

I

Los protocolos

- 2 Protocolos de la capa de enlace.
- 3 Protocolos de la capa de red.
- 4 Protocolos de la capa de transporte.
- 5 Protocolos de la capa de aplicación.

2

Protocolos de la capa de enlace

Los protocolos de la capa de enlace proporcionan los medios para que las computadoras se comuniquen unas con otras cuando haya un enlace físico común. Ese enlace podría aportarlo un Ethernet normal, una línea de POTS (*Plain Old Telephone Service*, Antiguo servicio telefónico sencillo), un anillo de FDDI (*Fiber Distributed Data Interface*, Interfaz de datos distribuidos por fibra) o incluso un palo de palomas mensajeras (véase el RFC 1149 para más información ¹).

Todos los protocolos de la capa superior dependen de la capa de enlace para la entrega real de los datos. Como existen los protocolos de la capa superior, el tráfico puede cruzar un *internetwork* con muchos protocolos de capa de enlace diferentes. De hecho, la mayoría de los *internetworks* encajan con esta descripción. Por ejemplo, dos LAN de oficina con una conexión PPP dedicada se parecerían al diagrama que se muestra en la Figura 2.1.

En las siguientes secciones, estudiaremos dos protocolos de la capa de enlace, PPP y Ethernet, y de un protocolo de la capa de enlace *encapsulado*, PPPoE. También hablaremos sobre el dispositivo de bucle de retorno local que ofrecen muchas pilas IP, del protocolo ARP y del método utilizado para determinar los tamaños de paquete que deberían utilizarse cuando se atravesaran *internetworks* como la que se ve en la Figura 2.1.

PPP

El RFC 1661 define el PPP (*Point-to-Point Protocol*, Protocolo punto a punto), que “proporciona un método estándar para transportar datagramas

¹ El Día de los inocentes en Estados Unidos y Gran Bretaña es el día de Internet. Cada año, el uno de abril, se publican una serie de RFC poco serios. El RFC 1149 es uno de ellos.

multiprotocolo por enlaces de punto a punto". PPP se utiliza para conectar muchas computadoras caseras a Internet y para ofrecer una conexión neutral en cuanto a distribuidor entre encaminadores.

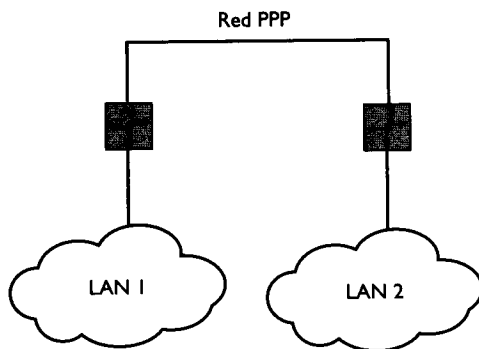


Figura 2.1. Dos LAN con una conexión PPP:

PPP ha suplantado al SLIP (*Serial Line IP*, IP de línea en serie), un protocolo anterior descrito en el RFC 1055. SLIP y SLIP comprimido (*Compressed SLIP*) carecen de algunas de las características que aporta PPP:

- SLIP no ofrece ningún mecanismo para transferir ningún protocolo aparte de IP.
- SLIP no tiene un mecanismo de comprobación de errores, dejando el tráfico abierto a la corrupción debido a los problemas de la línea telefónica.
- No hay ningún método para anunciar direcciones IP en SLIP. Debemos conocer la dirección IP de cada lado de la conexión antes de comenzar².

Estudio sobre PPP

Los marcos PPP tienen un encabezamiento de 5 bytes, hasta 1500 bytes de datos y después un tráiler de 3 bytes. El encabezamiento contiene un indicador de inicio de 1 byte (siempre 0x7e)³, un campo de dirección de 1 byte (siempre 0xff), un campo de control de 1 byte (siempre 0x03) y un campo de protocolo de 2 bytes (IP es 0x0021).

Todos los valores del campo de protocolo tienen que ser números impares y pueden indicar datos del LCP (*Link Control Protocol*, Protocolo de con-

² La gran mayoría de las conexiones PPP modernas permiten al cliente PPP "descubrir" su dirección IP consultando al servidor PPP. Lo que hace que los servidores PPP puedan ofrecer conexión por marcación (*dial-up*) a los usuarios finales sin obligarles a configurar su dirección IP cada vez que se conecten.

³ Siempre que veamos un número precedido del prefijo 0x, es un número hexadecimal. La mayor parte de los números con los que trataremos en las descodificaciones TCP/IP son hexadecimales.

tol de enlace), datos del NCP (*Network Control Protocol*, Protocolo de control de red) o datagramas encapsulados de un protocolo de nivel superior.

LCP proporciona un canal para la comprobación, configuración y establecimiento de enlace. Éste es el mecanismo utilizado para negociar las direcciones IP y otras opciones.

Los NCP son específicos de los protocolos de capa de red. Estos protocolos permiten que se negocien opciones adicionales.

PPP descodificado

La Figura 2.2 nos muestra un marco PPP típico³. El indicador de inicio no se muestra, porque en realidad es sólo un marcador para indicar tráfico significativo en lugar de ruido de línea. El primer byte (0xff) es el campo de Dirección. El segundo byte (0x03) representa al campo de Control. Los siguientes 2 bytes (0xc021) representan una petición eco LCP. El resto del paquete es carga útil específica de LCP.

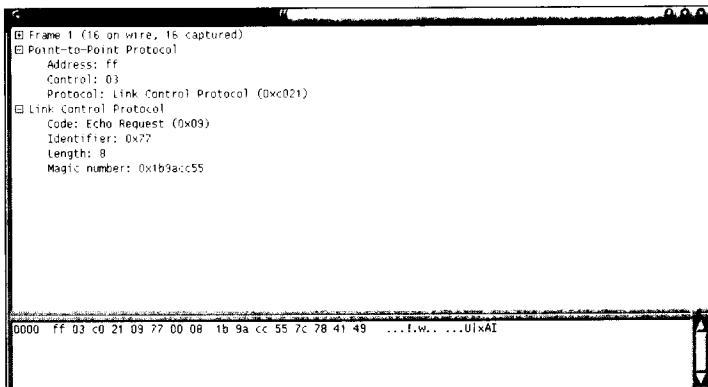


Figura 2.2. Un marco PPP

Ethernet

Aunque normalmente hablamos de Ethernet como de un único protocolo, en realidad dos protocolos distintos implementan Ethernet en la capa de enlace. El más utilizado es Ethernet II, descrito en el RFC 894. Menos utilizado es el Ethernet estilo 802.3 IEEE, definido por el IETF en el RFC 1042. Son necesarios los *hosts* de Internet para comprender Ethernet II y pueden implementar 802.3. Si ofrecen 802.3, debe ofrecerse Ethernet II, y debería ser el

³ Véase la sección “Ethereal” del capítulo 10, “Herramientas de revisión”, para más información sobre cómo leer estas pantallas Ethereal.

elemento predeterminado⁵. Explicaremos Ethernet en términos de Ethernet II y señalaremos las áreas en las que 802.3 difiere.

La especificación de Ethernet original fue publicada conjuntamente por DEC, Intel y Xerox en 1982. Utiliza un *bus* compartido (lógico o físico) para la comunicación y controla el acceso mediante acceso múltiple con sentido portador con detección de colisión (CSMA/CD). Esto significa que todos los *hosts* de una LAN comparten un dispositivo de comunicaciones común (*acceso múltiple*). Cuando un dispositivo tiene tráfico que enviar, escucha a la espera de una apertura (*sentido portador*) y después intenta enviar su mensaje. Si más de un *host* intenta enviar datos a la vez, se produce una *colisión*, y ambos *hosts* esperan un periodo corto de tiempo y lo vuelven a intentar. El periodo de espera es aleatorio para evitar que los equipos vuelvan a colisionar. Esto es muy parecido al mecanismo que utilizamos cuando intentamos mantener una conversación en una habitación llena de gente. Esperamos a que haya una pausa y entonces comenzamos a hablar. Si otra persona habla al mismo tiempo, se produce una pausa corta y alguien empezará a hablar mientras que los demás miembros de la conversación esperan.

Estudio sobre Ethernet

Los paquetes de Ethernet están compuestos de un encabezamiento de 14 bytes, un campo de datos de 46 a 1500 bytes y una CRC (*Cyclic Redundancy Check*, Comprobación de redundancia cíclica) de 4 bytes. Los primeros 6 bytes del encabezamiento forman la *dirección Ethernet* (también llamada dirección MAC o dirección de máquina) del sistema de destino. Los segundos 6 bytes representan la dirección MAC del sistema de origen. Los últimos 2 bytes del encabezamiento son el campo de Tipo⁶. La Tabla 2.1 muestra algunos de los códigos de tipo más utilizados.

Tabla 2.1. Códigos de tipo de Ethernet de interés.

Código de tipo	Tipo de carga útil
0x0800	IPv4
0x86DD	IPv6
0x0806	ARP
0x8035	RARP
0x809B	AppleTalk

⁵ Los RFC ofrecen una definición bastante estricta de lo que los *hosts deberían* hacer y de lo que *deben* hacer. Intentamos seguir la utilización que hacen de estos términos. Si un *host* “debe” hacer algo, no se considera que tenga una implementación de acuerdo con los estándares si falla al hacerlo. Si un *host* “debería” hacer algo, aún puede estar de acuerdo a los estándares incluso sin ese comportamiento. Estos términos se definen en el RFC 2119.

⁶ En Ethernet 802.3, los bytes 13 y 14 son un campo de Longitud. Esto exige que los paquetes 802.3 lleven información de código de tipo en otro encabezamiento dentro de su campo de Datos.

Como hay una longitud mínima de 46 bytes para el campo de datos de Ethernet, puede que los datos necesiten *relleno*, bytes adicionales que suban la cuenta de bytes del campo de datos hasta el tamaño mínimo.

Los paquetes de menos de 64 bytes (14 bytes de encabezamiento, 46 bytes de datos y un tráiler de 4 bytes) se llaman alfeñiques y se ignoran⁷. Los paquetes de más de 1518 bytes (14 bytes de encabezamiento, 1500 bytes de datos y 4 bytes de CRC) se llaman gigantes y también se ignoran.

Las direcciones MAC están compuestas de dos secciones. Los primeros 3 bytes son un campo de identificación de distribuidor. Los segundos 3 bytes representan un ID único para cada tarjeta⁸. Hay posibilidades de direcciones adicionales en la dirección de destino. Activar el bit de valor más bajo en el primer byte representa una dirección de multiconversión, un mensaje enviado a múltiples *hosts* de la red. Activar todos los bits de la dirección de destino representa una forma especial de multiconversión llamada *difusión*.

Es importante recordar que sólo porque un paquete esté dirigido a un *host* específico (o una dirección de multiconversión escuchada por múltiples *hosts*), no es privado. Todos los *hosts* de un segmento de Ethernet reciben todos los paquetes enviados en ese segmento. (Esto es lo que permite trabajar a los analizadores de red.) Normalmente, cada *host* procesa sólo aquellos paquetes que vienen dirigidos a él, incluidas multiconversiones y difusiones, e ignora los que están dirigidos a otros *hosts*. La dirección de destino se coloca al principio para permitir que los *hosts* que siguen este comportamiento puedan manipular el tráfico Ethernet de manera más eficiente.

Esto indica claramente una gran diferencia entre PPP y Ethernet. PPP es un protocolo punto a punto; el tráfico sólo lo pueden ver los dos puntos finales del circuito (y cualquier equipo especial introducido en éste). Ethernet es un medio de difusión; todo elemento de la LAN oirá cada porción de tráfico enviada⁹.

Ethernet descodificado

El diagrama de la Figura 2.3 muestra un marco de Ethernet II. En este marco, la dirección de destino está establecida en ff:ff:ff:ff:ff:ff (la dirección de difusión), la dirección de origen es 00:e0:98:7c:95:21 y el campo de Tipo es 0x0806 (ARP).

⁷ Toman ancho de banda y presentan una condición de problema que debería corregirse.

⁸ Aunque, en teoría, estos 3 bytes son únicos, existen algunas circunstancias que rompen esta regla. Algunas tarjetas Ethernet y algunos sistemas operativos (incluido Linux) nos permitirán establecer una dirección MAC diferente; la asignación de una dirección duplicada puede provocarnos serios problemas. Además, algunos distribuidores han tenido problemas de control de calidad dando como resultado la asignación de direcciones duplicadas. Afortunadamente, este último caso es bastante raro.

