Allpike, Cees de Groot, Alan Cox, Jonathon Naylor, Claes Ensson, Ron Nessim, John Minack, Jean-Pierre Cocatrix, Erez Strauss.

El traductor quisiera también reconocer el trabajo de Fernando Tricas, por su inestimable ayuda en el proceso de revisión del documento.

El mayor de los reconocimientos a la gran labor de Francisco Montilla *pacopepe@insflug.org* y el grupo Insflug *http://www.insflug.org* para conseguir aportar documentación de calidad al movimiento LiNUX. ¡Animo!

13 Copyright.

14 Anexo: El INSFLUG

El *INSFLUG* forma parte del grupo internacional *Linux Documentation Project*, encargándose de las traducciones al castellano de los Howtos, así como de la producción de documentos originales en aquellos casos en los que no existe análogo en inglés, centrándose, preferentemente, en documentos breves, como los *COMOs* y *PUFs* (**P**reguntas de **U**so **F**recuente, las *FAQs*. `:)`), etc.

Diríjase a la sede del Insflug para más información al respecto.

En ella encontrará siempre las **últimas** versiones de las traducciones «oficiales»: *www.insflug.org*. Asegúrese de comprobar cuál es la última versión disponible en el Insflug antes de bajar un documento de un servidor réplica.

Además, cuenta con un sistema interactivo de gestión de fe de erratas y sugerencias en línea, motor de búsqueda específico, y más servicios en los que estamos trabajando incesantemente.

Ponga su granito de arena; si detecta una errata o incorrección en este documento, por favor, diríjase a *http://www.insflug.org/documentos/Redes-En-Linux-Como/* y aporte su sugerencia o errata en línea.

En *http://www.insflug.org/insflug/creditos.php3* cuenta con una detallada relación de las personas que hacen posible tanto esto como las traducciones.

¡Diríjase a *http://www.insflug.org/colaboracion/index.php3* si desea unirse a nosotros!.

«Cartel» Insflug, *cartel@insflug.org*.