⁹ En las redes conmutadas de hoy en día, éste no es necesariamente el caso. Véase la breve explicación sobre los interruptores al final del Capítulo 1, "Prólogo a una guía práctica".

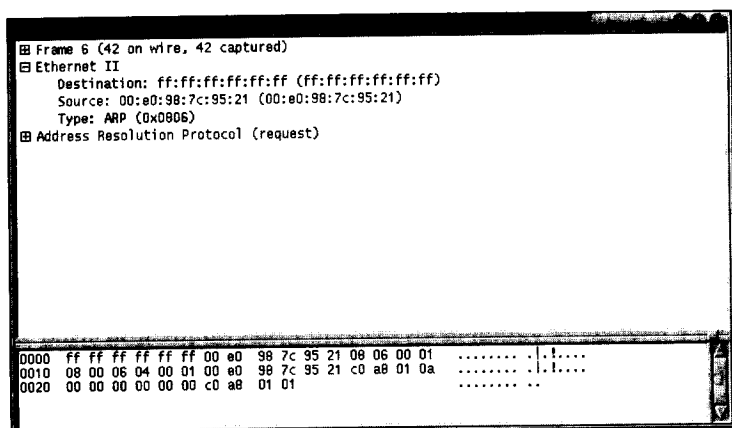


Figura 2.3. Un marco Ethernet capturado.

PPPoE

EL RFC 2516 describe PPPoE (*PPP over Ethernet*, PPP en Ethernet), un método de encapsular paquetes PPP en Ethernet. Este protocolo se utiliza normalmente para ofrecer servicio de DSL (*Digital Subscriber Line*, Línea de suscriptor digital) a los suscriptores.

Estudio de PPPoE

El protocolo PPOE separa el tráfico en dos clases, cada una de ellas enviada durante una etapa diferente de la sesión de red. La sesión comienza con una *etapa de descubrimiento* y después pasa a una *etapa de sesión PPP*. El ID de tipo de Ethernet es diferente para las dos etapas. En la etapa de descubrimiento, es 0x8863; en la etapa de sesión PPP, es 0x8864. El encabezamiento PPPoE es el mismo durante ambas fases del protocolo.

El encabezamiento PPOE es de 6 bytes y contiene cinco campos. Los primeros 4 bits componen el campo de Versión y se establecen en 0x1. Los segundos 4 bits denotan el campo de Tipo y se establecen en 0x1. El siguiente campo es de 1 byte de largo y representa el campo de Código; este campo se define de manera separada para las etapas de descubrimiento y de sesión PPP (véanse las dos siguientes secciones para más información). El siguiente campo es el campo de ID de sesión, que tiene 2 bytes de longitud (véase la sección “Etapa de descubrimiento” para la descripción de este campo). El último campo, el campo de Longitud, tiene 2 bytes e indica la longitud de la carga útil del PPPoE (excluyendo los encabezamientos PPPoE y Ethernet).

En una sesión PPPoE resultan implicados dos *hosts*. El *host* de inicio se llama cliente; es nuestro equipo. El servidor (también llamado *concentrador de acceso*) es el *host* con el que nos estamos conectando en el ISP.

Etapas de descubrimiento

Durante la etapa de descubrimiento, la carga útil del paquete PPPoE lleva una serie de *tags* bien definidos. Cada *tag* está compuesto de tres campos. El primero tiene 2 bytes de longitud y representa el tipo de *tag*. El segundo campo representa la longitud de *tag* y tiene también 2 bytes de largo. El tercero es el campo de Valor del *tag*, que tiene una longitud variable. La Tabla 2.2 muestra algunos tipos y valores de *tag* con su significado.

Tabla 2.2. Valores y tags de la etapa de descubrimiento de PPOE.

Tag	Nombre de tag	Significado y valor
0x0000	Fin de lista	Este <i>tag</i> indica que no hay más indicadores en el paquete. Su longitud de <i>tag</i> es de 0x0000; no hay valor de <i>tag</i> . Este <i>tag</i> es opcional.
0x0101	Nombre de servicio	Este <i>tag</i> se utiliza para solicitar o aprobar un servicio dentro de la sesión PPPoE (por ejemplo, calidad de servicio). El valor del <i>tag</i> lleva el nombre del servicio.
0x0102	Nombre del concentrador de acceso	Este <i>tag</i> da el nombre del concentrador de acceso (el servidor PPPoE). El nombre es un identificador único. El valor del <i>tag</i> es con frecuencia una representación de la dirección MAC del servidor.
0x0105	Específico del distribuidor	Este <i>tag</i> puede utilizarse para pasar información patentada del distribuidor. El contenido del campo de Valor de <i>tag</i> comienza con un ID de distribuidor de 4 bytes, que es 0x00, seguido de la porción del distribuidor de 3 bytes de la dirección MAC de Ethernet. No se recomienda la utilización de este <i>tag</i> y puede ser ignorado legalmente por cualquier implementación.
0x0201	Error de nombre de servicio	Este <i>tag</i> tiene normalmente una longitud de 0x00 e indica que el servicio solicitado no puede ser concederse. Si hay datos en el campo de Valor de <i>tag</i> , deben contener una explicación que se pueda imprimir de la negación de la petición de servicio.

Iniciación del descubrimiento activo de PPPoE

La etapa de descubrimiento comienza cuando el cliente envía un paquete de PADI, (*PPPoE Active Discovery Initiation*, Iniciación del descubrimiento activo de PPPoE) a la dirección de difusión de Ethernet. El campo de Código

se establece en 0x09. El ID de sesión se establece en 0x0000. El paquete PADI debe llevar un *tag* de nombre de servicio y puede no llevar otros *tags*.

Oferta de descubrimiento activo de PPPoE

Cualquier concentrador de acceso de la red local puede responder con un paquete de PADO (*PPPoE Active Discovery Offer*, Oferta de descubrimiento activo de PPPoE) si es capaz de servir la solicitud. En un paquete PADO, el campo de Código se establece en 0x09 y el ID de sesión se establece en 0x0000.

El paquete PADO debe llevar un *tag* de nombre de servicio que se corresponda con el del paquete PADI y un *tag* de nombre AC que contenga su propio nombre único. Puede contener también cualquier cantidad de *tags* de nombre de servicio que indiquen los servicios que ofrece el concentrador de acceso.

Solicitud de descubrimiento activo de PPPoE

Habiendo recibido uno o más paquetes PADO, el cliente selecciona uno y responde con un paquete de PADR (*PPPoE Active Discovery Request*, Solicitud de descubrimiento activo de PPPoE). El campo de Código se establece en 0x19 y el ID de sesión se establece en 0x0000. El paquete PADR debe contener un *tag* de nombre de servicio que indique el servicio que solicita el cliente. También puede contener otros *tags*.

Confirmación de sesión de descubrimiento activo de PPPoE

El servidor responde a un paquete PADR con un paquete de PADS (*PPPoE Active Discovery Session Confirmation*, Confirmación de sesión de descubrimiento activo de PPPoE). El campo de Código se establece en 0x65 y el ID de sesión se establece en un valor de 2 bytes que identifica de manera única esta sesión.

El paquete PADS debe contener un *tag* de nombre de servicio y puede contener otros *tags*. Si no puede aceptar el nombre de servicio del PADR, debe responder con un PADS que contenga un *tag* de error de nombre de servicio. El ID de sesión se establece en 0x0000.

Cuando se ha enviado el paquete PADS, se ha iniciado la sesión PPPoE y comienza la etapa de sesión PPP.

Terminación de descubrimiento activo de PPPoE

Cualquier *host* implicado en la sesión PPPoE puede cerrar la sesión enviando un paquete de PADT (*PPPoE Active Discovery Terminate*, Terminación de descubrimiento activo de PPPoE). El paquete PADT tiene un campo de Código de 0xa7 y un ID de sesión que se corresponde con el identificador único de las sesiones. En un paquete PADT no se necesitan *tags*.

Después de que se haya recibido un paquete PADT no se puede enviar más tráfico por una sesión PPPoE.

Etapas de sesión PPP

Cuando se ha completado la etapa de descubrimiento, puede dirigirse tráfico PPP normal dentro de la sesión PPPoE. Todo el tráfico Ethernet es de unic conversión a lo largo de la sesión. El campo de Código de PPPoE se establece en 0x00 para todo el tráfico de la sesión y la carga útil de PPPoE es un paquete PPP.

PPPoE descodificado

La Figura 2.4 muestra un marco PADI. El campo de Versión es 0x1. El campo de Tipo es 0x1. El campo de Código es 0x09 (PADI). El ID de sesión es 0x0000. El campo de Longitud de carga útil es 0x0004. El paquete tiene sólo un *tag*, un *tag* de nombre de servicio. El tipo de *tag* es 0x0101. La longitud del tag es 0x0000.

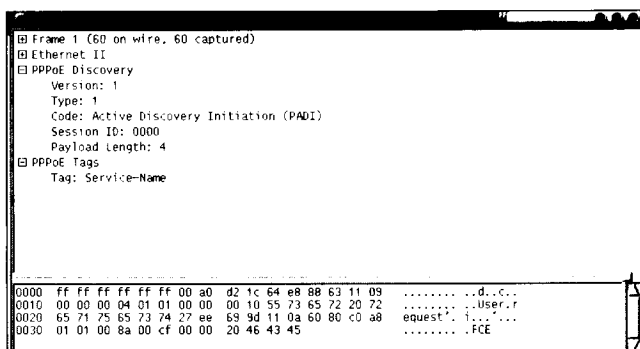


Figura 2.4. Un marco PADI de PPPoE.

La Figura 2.5 muestra un marco PADO. El campo de Versión se establece en 0x1, igual que el campo de Tipo. El campo de Código es 0x07 (PADO). El ID de sesión es 0x0000. El campo de Longitud de carga útil es 0x002a. Este marco tiene tres *tags*: un *tag* de nombre de concentrador de acceso, un *tag* de nombre de servicio y un *tag* de *cookie*. El *tag* de nombre de concentrador de acceso tiene un tipo de *tag* de 0x0102. Su longitud de *tag* es de 0x000e. Contiene la cadena de 15 bytes “crashtestdummy” como su valor de *tag*.

En la Figura 2.6 podemos ver un marco PADR. En este paquete los campos de Versión y de Tipo se establecen en 0x1. Su campo de Código es 0x19 (PADR). El ID de sesión aún es 0x0000. El campo de Longitud de carga útil es 0x0018. Este paquete lleva dos *tags*: un *tag* de nombre de servicio y un *tag* de *cookie*.

La Figura 2.7 muestra un marco PADS. Sus campos de Versión y de Tipo son 0x1. El campo de Código se establece en 0x65 (PADS). Se ha asignado el ID de sesión y es 0x0001. La Longitud de carga útil es 0x0004. Este paquete sólo tiene un *tag*, el *tag* de nombre de servicio.

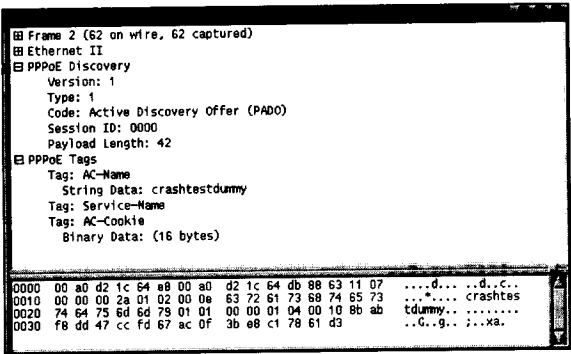


Figura 2.5. Un marco PADO de PPPoE.

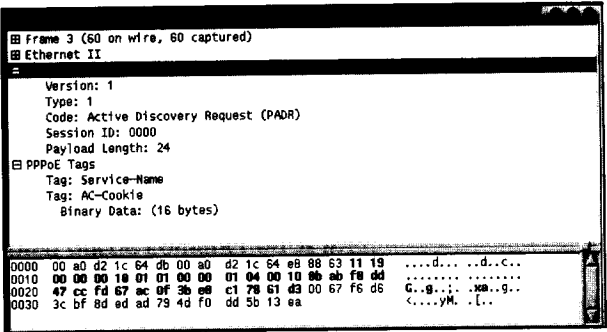


Figura 2.6. Un marco PADR de PPPoE.

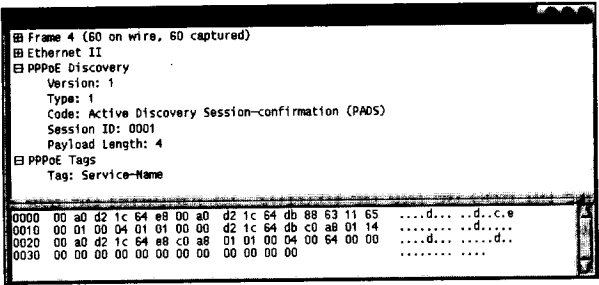


Figura 2.7. Un marco PADS de PPPoE.

En la Figura 2.8 se muestra un típico marco de sesión PPPoE. Tiene unos campos de Versión y de Tipo de 0x1. El campo de Código es 0x00 (Sesión). El ID de sesión es 0x0001. La Longitud de carga útil es 0x0016. La carga útil de este paquete es un paquete LCP de PPP.

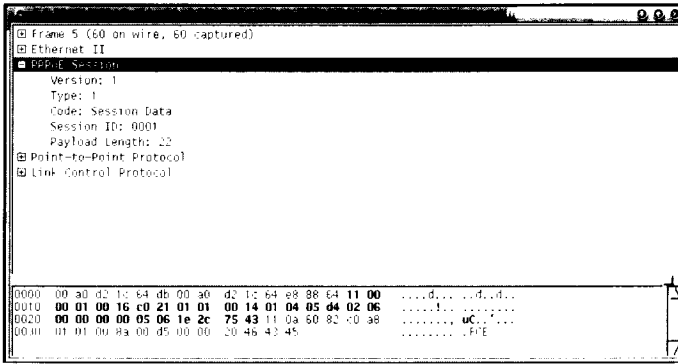


Figura 2.8. Un marco de sesión de PPPoE.

La interfaz de bucle de retorno

La *interfaz de bucle de retorno* es un dispositivo lógico que ofrece la mayoría de las pilas de protocolos IP. Utiliza la red de *clase A* 127 y normalmente se le asigna la dirección 127.0.0.1 y el nombre *localhost*. En un *host* Linux, la interfaz lo se utiliza para hacer un bucle de retorno, como mostramos aquí:

```
[root@cherry /root]# ifconfig lo
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:3924  Metric:1
        RX packets:18 errors:0 dropped:0 overruns:0 frame:0
        TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0

[root@cherry /root]#
```

El tráfico enviado a la dirección de bucle de retorno no aparece nunca en ninguna red adjunta. Un aspecto especial de esta interfaz es que el tráfico enviado desde un *host* a sí mismo se redireccionará a la dirección de bucle de retorno.

La interfaz de bucle de retorno se utiliza normalmente para establecer conexiones de red entre aplicaciones basadas en *cliente-servidor* hospedadas en la misma máquina. (como el sistema de ventanas X11). Esto ayuda a evitar el tráfico de la red local e incluso permite que dichas aplicaciones funcionen en ausencia de una red física.

La Figura 2.9 muestra una interfaz de bucle de retorno con un PPP y una interfaz Ethernet. El flujo lógico de tráfico a y desde la capa IP lo muestran las flechas del diagrama.

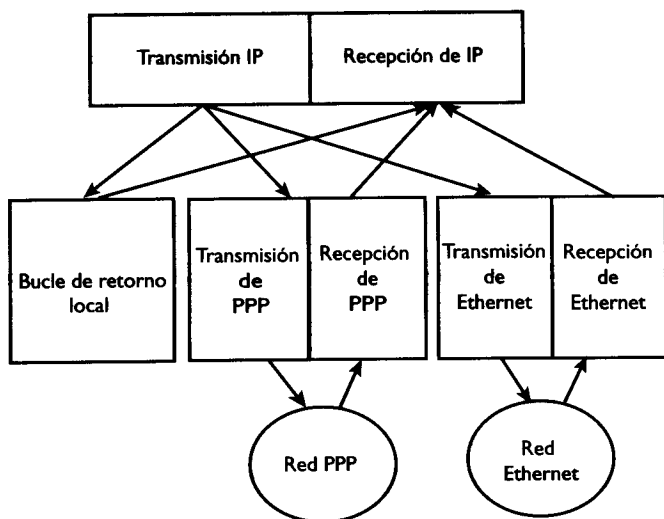


Figura 2.9. Diagrama de la interfaz de bucle de retorno.

ARP y RARP

Si el protocolo de la capa de enlace soporta múltiples *hosts* (como Ethernet II), IP debe tener un medio de relacionar una dirección IP con la dirección de la capa de enlace. Esto se consigue mediante ARP (*Address Resolution Protocol*, Protocolo de resolución de direcciones). El ARP de IP en Ethernet está definido en el RFC 826. RARP (*Reverse ARP*, ARP inverso) es un protocolo muy relacionado que se define en el RFC 903.

Si volvemos a pensar en la analogía de la habitación llena de gente hablando que utilizamos para representar una LAN Ethernet, ARP sería como preguntar: “¿quién es Tom?” Cuando alguien conteste, podemos empezar a hablar con él. En realidad, ARP pregunta: “¿quién tiene la dirección IP X.X.X.X?” Cuando obtiene respuesta, comienza una conversación de nivel IP con ese *host*.

ARP NO ES ENCAMINABLE

Como ARP va directamente con los protocolos de la capa de enlace, no con un protocolo de la capa de red, no es encaminable ¹⁰.

¹⁰ Como algunos *hosts* no pueden determinar adecuadamente su propia red y ARP para direcciones de redes remotas, existe un método alternativo. *Proxy ARP* permite que un encaminador conteste a peticiones ARP de *hosts* de cualquier red remota a la que sepa llegar. Véase el Capítulo 3 para más información sobre el encaminamiento.

ARP

ARP necesita dos tipos de paquetes: peticiones ARP (que piden la dirección MAC que se corresponde con una dirección IP) y respuestas ARP (que dan la correspondencia entre dos direcciones). En la mayoría de los casos, la respuesta la envía la máquina que tiene asignada la dirección IP que se solicita. Algunos *hosts* más antiguos no pueden determinar adecuadamente si un *host* es local de su red. En este caso, ARP proporciona un mecanismo para contestar a los ARP inapropiados de esos sistemas, *proxy ARP*. Linux nos permite configurar un *host* para dar respuestas de *proxy ARP* configurando una entrada de caché ARP tal y como se edita con el indicador. Cuando está establecido este marcador, también puede estarlo el indicador y las entradas de caché ARP pueden hacerse a un único *host* o a toda una subred.

Los paquetes de peticiones ARP están compuestos de lo siguiente: un tipo de hardware de 2 bytes, un tipo de protocolo de 2 bytes, una longitud de dirección de hardware de 1 byte, una longitud de dirección de protocolo de 1 byte, un *opcode* de 2 bytes con un valor de 0001, una dirección de hardware de remitente de longitud variable, una dirección de protocolo de remitente de longitud variable, una dirección de hardware de objetivo de longitud variable y una dirección de protocolo de objetivo de longitud variable. La dirección de hardware de objetivo normalmente se rellena con ceros.

Los paquetes de respuesta ARP son parecidos a los de peticiones descritos anteriormente, excepto en que el valor de *opcode* es 0002 y que el campo de Dirección de hardware de objetivo se rellena con la dirección MAC apropiada.

Aunque ARP se utiliza generalmente para determinar la dirección de hardware de otro sistema, algunas máquinas están configuradas para llevar a cabo un “ARP gratuito”. Cuando el *host* saca su tarjeta de interfaz de red, utiliza ARP como su propia dirección IP. Este mecanismo puede ayudar a reducir los conflictos de direcciones IP.

El sistema operativo mantiene la información ARP en un *host* en una tabla ARP.

En una plataforma Linux, puede verse esta información con el comando ARP, como en este ejemplo:

```
[root@cherry /root]# arp -a
? (192.168.1.1) at 00:A0:D2:1C:64:E8 [ether] on eth0
? (192.168.1.11) at 00:C0:F0:26:B1:EC [ether] on eth0
[root@cherry /root]#
```

Las entradas de la tabla ARP (más apropiadamente, la caché ARP) pueden ser permanentes (creadas con las opciones *o*) o dinámicas (descubiertas automáticamente). Normalmente, las entradas dinámicas tienen un tiempo muerto de 15 minutos; es decir, a un *host* que no se haya comunicado en 15 minutos o más se le volverá a pasar por ARP. Este proceso anticuado de las entradas de caché ARP minimiza el peligro de que una tarjeta de red sea re-

emplazada en un *host* remoto, cambiando así la relación dirección MAC-dirección IP.

RARP

RARP se utiliza para permitir que un *host* consiga su propia dirección IP dinámicamente¹¹. Una gran diferencia es que en ARP, cada *host* es responsable de hacer la correspondencia de su propia dirección IP con su dirección MAC, mientras que en RARP, se configuran uno o más servidores para proporcionar la correspondencia de los *hosts* que estén haciendo RARP.

Los paquetes RARP están identificados con el tipo de Ethernet II de 0800. Las peticiones tienen un *opcode* RARP de 3 y las respuestas tienen un *opcode* de 4. No hay *opcodes* de “error” o “no encontrado”, porque otro *host* de la red puede responder a la respuesta, pero, por lo demás, los paquetes RARP son parecidos a la estructura de los paquetes ARP.

Uno de los principales fallos de RARP es que es no encaminable. Como es un protocolo de capa de enlace, está confinado a una única LAN. Esto significa que cada LAN que tenga *hosts* haciendo RARP debe tener al menos un residente servidor RARP configurado.

ARP descodificado

En la petición ARP que mostramos en la Figura 2.10, mango está solicitando una dirección de hardware a cherry. El tipo de hardware está establecido en 0x0001 (Ethernet) y el tipo de protocolo en 0x0800 (IPv4). La longitud de dirección de hardware es 0x06 y la longitud de dirección de protocolo es 0x04, 6 y 4 bytes respectivamente. El *opcode* ARP está establecido en 0x0001 (petición). La dirección de hardware del remitente está establecida en 00:a0:d2:1c:64:a8 y la dirección de protocolo del remitente es 192.168.1.1. La dirección de hardware del objetivo es desconocida y se establece en 00:00:00:00:00:00, mientras que la dirección de protocolo del objetivo se establece en 192.168.1.10.

Cuando cherry responde a la petición de mango, la mayor parte de los datos siguen siendo los mismos. El *opcode* ARP se cambia a 0x0002 (respuesta) y las direcciones del remitente y del objetivo se han intercambiado. Como cherry le está dando a mango su dirección de hardware, la dirección de hardware del remitente se ha cambiado de todo 0 a 00:e0:98:7c:95:21 (véase la Figura 2.11).

La Figura 2.12 muestra un paquete RARP enviado por un servidor Sun que se está cargando de la red. Igual que en los paquetes ARP que vimos anteriormente, el tipo de hardware es 0x0001 (Ethernet), el tipo de protocolo es 0x0800 (IPv4), la longitud de hardware es 0x06 (6 bytes) y la longitud de pro-

¹¹ RARP ha sido en su mayor parte superado por los protocolos BOOTP y DHCP.

protocolo es 0x04 (4 bytes). El *opcode* es 0x0003, representando una petición RARP. Las direcciones de hardware de remitente y de objetivo están ambas establecidas en 08:00:20:7e:40:af, la dirección MAC de la máquina que está solicitando una dirección IP. Las direcciones IP de remitente y de objetivo están ambas establecidas en 255.255.255.255.

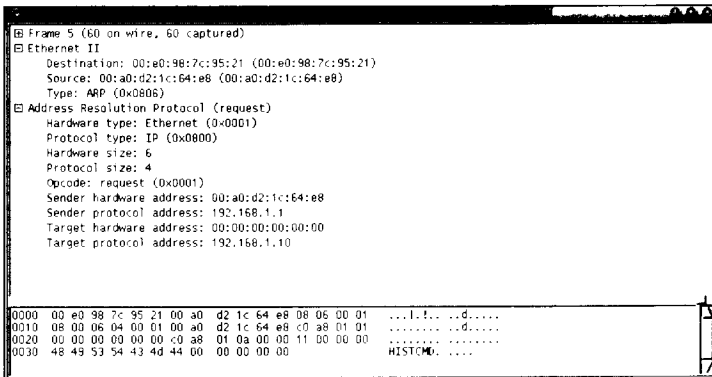


Figura 2.10. Un paquete de petición ARP.

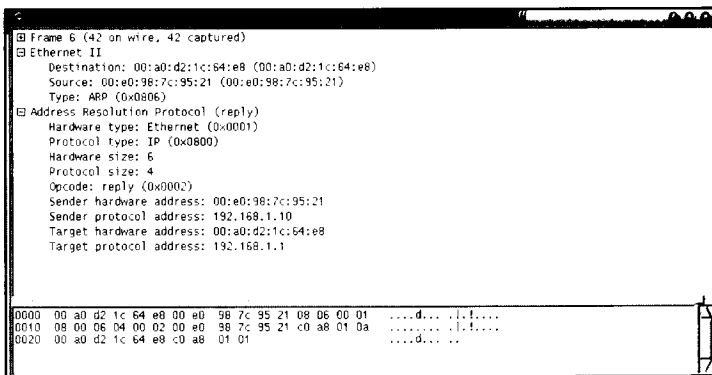


Figura 2.11. Un paquete de respuesta ARP.

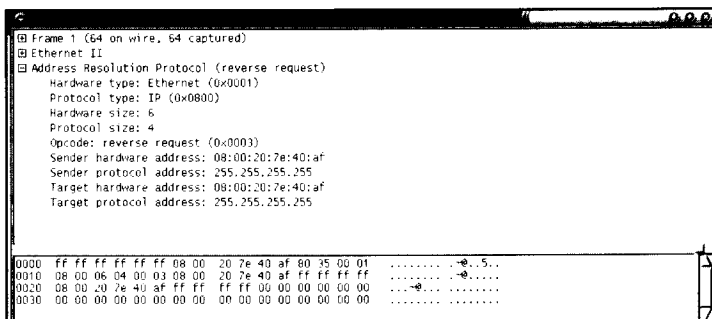


Figura 2.12. Un paquete RARP.

MTU

Cada uno de los protocolos de la capa de enlace impone diferentes límites a los paquetes que lleva. Entre estas restricciones está el tamaño máximo del paquete, llamado MTU (*Maximum Transmission Unit*, Unidad de transmisión máxima). Normalmente, PPP utiliza una MTU de 296, Ethernet una de 1500 y FDDI y Token Ring de 4Mbps una de 4464.

Como cada capa de enlace puede definir una MTU diferente y un paquete podría atravesar múltiples redes camino de su destino final, se define una *MTU de ruta*. En el Capítulo 3. “Protocolos de la capa de red”, veremos la fragmentación IP (un posible resultado de tener MTU diferentes) y cómo evitarla utilizando el descubrimiento de la MTU de ruta. En la Figura 2.13 se muestra un ejemplo de MTU de ruta.

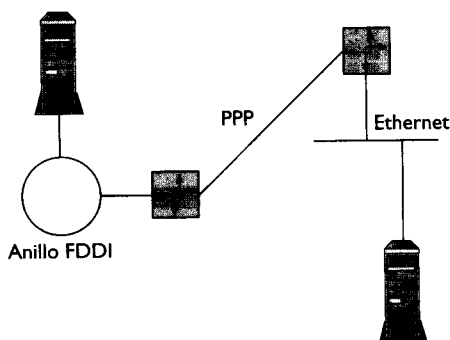


Figura 2.13. Una internet con varios protocolos de capa de enlace representados.

En este ejemplo, la MTU de ruta es 296 porque el enlace PPP que se encuentra entre la LAN Ethernet por un lado y el anillo FDDI por el otro ha negociado un índice más bajo del que se utiliza normalmente.

3

Protocolos de la capa de red

La capa de red (o capa de *internetwork*) marca el punto en que el tráfico escapa de los confines de una red única. También señala el punto donde las direcciones dan con su primera capa de abstracción (como mostramos en la sección “ARP” del Capítulo 2, “Protocolos de la capa de enlace”). En nuestro debate sobre la capa de red, cubriremos las direcciones IP, las subredes y superredes, el encaminamiento IP, el filtrado de paquetes, la NAT (*Network Address Translation*, Traducción de direcciones de red) y la versión 4 de IP (la versión que más se utiliza actualmente).

Debido a que las direcciones IP, las subredes y el encaminamiento están ligados de forma muy intrincada, es difícil hablar de uno de ellos sin hacer referencia a los otros. Si esto le provoca algo de confusión cuando lo lea, por favor, satisfaga su curiosidad recurriendo a la sección apropiada en busca de una aclaración.

Direcciones IP

Antes de hablar del protocolo IP, nos tomaremos algo de tiempo para repasar las direcciones IP. Se asignan una gran variedad de direcciones IP a las organizaciones. En la mayoría de los casos son asignadas por el ISP de las empresas, aunque una compañía puede solicitar sus propias direcciones IP en algunas circunstancias excepcionales. Cualquiera de estas direcciones IP asignadas puede encaminarse por Internet a menos que sea bloqueada por un *firewall*¹.

¹ Más concretamente, podrían bloquearse por un filtro de paquetes. Los filtros de paquetes con frecuencia son parte de un *firewall*. Esto es una diferencia pequeña, pero importante.

Las direcciones IP indican el tamaño de la red IP local y (en algunos casos) para qué se utiliza. Esto se consigue dividiendo las redes en clases, como muestra la Tabla 3.1.

Tabla 3.1. Clases de direcciones de red.

Tipo de red	Primer byte de dirección	Patrón de identificación	Notas
Clase A	0-127	El primer bit es un cero	Las redes de clase A proporcionan 16.777.214 <i>hosts</i> por red. La dirección de red es de 1 byte de longitud.
Clase B	128-191	Los primeros 2 bytes son 10	Las redes de clase B proporcionan 65.534 <i>hosts</i> por red. La dirección de red es de 2 bytes de longitud.
Clase C	192-223	Los primeros 3 bytes son 110	Las redes de clase C proporcionan 254 <i>hosts</i> por red. La dirección de red es de 3 bytes de longitud.
Clase D	224-239	Los primeros 4 bytes son 1110	Las direcciones de clase D se utilizan para tráfico de multiconversión.
Clase E	240-247	Los primeros 5 bytes son 11110	Las direcciones de clase E están reservadas para uso experimental.

Además de estas divisiones de las direcciones IP, se hace otra distinción. Algunas de estas direcciones se apartan para direcciones privadas. El RFC 1918 aparta las siguientes direcciones para *hosts* internos de una compañía:

- 10.0.0.0
- 172.16-31.0.0
- 192.168.0-255.0

Estas direcciones pueden utilizarse como veamos que encajan, siempre que no sean encaminadas por Internet. Permite que todo el mundo utilice el mismo conjunto de direcciones para sus redes internas. La amplia utilización de las direcciones del RFC 1918 es posible por el uso de NAT, que explicaremos más adelante en este capítulo y también en el Capítulo 11, “Herramientas de seguridad”. Por favor, ayudemos a conservar las direcciones IP disponibles utilizando direcciones privadas. También es mejor para la seguridad de esas redes la utilización del espacio de dirección del RFC 1918 para nuestras redes internas. Utilizaremos las direcciones del RFC 1918 en los ejemplos tanto como nos sea posible. (Véase la sección “Traducción de direcciones de red”, en este mismo capítulo, para más información.)

Debido a que las redes de clase A y B son demasiado grandes para una LAN normal, y como a veces tampoco encaja una red de clase C, existe un método para dividir una red en porciones más pequeñas. Este método se llama subred. Las subredes se forman con la *máscara de subred* o *máscara de red*, otro número de 4 bytes. Hablaremos de las subredes en la siguiente sección, por lo que dejaremos la explicación para más adelante.

Cuando un equipo transmite tráfico IP, debe tomar una decisión sobre si el *host* objetivo es local o remoto. Los *hosts* locales se direccionan directamente, lo que significa que se enviará el tráfico a la dirección MAC que se corresponda con la dirección IP objetivo. El tráfico de los *hosts* remotos se enviará a la pasarela local de esa red remota (veremos el encaminamiento en una sección posterior; por ahora, sólo diremos que se produce). Esta decisión de encaminamiento se toma basándose en la dirección IP local (o de origen) con su máscara de red y la dirección IP de destino. La dirección IP de origen sufre una operación AND² bit a bit con la máscara de red, igual que la dirección de destino. Después, se comparan los dos números resultantes (como muestra la Figura 3.1).

11000000 10101000 00000001	0001000	192.168.1.16	Dirección
11111111 11111111 11111111	0000000	255.255.255.0	Máscara de red
11000000 10101000 00000001	0001000	192.168.1.16	Red

Figura 3.1. Dirección IP y máscara de red con AND.

Si los resultados de estas dos operaciones AND bit a bit son iguales, los dos equipos son locales. En el caso contrario, los dos equipos son remotos.

Ejemplo 3.1. Decisión de dirección local.

Origen	Destino
Dirección: 192.168.1.100	Dirección: 192.168.1.25
Máscara de red: 255.255.255.0	Máscara de red: 255.255.255.0
Resultado: 192.168.1.0	Resultado: 192.168.1.0

Como los resultados del Ejemplo 1 concuerdan (la dirección de red de los dos *hosts* sufrieron AND con la máscara de subred local), estos *hosts* son locales el uno con respecto al otro; se utilizará la dirección directa.

² El AND bit a bit significa que se comprueba cada bit de las cosas. Si ambas son verdaderas (1), entonces el resultado es verdadero. Si una o ambas son falsas (0), entonces el resultado es falso. Si se ha hecho AND bit a bit a 10 (00001010) y a 12 (00001100), el resultado sería 8 (00001000).

Ejemplo 3.2. Decisión de dirección remota.

Origen	Destino
Dirección: 192.168.1.100	Dirección: 192.168.10.25
Máscara de red: 255.255.255.0	Máscara de red: 255.255.255.0
Resultado: 192.168.1.0	Resultado: 192.168.10.0

Como los resultados del Ejemplo 2 no concuerdan, estos *hosts* no son locales el uno respecto al otro; será necesario encaminar el tráfico entre ellos.

Subredes y superredes

Las subredes y las superredes proporcionan un medio para utilizar las direcciones IP de manera más eficiente. Las subredes llevan en funcionamiento casi tanto como las direcciones IP, pero las superredes son mucho más modernas (y, por tanto, no se conocen tan bien). En esta sección, cubriremos la mecánica de las subredes, daremos algunos ejemplos prácticos y hablaremos de la mecánica de las superredes.

Estudio de las subredes

La mayoría de las empresas se encuentran con que una dirección de clase A es o demasiado grande o demasiado pequeña para sus necesidades. Tomemos por ejemplo una pequeña compañía de software. Digamos que tiene cuatro redes en la oficina: una LAN para los departamentos de contabilidad y recursos humanos, así como para los ejecutivos; una LAN para que los desarrolladores prueben las versiones inestables de su producto; la LAN que utilizan todos los demás; y la red segura en la que se encuentran los *hosts* accesibles a Internet de la empresa. Ninguna de las LAN tiene más de 14 equipos (incluido el encaminador que conecta las cuatro LAN). Esta empresa no necesita cuatro redes de clase C; puede arreglárselas con sólo una dividiéndola en porciones del tamaño adecuado.

Las subredes nos ofrecen un método de fraccionar las redes. Nos dan un mecanismo para dividir la porción de *host* de la dirección IP³ en dos secciones: una porción de subred y una porción de *host*. El diagrama de la Figura 3.2 nos muestra esta división.

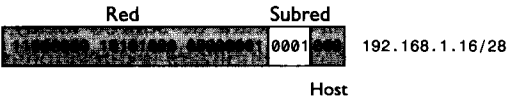


Figura 3.2. Una dirección IP mostrando los campos de subred y de host.

³ La porción de *host* puede ser de 1, 2 o 3 bytes, dependiendo de la clase de la dirección.

Para dividir una red en subredes, debemos tener una máscara de subred⁴. Las máscaras se especifican indicando qué bits se utilizan para direcciones de red o de subred. Sería bastante tedioso decir que para la dirección de red 192.168.1.0, la máscara era el siguiente patrón de bits: 1111111111111111111111111100000000. En su lugar, indicamos la máscara en decimales con puntos (como una dirección IP), 255.255.255.0, o mostrando el número de bits utilizado en la máscara como sufijo de la dirección IP, así: 192.168.1.0/24.

El número de bits del campo de la subred nos dice cuántas subredes y *hosts* hay disponibles utilizando esta máscara. En el campo de *host*, la dirección de todo unos representa la dirección de difusión y la dirección de todo ceros representa la dirección de red, igual que una dirección IP estándar. El campo de subred sólo necesita que la dirección de todo unos se reserve para una dirección de todo subredes, aunque algunas de las primeras implementaciones utilizaban las direcciones de todo ceros para este propósito⁵. Por ejemplo, la dirección y red 192.168.1.33/27 se dividen en la Tabla 3.2.

Tabla 3.2. Una dirección IP dividida en subredes.

1.º Byte	2.º Byte	3.º Byte	4.º Byte	Decimal	Notas con puntos
11000000	10010100	00000001	00100001	192.168.1.33	La dirección IP.
11111111	11111111	11111111	11100000	255.255.255.224	La máscara de subred.
11000000	10010100	00000001	XXXXXXXX	192.168.1.0	Red, 3 bytes de una dirección de clase C.
XXXXXXXX	XXXXXXXX	XXXXXXXX	001XXXXX	1	Subred.
XXXXXXXX	XXXXXXXX	XXXXXXXX	XXX00001	1	El host.

La Tabla 3.2 nos muestra que el *host* 192.168.1.33/27 es el *host* 1 de la red 192.168.1.0, subred 1. ¿Un poco confuso? No se preocupe, resulta más fácil después de unas cuantas repeticiones. Veamos algunos ejemplos más (Dejaremos la representación binaria de cada ejemplo como ejercicio):

- 192.168.1.10/24 es el *host* 10 de la red 192.168.1.0; esta red no está dividida en subredes.

⁴ En realidad, cada dirección IP utiliza una máscara de subred. Allí donde no se da la máscara, se toma la máscara predeterminada de esa clase de red (24 bits para una dirección de clase C, 16 para una dirección de clase B y 8 para una dirección de clase A).

⁵ Si nos encontramos en una red que incluye sistemas basados en BSD4.2 (como los equipos Sun o Ultrix viejos), deberíamos evitar la utilización de la subred de todo ceros. Si no tenemos tal cosa en nuestro entorno, no tenemos por qué preocuparnos.

- 192.168.1.26/28 es el *host* 10 de la red 192.168.1.0, subred 1.
- 192.168.1.128/28 es la dirección de red (*host* 0) de la red 192.168.1.0, subred 8.
- 192.168.1.127/28 es la dirección de difusión (*host* 16) de la red 192.168.1.0, subred 7.
- 192.168.1.26/30 es el *host* 2 de la red 192.168.1.0, subred 6.

Las subredes funcionan igual de bien en redes que no sean de la clase C:

- 172.16.1.10/24 es el *host* 10 de la red 172.16.0.0, subred 1.
- 172.16.1.129/25 es el *host* 1 de la red 172.16.0.0, subred 3.

La máscara de red se define con el comando `/sbin/ifconfig`, como sigue:

```
/sbin/ifconfig eth0 192.168.1.10 netmask 255.255.255.0
```

Este ejemplo es un poco artificioso, 24 bits es la máscara predeterminada para una dirección de clase C. Sin embargo, si vamos a tomarnos el trabajo de definir una máscara de subred, probablemente deseemos otro valor que no sea el predeterminado. Si queremos asignar una máscara de 28 bits, deberíamos hacer esto:

```
/sbin/ifconfig eth0 192.168.1.10 netmask 255.255.255.224
```

Representa el *host* 10 de la red 192.168.1.0, subred 0.

Probablemente sea útil tener una pequeña tabla de máscaras de subred y el número de subredes y *hosts* que proporcionan⁶ (véase la Tabla 3.3).

Tabla 3.3. Redes divididas en subredes de varios tamaños.

Dirección de red	Máscara de subred	Redes disponibles	Hosts por subred	Número total de hosts por red
192.168.1.0	255.255.255.240	15	14	210
192.168.1.0	255.255.255.192	3	62	186
172.16.0.0	255.255.255.128	511	126	64386
172.16.0.0	255.255.255.254	127	510	64770

EJEMPLO REAL

Hace varios años, trabajé para una empresa de puesta a punto de sistemas y era el único del equipo que conocía realmente las máscaras de subred. Un día, se presentó un cliente con la necesidad urgente de reconfigurar un esquema de redes. La compañía acababa de abrir otra oficina y necesitaba unirla con las dos ya existentes. Aunque habíamos

⁶ Como sólo se llega a comprender de verdad las subredes utilizándolas repetidamente, dejaremos la finalización de esta hoja como ejercicio. También hay herramientas disponibles para calcular máscaras de subred. Ahorran mucho tiempo, pero deberíamos saber cómo funciona todo esto antes de empezar a confiar en ellas.

planeado el crecimiento, la nueva oficina tenía más *hosts* de lo que se había proyectado. Sólo había que rehacer las máscaras y asignar intervalos de direcciones nuevos. Podrán imaginarse a quién llamaron para que lo hiciera. El único inconveniente era que mi mujer y yo estábamos en el hospital, donde se estaba preparando para tener un bebé; afortunadamente, no se enfadó demasiado cuando recibí la llamada. No tenía herramientas para hacer subredes, así que terminé haciendo los cálculos mentalmente y diseñando un mapa en el envoltorio de unos guantes para pasárselo a nuestro cliente.

Un *host* sólo conoce su propia máscara de subred. Cualquier comparación que haga está basada solamente en su propia máscara. Esto puede causar problemas a veces. Si dos *hosts* de una red local tienen máscaras de subred diferentes, puede que no sean capaces de comunicarse con IP.

Estudio de las superredes

Del mismo modo en que las subredes se utilizan para crear redes de un tamaño adecuadamente más pequeño, existe un método para alargar redes, proporcionando así redes más grandes y encaminadas más fácilmente. Las superredes ayudan a resolver dos problemas. En primer lugar, no hay redes de tamaño adecuadamente más grande para las organizaciones que las necesitan (las redes de clase C no son lo suficientemente grandes para la mayoría de las empresas y no hay suficientes redes de clase B para dar a las compañías que en realidad sólo necesitan 10 o 12 bits de espacio de dirección de *host*). En segundo lugar, las tablas de encaminamiento de los encaminadores del núcleo de Internet se hacen poco flexibles cuando se llenan de muchas redes de clase C diferentes en la periferia de Internet.

El RFC 1518 y el RFC 1519 definen el CIDR (*Classless Interdomain Routing*, Encaminamiento entre dominios sin clase, pronunciado “cider”), o superredes. Este mecanismo permite la combinación de grandes bloques de direcciones contiguas. Esta agregación de redes ayuda a aligerar el problema de encaminamiento y también da a las organizaciones de tamaño medio el espacio de dirección de *host* que necesitan.

Para la utilización de las superredes es necesario que los encaminadores conozcan la máscara de red de las redes que llevan. Las implementaciones actuales de los principales protocolos de encaminamiento (RIP-2, BGP-4 y OSPF) llevan toda la información necesaria.

CÓMO COMPRENDER LAS SUPERREDES

Para comprender las superredes y cómo alivian los dos problemas mencionados anteriormente, veamos un ejemplo:

Un ISP pequeño, Small-ISPnet, tiene un bloque de CIDR de 192.168.192.0/20 (recordemos que esto significa que controla desde la dirección 192.168.192.0 a la 192.168.207.255, un total de 16 redes de clase C). Como el ISP solo controla estas direcciones, cualquier otro encaminador del núcleo de Internet necesita sólo una entrada de tabla de encaminamiento para todo el ISP, en lugar de 16 entradas separadas.

Small-ISPnet ofrece servicios a tres empresas, además de a sus clientes individuales. A SmallCo y LittleCo (las más pequeñas de las tres empresas) se les asigna 192.168.206.0/23

y 192.168.204.0/23, respectivamente, dando a cada compañía 510 posibles direcciones de host para sus redes. A MidSize, Inc. se le asigna 192.168.200.0/22, cediendo 1.022 direcciones de host. Todas estas asignaciones dejan a Small-ISPnet con la red 192.168.192.0/21 para su propio uso (o una subdivisión posterior, si es necesario).

Encaminamiento

El *encaminamiento* es una función principal de la capa de red y es uno de los puntos fuertes de IP. Uno de los objetivos de diseño del protocolo IP fue que una IP de internet fuera capaz de sobrevivir a pérdidas casi catastróficas de sus conexiones de *internetwork*. IP encamina cada datagrama de forma separada, de modo que si la ruta de datos seguida por un paquete resulta bloqueada, el siguiente paquete será, sencillamente, encaminado por una ruta diferente. Esto funciona muy bien en una red combinada.

REDES SENCILLAS Y COMBINADAS

Normalmente, las redes se clasifican como sencillas o combinadas (a veces incluso “profundamente combinadas”). Una red sencilla es aquella en la que hay pocas conexiones entre redes. Con frecuencia, el tráfico tiene solamente una única ruta posible a través de una red sencilla. Una red combinada es aquella en la que existen muchas conexiones entre redes. El tráfico puede pasar a menudo por múltiples rutas entre puntos de una red combinada.

Una de las claves de esta capacidad es que cualquier equipo tiene que conocer sólo su siguiente conexión para llegar a cualquier otro punto de la internet. En realidad, para muchos equipos, sólo es necesaria una única conexión; llamada *pasarela predeterminada*. Este conjunto de información se llama *tabla de encaminamiento*.

Una red sencilla

Comencemos con un ejemplo bastante sencillo de encaminamiento. Utilizaremos la internet que mostramos en la Figura 3.3.

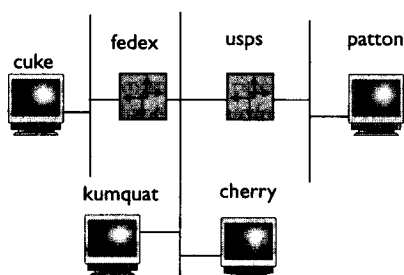


Figura 3.3. Una internet pequeña.

Cada uno de estos *hosts* (incluidos los encaminadores) tiene una tabla de encaminamiento; las mostramos aquí:

cuke:

```
[pate@cuke ~]$ netstat -rn
Tabla de encaminamiento IP de núcleo
```

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
192.168.2.51	0.0.0.0	255.255.255.255	UH	0 0		0 eth0
192.168.2.0	0.0.0.0	255.255.255.0	U	0 0		0 eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0		0 lo
0.0.0.0	192.168.2.1	0.0.0.0	UG	0 0		0 eth0

fedex:

```
[pate@fedex ~]$ netstat -rn
Tabla de encaminamiento IP de núcleo
```

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
192.168.1.2	0.0.0.0	255.255.255.255	UH	0 0		0 eth0
192.168.1.0	0.0.0.0	255.255.255.0	U	0 0		0 eth0
192.168.2.1	0.0.0.0	255.255.255.255	UH	0 0		0 eth1
192.168.2.0	0.0.0.0	255.255.255.0	U	0 0		0 eth1
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0		0 lo
0.0.0.0	192.168.1.1	0.0.0.0	UG	0 0		0 eth0

kumquat:

```
[pate@kumquat ~]$ netstat -rn
Tabla de encaminamiento IP de núcleo
```

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
192.168.1.12	0.0.0.0	255.255.255.255	UH	0 0		0 eth0
192.168.1.0	0.0.0.0	255.255.255.0	U	0 0		0 eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0		0 lo
0.0.0.0	192.168.1.1	0.0.0.0	UG	0 0		0 eth0

cherry:

```
[pate@cherry ~]$ netstat -rn
Tabla de encaminamiento IP de núcleo
```

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
192.168.1.10	0.0.0.0	255.255.255.255	UH	0 0		0 eth0
192.168.1.0	0.0.0.0	255.255.255.0	U	0 0		0 eth0
192.168.2.0	192.168.1.2	255.255.255.0	U	0 0		0 eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0		0 lo
0.0.0.0	192.168.1.1	0.0.0.0	UG	0 0		0 eth0

```
[pate@cherry ~]$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:E0:98:7C:95:21
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:3 Base address:0x200
```

usps:

```
[pate@usps ~]$ netstat -rn
Tabla de encaminamiento IP de núcleo
```

Destination	Gateway	Genmask	Flags	MSS Window	irrtt	Iface
192.168.0.2	0.0.0.0	255.255.255.255	UH	0 0		0 eth0
192.168.0.0	0.0.0.0	255.255.255.0	U	0 0		0 eth0
192.168.1.1	0.0.0.0	255.255.255.255	UH	0 0		0 eth1
192.168.1.0	0.0.0.0	255.255.255.0	U	0 0		0 eth1
192.168.2.0	192.168.1.2	255.255.255.0	U	0 0		0 eth1
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0		0 lo
0.0.0.0	192.168.0.1	0.0.0.0	UG	0 0		0 eth0

patton:

```
[pate@patton ~]$ netstat -rn
Tabla de encaminamiento IP de núcleo
```

Destination	Gateway	Genmask	Flags	MSS Window	irtt	Iface
192.168.0.12	0.0.0.0	255.255.255.255	UH	0 0	0	eth0
192.168.0.0	0.0.0.0	255.255.255.0	U	0 0	0	eth0
192.168.1.0	192.168.0.2	255.255.255.0	U	0 0	0	eth0
192.168.2.0	192.168.0.2	255.255.255.0	U	0 0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0	0	lo
0.0.0.0	192.168.0.1	0.0.0.0	UG	0 0	0	eth0

Dada esta información, trabajemos con algunos ejemplos.

Ejemplo de encaminamiento local

Si cherry quiere enviar tráfico a kumquat, primero compara su propia dirección IP con la dirección IP de kumquat en vista de su propia máscara de subred. 192.168.1.10 y 192.168.1.12 están en la misma red de clase C, y con una máscara de /24, también están en la misma subred. Como la dirección IP de kumquat es local, el datagrama IP se dirige a kumquat y se pone en un marco Ethernet con la dirección MAC de kumquat.

Cuando kumquat recibe el marco, lee la dirección IP y ve que el datagrama está dirigido a sí mismo. Cualquier tráfico de retorno pasaría por los mismos pasos.

Ejemplo de encaminamiento de salto único

Cuando cherry envía tráfico a patton, de nuevo compara las direcciones IP. En este caso, 192.168.1.10 está en una red distinta a 192.168.0.12. por lo que cherry tendrá que enviar su tráfico mediante un encaminador. Después, cherry consulta su tabla de encaminamiento. No hay rutas explícitas para la red 192.168.0.0, pero hay una entrada de pasarela predeterminada, usps, por lo que cherry encaminará a través de ella. El datagrama IP se dirige a patton y se pone en un marco Ethernet con la dirección MAC de usps.

Cuando usps recibe el marco Ethernet, ve que la dirección IP no es local. Como usps está configurado para enviar (o encaminar) paquetes, comprueba su tabla de encaminamiento IP para determinar el siguiente salto del datagrama IP. Aquí, patton se encuentra en una red conectada localmente, por lo que el datagrama se pone en un marco Ethernet con la dirección MAC de patton y se envía a través de la interfaz de red adecuada.

Cuando patton recibe el marco, comprueba la dirección IP del datagrama IP y determina que éste es para entrega local.

Después de que la pila IP haya terminado con el datagrama y la aplicación haya devuelto una respuesta, patton comprueba su propia dirección IP con la de cherry. Descubre que cherry no es local, por lo que patton tendrá que utilizar un encaminador para entregar el datagrama que lleva la respuesta.

A continuación, patton comprueba su tabla de encaminamiento y descubre que usps es el encaminador adecuado para la red 192.168.1.0/24. El datagrama se dirige entonces a cherry y se pone en un marco Ethernet dirigido a usps.

Por último, *usps* comprueba la dirección del datagrama IP y ve que es para *cherry*. Al comprobar su tabla de encaminamiento, *usps* descubre que *cherry* está conectado localmente. El datagrama se pone en un marco Ethernet nuevo dirigido a *cherry* y se envía a través de la interfaz de red adecuada.

Ejemplo de encaminamiento de dos saltos

Cuando *cuke* quiere enviar un datagrama IP a *patton*, primero compara su dirección IP con la de *patton* en vista de su propia máscara de subred. 192.168.2.51 y 192.168.0.12 están en diferentes redes, por lo que *cuke* comprueba su tabla de encaminamiento en busca de una entrada apropiada. No hay una entrada de ruta explícita para la red 192.168.0.0, así que *cuke* utiliza su pasarela predeterminada. El datagrama IP dirigido a *patton* se pone en un marco Ethernet dirigido a *fedex*.

Después de recibir el marco Ethernet, *fedex* comprueba el datagrama IP adjunto y ve que está dirigido a 192.168.0.12. La red 192.168.0.0 no está unida directamente a *fedex*, por lo que mira en su tabla de encaminamiento para ver dónde tiene que enviar el datagrama. No hay una entrada de ruta explícita, así que *fedex* utiliza su pasarela predeterminada, *usps*. El datagrama IP se pone en un marco Ethernet dirigido a *usps* y se envía a través de la interfaz de red apropiada.

Cuando *usps* recibe el marco Ethernet, comprueba el datagrama IP y descubre que está dirigido a 192.168.0.12. La red 192.168.0.0 está unida directamente, por lo que *usps* incluye el datagrama IP en un marco Ethernet dirigido a *patton* y lo envía a través de la interfaz de red apropiada.

Después *patton* recibe el marco Ethernet y descubre que el datagrama IP adjunto está dirigido a sí mismo. Después de manipular el datagrama IP, si hay que enviar una respuesta, *patton* sigue la misma serie de pasos para enviarla.

Dejamos el trazado de la ruta de retorno del datagrama IP como ejercicio.

Encaminamiento por grados

Todo encaminamiento en un *host* se lleva a cabo de acuerdo con la tabla de encaminamiento de ese *host*. En esta tabla existen tres tipos básicos de entrada, correspondiéndose con tres clases de encaminamiento: redes unidas localmente (encaminamiento directo), rutas asignadas estáticamente (encaminamiento estático) y rutas asignadas dinámicamente (encaminamiento dinámico). En las siguientes secciones los estudiaremos por separado.

Encaminamiento directo

Cuando dos *hosts* están en la misma red y subred, no es necesario ningún encaminador para pasar tráfico entre ellos. Los datagramas IP se envían entre ellos utilizando marcos de capa de enlace dirigidos directamente.

Ésta es la forma de encaminamiento más sencilla disponible. Las entradas se añaden automáticamente a una tabla de encaminamiento de *hosts* cuando se agrega una red nueva con el comando `/sbin/ifconfig`.

Encaminamiento estático

En redes sencillas, o porciones sencillas de redes combinadas, las entradas de encaminamiento pueden introducirse a mano en una tabla de encaminamiento de *host* con el comando `/sbin/ifconfig`. Normalmente, esto se hace sólo cuando no es probable que la ruta cambie, porque cualquier cambio debe introducirse también a mano. La entrada de pasarela predeterminada en una tabla de encaminamiento es casi siempre una entrada de encaminamiento estático.

Encaminamiento dinámico

Cuando es probable que las rutas entre redes cambien, es más apropiado el encaminamiento dinámico. Se basa en un protocolo de encaminamiento subyacente como RIP-2, OSPF o BGP para pasar información sobre las rutas a las redes entre encaminadores. En el Capítulo 5, “Protocolos de la capa de aplicación”, veremos con más detalle RIP-2.

El encaminamiento dinámico es la clave de una capacidad de internet combinada para sobrevivir perdiendo conexiones entre redes.. En la Figura 3.4 podemos ver un ejemplo sencillo.

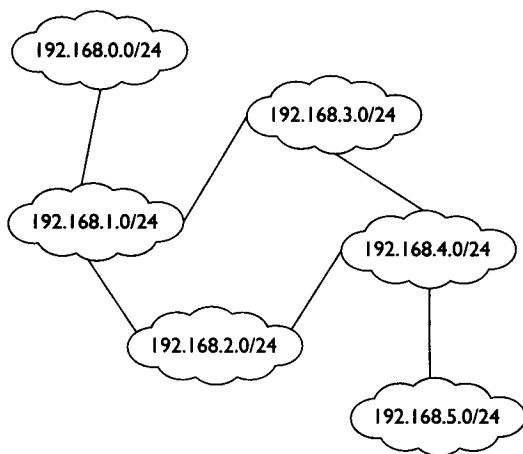


Figura 3.4. Una internet combinada pequeña.

En esta internet, la red 192.168.0.0/24 puede enviar tráfico a la red 192.168.5.0/24 a través de la red 192.168.2.0/24 o de la red 192.168.3.0/24 (sin embargo, todo el tráfico fluirá a través de la red 192.168.1.0/24). El tráfico fluirá normalmente por esta ruta:

```
192.168.0.0/24 a
192.168.1.0/24 a
192.168.2.0/24 a
192.168.5.0/24
```

El tráfico encaminado a través de la red 192.168.3.0/24 necesitará un salto adicional ya que pasa a través de la red 192.168.4.0/24. Este salto adicional

se refleja en las entradas de la tabla de encaminamiento pasadas entre los encaminadores.

Si la conexión entre 192.168.1.0/24 y 192.168.2.0/24 cayera por alguna razón, las tablas de encaminamiento se actualizarían para mostrar que la mejor ruta nueva era como sigue:

```
192.168.0.0/24 a
192.168.1.0/24 a
192.168.3.0/24 a
192.168.4.0/24 a
192.168.5.0/24
```

Aunque es más larga que la ruta antigua, aún funciona y pasa a ser la mejor ruta. Cuando se recupera el enlace entre 192.168.1.0/24 y 192.168.2.0/24, la tabla de encaminamiento volverá a la ruta más corta a través de la red 192.168.2.0/24.

Cómo convertir nuestro equipo Linux en un encaminador

Todos los equipos Linux son capaces de encaminar tráfico IP. Esta capacidad está desactivada por omisión en la mayoría de las distribuciones Linux. Un parámetro de núcleo llamado `ip_forward` controla esta funcionalidad. El siguiente comando permitirá el encaminamiento en nuestro sistema:

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

Si nuestro equipo está conectado a una o más redes IP, comenzará a encaminar el tráfico entre ellas. Desde luego, encaminará sólo el tráfico que se le envíe como encaminador, por lo que deberíamos informar a los *hosts* de las redes conectadas de que este encaminador está ahora activo. Para una red sencilla, esto probablemente signifique la adición de nuestro encaminador como la pasarela predeterminada de nuestros *hosts* individuales. Para una red más implicada, iniciaremos los demonios `routed`, `gated` o `zebra` y dejaremos que anuncien nuestro nuevo encaminador. Para más información sobre estos programas, véanse los Capítulos 5 y 7.

Trucos más avanzados

El núcleo Linux 2.4 incluye la capacidad de ir más allá de este encaminamiento sencillo. Algunas de las capacidades de encaminamiento avanzadas con el nuevo núcleo incluyen encaminamiento basado en el tipo de servicio, reencaminamiento de tráfico basándose en el protocolo de la capa de aplicación, filtrado de paquetes, traducción de direcciones de red, formación de tráfico y otros trucos.

Filtrado de paquetes

El *filtrado de paquetes* es el proceso de determinar si encaminar, aceptar o denegar un paquete basándose en la información del encabezamiento de

paquete. La forma más sencilla de filtrado de paquetes se basa en una comparación de las direcciones IP de origen y de destino (véase la Figura 3.5).

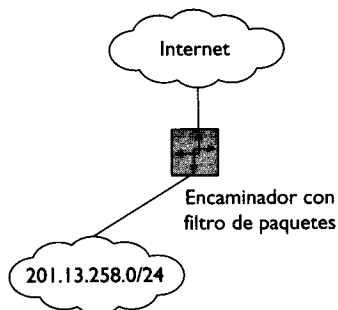


Figura 3.5. Una red pequeña.

Dada una red como la que mostramos en la Figura 3.5, el filtro de paquetes podría configurar normas para bloquear tráfico como éste (utilizando un sencillo pseudo código de configuración explicado aquí).

```
deny 10.0.0.0/24 201.13.25.0/8
deny 172.16.0.0/20 201.13.25.0/8
deny 192.168.0.0/16 201.13.25.0/8
deny 127.0.0.1/24 201.13.25.0/8
deny 201.13.25.0/8 201.13.25.0/8
allow 0.0.0.0/32 201.13.25.100/0
deny all
```

Cada norma es de esta forma:

```
action source/mask destination/mask
```

Las máscaras de estas normas no son máscaras de subred; cada bit representa un bit cambiable, comenzando por el bit menos significativo de la dirección. Por ejemplo, la combinación destino-máscara 201.13.25.0/8 representa a todas las direcciones desde 201.13.25.0 a 201.13.25.255 (cualquier bit del último octeto puede establecerse en cualquier valor).

Las cinco primeras normas niegan las conexiones de tres clases de tráfico: las direcciones privadas del RFC 1918 (explicadas anteriormente), las direcciones de bucle de retorno locales y la red interna (para evitar tráfico falseado). La sexta norma permite el tráfico desde cualquier *host* (que no esté ya denegado) que esté dirigido a 201.13.25.100. La última norma niega todo lo demás⁷.

Netfilter e iptables proporcionan las capacidades de filtrado de paquetes de Linux. Explicaremos estas herramientas en su propia sección en el Capítulo 11.

⁷ Estas normas no son adecuadas para asegurar una red. Las damos sólo como ejemplos.

Traducción de direcciones de red

La NAT (*Network Address translation*, Traducción de direcciones de red) es el método para hacer correspondencia de un espacio de direcciones privadas internas (normalmente compuesto de direcciones del RFC 1918) con una o más direcciones IP encaminables. Esta funcionalidad puede ayudar a ofrecer seguridad (ocultando nuestra estructura de dirección interna y evitando la conexión con nuestros *hosts* internos) y a minimizar el coste adicional de direcciones IP (utilizando un espacio de direcciones privadas en lugar de tener que adquirir direcciones IP encaminables de nuestro ISP).

El *enmascaramiento* es una forma común de NAT. El enmascaramiento IP hace correspondencia entre cualquier espacio de direcciones internas con una dirección IP única. Esto se utiliza normalmente para permitir que varios PC de una casa u oficina pequeña compartan una única conexión a Internet por marcación, DSL o módem-cable.

IPv4

El IP (*Internet Protocol*, Protocolo Internet) se define en el RFC 791. IP proporciona direcciones encaminables, fragmentación y reagrupación y opciones de entrega basadas en el tipo de servicio (TOS).

Estudio de IPv4

Normalmente, el encabezamiento IPv4 tiene 20 bytes de longitud. Contiene un campo de Versión de 4 bits (siempre será 4), un campo de Longitud de encabezamiento de 4 bits (normalmente será 5, indicando cinco palabras de 4 bytes cada una, con un total de 20 bytes), un campo de TOS de 1 byte, un campo de TTL (*Total Length*, Longitud total) de 2 bytes, un campo de Identificación de 2 bytes, un campo de Indicadores de 3 bits, un campo de Desplazamiento de fragmentos de 13 bits, un campo de TTL (*Time to Live*, Tiempo de vida) de 1 byte, un campo de Protocolo de 1 byte, un campo de Suma de verificación de encabezamiento de 2 bytes, un campo de Dirección de origen de 4 bytes y un campo de Dirección de destino de 4 bytes.

TOS

El byte de TOS está dividido en tres secciones. En la primera sección, se utilizan 3 bits para un campo de precedencia, que se utiliza poco hoy en día. En la segunda sección, se utilizan 4 bits para indicar TOS y sólo puede esta-

blecerse uno de ellos. Los cuatro indicadores de TOS posibles son *delay*, *throughput*, *reliability* y *cost*. Si no se activa ninguno de estos bits se producirá un encaminamiento normal. La tercera sección es un espacio reservado de 1 bit; este bit está reservado y debe establecerse en 0.

Los bits de TOS permiten un mejor control sobre cómo se encaminará un paquete. Una aplicación interactiva como *ssh* debería establecer el bit de retraso (*delay*), indicando que los paquetes deberían encaminarse para minimizar el retraso. Una aplicación que esté transfiriendo datos debería establecer el bit de rendimiento (*throughput*) para obtener un encaminamiento dirigido hacia la maximización del rendimiento. Estas opciones se explican más exhaustivamente en el RFC 1349. Además de encaminamiento TOS, el núcleo Linux 2.4 aporta herramientas potentes para dar forma al tráfico de red basándose en un cierto número de posibles claves; estas herramientas se explicarán en el Capítulo 11.

Longitud total

El campo de Longitud total indica la longitud total del datagrama IP (incluido el encabezamiento) en bytes. IP está limitado a 65535 bytes, 65515 de los cuales pueden cargarse. Si el datagrama IP se fragmenta (véase la sección “Indicadores de fragmentación”, un poco más adelante), el campo de Longitud total indica el tamaño del fragmento. En la práctica, TCP y UDP limitan el tamaño de sus datos para evitar enviar paquetes extremadamente grandes (8192 bytes es un tope normal).

Identificación

Cada datagrama de un *host* está identificado de manera única por el campo de Identificación de 2 bytes. Esta identificación no cambia entre fragmentos de un único datagrama, lo que permite que éste se reagrupe en el destino.

Indicadores de fragmentación

El campo de Indicadores de 3 bits contiene dos indicadores que tratan de la fragmentación y un bit reservado. El primer bit está reservado y debe ser 0. El segundo bit es el indicador de No fragmentar. Si está establecido este bit, indica que no se debería fragmentar este bit; en su lugar, debería generarse un mensaje de error de ICMP (*Internet Control Message Protocol*, Protocolo de mensajes de control de Internet)⁸. El tercer bit es el indicador de Último fragmento. Este bit se establece cuando no hay más fragmentos de este datagrama.

⁸ Esto hace que el bit de No fragmentar sea útil para las herramientas de diagnóstico.

Campo de Desplazamiento de fragmentación

Si el datagrama no se ha fragmentado, el campo de Desplazamiento de fragmentación se establecerá en todo ceros. Si se ha fragmentado, el campo indicará el número de palabras de 8 bytes que separan a este fragmento del inicio del datagrama. En la sección “Fragmentación IP” aparece un ejemplo de fragmentación.

TTL

El campo TTL indica el número de saltos que puede hacer el paquete antes de descartarse. Cuando se crea un paquete, se le da un TTL de 64. Después, cada vez que pasa por un encaminador, el TTL disminuye en 1. Esto evita que los paquetes que no se pueden entregar vaguen para siempre por la red.

Protocolo

El campo Protocolo indica qué protocolo va en la carga útil del datagrama IP. Nos ocuparemos de tres posibles valores: 0x06 (TCP), 0x11 (UDP) y 0x01 (ICMP). Trataremos estos protocolos en el siguiente capítulo.

Suma de comprobación IP

La suma de comprobación se calcula sólo con los datos del encabezamiento; nada de la carga útil se utiliza en el cálculo. El campo de Suma de comprobación se llena inicialmente con ceros.

Direcciones de origen y de destino

Los campos de Direcciones de origen y de destino contienen las direcciones IP de los *hosts* remitente y objetivo. Estos campos no se cambian en el flujo de datos IP a través de una red, excepto en el caso de datagramas encaminados de origen (véase la Tabla 3.4). Por seguridad, el encaminamiento de origen normalmente ya no se utiliza.

Opciones IP

Los encabezamientos IP excederán de 20 bytes cuando se incluyan opciones IP. En este caso, el campo de Longitud de encabezamiento se volverá a poner en el valor 0xf y se harán disponibles los siguientes campos adicionales: Copiar a través de puerta (1 bit), Clase de opción (2 bits) y Número de opción (5 bits). El campo de Copiar a través de puerta indica si estas opciones IP deberían añadirse a cada fragmento de un paquete IP (cuando se establece en 1) o no (cuando se establece en 0). El campo Clase de opción tiene dos valores no reservados, 00 indica códigos de control de tráfico y 10 indica códigos de depuración y de medida. Algunas de las opciones se muestran (con breves explicaciones) en la Tabla 3.4.

Tabla 3.4. Opciones IP.

Clase	Código de opción	Nombre de opción	Longitud	Descripción
00	00000	Fin de lista de opciones	—	Indica que no hay más opciones en el campo de opciones.
00	00010	Manipulación de seguridad	11	Define cómo hay que manipular un paquete (en un entorno militar). Véase RFC 1108 para más detalles.
00	00011	encaminamiento de origen flexible	Variable	Indica que un paquete debería atravesar un conjunto de encaminadores especificado. También pueden atravesarse otros encaminadores.
10	00100	Recoger marcas de tiempo de Internet	Variable	Indica que un paquete debería ser marcado con la hora actual de cada <i>host</i> que atraviese.
00	00111	Ruta de registro	Variable	Indica que un paquete debería marcarse con la dirección IP de cada <i>host</i> que atraviese.
00	01001	Encaminamiento de origen estricto	Variable	Indica que un paquete debería atravesar un conjunto de encaminadores especificado. Sólo pueden atravesarse los encaminadores enumerados.

Fragmentación IP

Los datagramas IP pueden ser mucho más grandes que la MTU de la ruta que sigue el datagrama cuando se entrega. La Figura 3.6 muestra un ejemplo de una red pequeña en la que podría ocurrir esto.

La red Token Ring tiene una MTU de 4096, la Ethernet tiene una MTU de 1500 y la conexión PPP tiene una MTU de 576.

Si se envía un datagrama IP de 4096 por Token Ring desde Dexter a Wally en la Ethernet remota, se fragmentará en tres datagramas nuevos (cada uno de ellos con el mismo número de ID) cuando pase por el encaminador 1 en la Ethernet local. Los dos primeros datagramas tienen 1500 bytes de largo y el tercero 1156. Recordemos que cada uno lleva un encabezamiento IP, así que hemos añadido 40 bytes (para los dos encabezamientos IP nuevos), más el relleno para alcanzar la longitud de palabra de 8 bytes necesaria añadido al coste adicional del datagrama inicial.

Cuando se trasladan desde la Ethernet local al enlace PPP a través del encaminador 2, cada datagrama IP se vuelve a fragmentar. Los dos primeros datagramas se fragmentan en tres nuevos datagramas cada uno: 576, 576 y 388 bytes. El tercer datagrama se ha fragmentado ahora en tres datagramas

también: 576, 576 y 44 bytes. Estos nueve datagramas representan 160 bytes adicionales de encabezamiento IP y cierta cantidad de relleno.

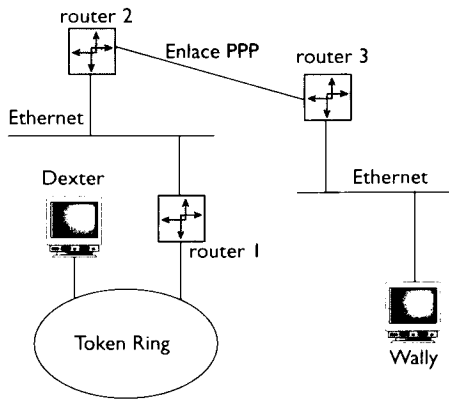


Figura 3.6. Fragmentación IP en una red pequeña.

Cuando los nueve datagramas pasan a través del encaminador 3 en la Ethernet remota, no se reagrupan. La reagrupación del datagrama inicial se produce solamente en el punto final de la transmisión.

El ancho de banda adicional utilizado por los datagramas fragmentados, así como la utilización de CPU necesaria para fragmentar y reagrupar los paquetes, hacen que el descubrimiento de la MTU de ruta inicial sea una opción deseable.

IPv4 descodificado

En el datagrama IP que se muestra en la Figura 3.7, la versión IP se establece en 4. La longitud de encabezamiento se establece en 5, lo que indica que no hay encabezamientos. Los indicadores Precedencia y TOS se establecen en 0x00, lo que significa que no se establece ningún indicador de manipulación especial. (Esto se etiqueta como “Servicios diferenciados” en la captura de pantalla Ethereal.) La longitud total se establece en 0x003c (o 64 bytes). El ID de datagrama se establece en 0x0054. El campo de Indicadores se establece en 010, indicando que está establecido el indicador de No fragmentar. Como no hay fragmentación, el desplazamiento de fragmentos es cero. El TTL es 0x40 (64), mostrando que este datagrama no ha pasado a través de ningún encaminador. El Protocolo se establece en 0x06 (TCP). La Suma de verificación es 0x3c66, lo que es correcto para este datagrama. La Dirección de origen es 127.0.0.1 y la Dirección de destino es 127.0.0.1. Este

paquete era parte de una conexión TCP de la interfaz de bucle de retorno de un *host*.

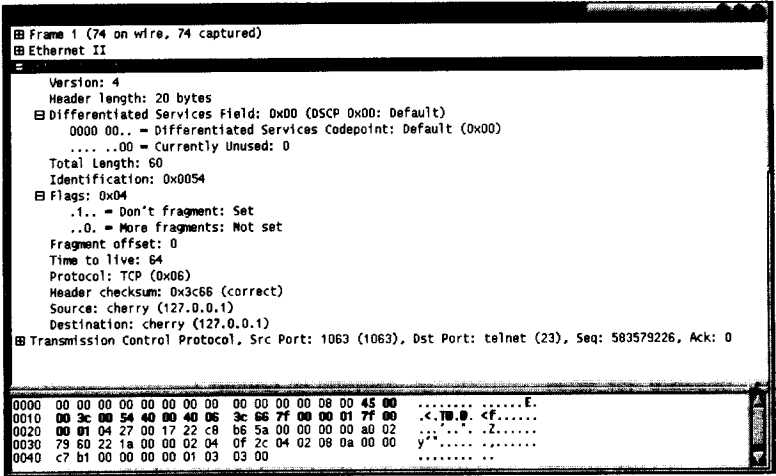


Figura 3.7. Datagrama IP.

4

Protocolos de la capa de transporte

La capa de transporte traslada datos entre *hosts* para la aplicación que tiene por encima e información de diagnóstico sobre las conexiones entre los *hosts* de la red. Esta información se traslada a través de TCP (y el T/TCP relacionado), UDP o ICMP. Estos protocolos son bastante diferentes en funciones y apariencia (como veremos en sus secciones respectivas). TCP garantiza un flujo de datos fiable, permitiendo que la implementación de la aplicación ignore dichos detalles. UDP no garantiza la entrega, lo que fuerza a la aplicación a garantizar la fiabilidad. ICMP no traslada los datos según el sentido tradicional, pero los traslada por la red o por otras conexiones de la red. En este capítulo explicaremos los puertos y los *sockets*, TCP, UDP e ICMP.

Puertos y *sockets*

En la capa de transporte hacemos referencia a los protocolos de la capa superior mediante sus números de puerto. Son números de 2 bytes que se corresponden con una aplicación o proceso en particular. Algunos números de puerto están reservados o registrados y están recogidos en el RFC 1700. El acceso a dichos puertos está controlado frecuentemente por un *demonio* como *inetd* (*http* en el puerto 80 es una excepción común a esta norma). Otros puertos se asignan a una aplicación “al vuelo” (los servidores o clientes se pueden adherir a un puerto de esta forma).

inetd (y otros demonios similares) actúan como la “caja de conmutadores” de un *host*. Este proceso está configurado para escuchar en múltiples puertos y después captar un proceso servidor del tipo apropiado para cada conexión entrante. *inetd* es susceptible a varios tipos de ataques y normalmente es reem-

plazado o suplementado por otras herramientas. *xinetd* y los *TCP wrappers* son ejemplos de servidores de reemplazo y suplemento, respectivamente, y los explicamos en el Capítulo 11, “Herramientas de seguridad”.

Los puertos reservados son aquéllos que tienen un valor del 1 al 1023 (ambos incluidos). La utilización de estos puertos está restringida a los procesos iniciados por la raíz. Estos puertos se asignan a las aplicaciones a través de la IANA (*Internet Assigned Numbers Authority*, Autoridad de Números Asignados de Internet). Como la utilización de estos puertos está controlada por un núcleo de estándares, a veces se les llama “puertos bien conocidos”.

Los puertos del intervalo de 1024 a 65535 (inclusive) se llaman puertos registrados si están enumerados en el RFC 1700. Aunque están enumerados por la IANA, no están estandarizados. Un proceso servidor puede adjuntarse a cualquiera de los puertos de este intervalo, pero consideramos una buena práctica intentar evitar la utilización de números reservados utilizados con frecuencia.

También los procesos clientes utilizan este intervalo de puertos. Por ejemplo, cuando un usuario hace una conexión Telnet con un *host* remoto, el cliente Telnet local se une a un puerto del intervalo no restringido, por ejemplo, el 1027. El cliente Telnet intenta entonces hacer una conexión con el servidor en el puerto 23. Como el acceso Telnet se manipula normalmente con *inetd*, recibe la petición entrante e inicia un proceso *in.telnetd* para tratar con la sesión Telnet.

Puertos utilizados normalmente

Las Tablas 4.1, 4.2 y 4.3 (extraídas del archivo */etc/services*) representan servicios utilizados de forma muy frecuente. Algunos se ofrecen a través de TCP y UDP, otros no.

Tabla 4.1. Servicios proporcionados internamente en la pila IP.

Puerto/Protocolo	Nombre de servicio
7/tcp	echo
7/udp	echo
9/tcp	discard
9/udp	discard
11/tcp	systat
13/tcp	daytime
13/udp	daytime
15/tcp	netstat
17/tcp	qotd
19/tcp	chargen
19/udp	chargen
37/tcp	time
37/udp	time

Tabla 4.2. Servicios proporcionados por demonios externos en los puertos restringidos.

Puerto/Protocolo	Nombre de servicio
20/tcp	ftp-data
21/tcp	ftp
22/tcp	ssh
22/udp	ssh
23/tcp	telnet
25/tcp	smtp
53/tcp	domain
53/udp	domain
67/tcp	bootps
67/udp	bootps
68/tcp	bootpc
68/udp	bootpc
69/udp	tftp
79/tcp	finger
80/tcp	www
80/udp	www
88/tcp	kerberos
88/udp	kerberos
98/tcp	linuxconf
110/tcp	pop3
110/udp	pop3
111/tcp	sunrpc
111/udp	sunrpc
113/tcp	auth
137/tcp	netbios-ns
137/udp	netbios-ns
138/tcp	netbios-dgm
138/udp	netbios-dgm
139/tcp	netbios-ssn
139/udp	netbios-ssn
161/udp	snmp
162/udp	snmp-trap
220/tcp	imap3
220/udp	imap3
389/tcp	ldap
389/udp	ldap
443/tcp	https
443/udp	https
512/tcp	exec
513/tcp	login
513/udp	who
514/tcp	shell
514/udp	syslog
520/udp	route
543/tcp	klogin
544/tcp	kshell
636/tcp	ssl-ldap
873/tcp	rsync
873/udp	rsync

Tabla 4.3. Servicios proporcionados por demonios externos en los puertos registrados.

Puerto/Protocolo	Nombre de servicio
2401/tcp	cvspserver
2401/udp	cvspserver
3306/tcp	mysql
3306/udp	mysql
4559/tcp	hylafax
5308/tcp	cfengine
5308/udp	cfengine
5432/tcp	postgres
5432/udp	postgres
6667/tcp	ircd
6667/udp	ircd
10080/udp	amanda
10081/tcp	kamanda
10081/udp	kamanda
10082/tcp	amandaidx
10083/tcp	amidxtape

¿Qué es un *socket*?

Como un número de puerto no es de mucha ayuda de forma aislada, normalmente se les pone en contexto como *sockets*, que son una dirección IP con un número de puerto. Así, una conexión Telnet de cherry a mango podría tener los siguientes *sockets*:

Lado del cliente
 Socket cherry
 192.168.1.10:1027

Lado del servidor
 Socket mango
 192.168.1.1:23

TCP

TCP (*Transmission Control Protocol*, Protocolo de Control de Transmisión), definido en el RFC 793 y clarificado en el RFC 1123, está más implicado que UDP, pero con el coste añadido de la ejecución de TCP obtenemos un conjunto de características y una fiabilidad mayores. Las sesiones dadas por TCP son la causa de gran parte de la sobrecarga del protocolo, pero ofrecen muchos de los beneficios de TCP.

Las sesiones suministran una conexión continua entre dos procesos, un mecanismo para el seguimiento de la cantidad de datos presentes en una conexión, un método para la difusión de los datos perdidos y una forma de acusar recibo de los paquetes que se han recibido. Proporcionan un sistema de control de congestión sin necesidad de que la aplicación lo manipule.

También controla el tamaño de los paquetes que se envían y mantiene un flujo ordenado de paquetes que vuelven a la aplicación, incluso pudiendo recibir paquetes fuera de orden (o duplicados) de la capa IP.

Un estudio de TCP

En nuestra explicación de TCP, cubriremos la estructura del encabezamiento TCP, el inicio y finalización de la sesión y T/TCP. Cada una de estas secciones se construye a partir de la información contenida en las anteriores, así que tenemos que leerlas en orden.

El encabezamiento

Un encabezamiento TCP se compone de los siguientes campos: un número de Puerto de origen de 2 bytes, un número de Puerto de destino de 2 bytes, un número de Secuencia de 4 bytes, un número de Acuse de recibo de 4 bytes, un Tamaño de encabezamiento de 4 bits, un campo Reservado de 6 bits, una sección Indicador de 6 bits, un Tamaño de ventana de 2 bytes, una Suma de comprobación de 2 bytes, un Puntero de datos urgentes de 2 bytes y opciones.

En todos los segmentos TCP se llevan los números de puerto de los procesos de origen y de destino y se define el *socket* de cada lado de la conexión, como describimos en la sección anterior, “Puertos y *sockets*”.

El número de Secuencia identifica el primer byte de datos incluido en este segmento. Los bytes de una corriente TCP no se cuentan desde cero; en lugar de ello, se cuentan desde un punto casi aleatorio sobre el que se ha llegado a un acuerdo al comienzo de la sesión TCP. (Definimos este proceso en la siguiente sección, “Comienzo y finalización de sesiones”).

El número de Acuse de recibo está unido a la secuencia del otro extremo de la conexión. Representa el siguiente byte que este equipo está esperando que llegue del otro.

El Tamaño de encabezamiento es el número de palabras de 4 bytes utilizado por el encabezamiento TCP. Normalmente, el encabezamiento es de 20 bytes, dando un valor de 4 bits de 5. Frecuentemente se utilizan las opciones en el comienzo de la sesión TCP. Cuando están presentes, el Tamaño de encabezamiento está establecido como 6 (24 bytes) y se establece una Opción de 0x0204xXXXX. El XXXX es el número de 2 bytes correspondiente al MSS (*maximum segment size*, Tamaño de segmento máximo) permitido por el *host* remitente. Si no se utiliza esta opción, el MSS se establece como 536 (para un total de 576 bytes, después de añadir 20 para cada uno de los encabezamientos IP y TCP estándar). Se hace algún uso de las opciones para mejorar el rendimiento en redes muy cargadas (LFN). Otras opciones son *Noop* 0x01 y la Lista de final de opción (*end-of-option List*) 0x00; cada una de ellas es de sólo 1 byte.

TCP EN LFN

La capacidad de conexión es el producto del ancho de banda (en bits por segundo) y del tiempo de viaje (en segundos). Como este número es muy grande en las WAN modernas (más de 320.000.000 bits, o 40.000.000 bytes, para una OC3 entre Nueva York y California), el tamaño de ventana no es capaz de manipular de forma efectiva el control de flujo. Hay dos opciones para ayudar a evitar este problema.

La primera, llamada escalado de ventana, cambia el tamaño de la ventana TCP de 2 a 4 bytes. Esto se hace ensanchando la ventana hasta 14 bits. La opción de escalado de ventana es 0x03030X, donde X es el tamaño del ensanchamiento que vamos a hacer). Sólo podemos llevar a cabo el escalado de ventana en los paquetes SYN y SYN-ACK del comienzo de la sesión TCP.

La segunda opción recae en el sello temporal (0x080aSSSSSSSSRRRRRRRR, donde SSSSSSSS es el sello temporal y RRRRRRRR es la respuesta al sello temporal), que ayuda a evitar que se coloquen datos duplicados en la sección incorrecta del *buffer*. Podemos utilizar el sello temporal durante una sesión TCP.

A continuación de los bits reservados hay una serie de indicadores de 1 bit utilizados por TCP:

- URG. Este segmento contiene datos urgentes.
- ACK. El número de acuse de recibo contiene un valor válido.
- PUSH. Solicita una manipulación más rápida del paquete.
- RST. Solicita una finalización expedita del proceso remoto porque el proceso local se está cerrando o se ha cerrado.
- SYN. Establece el número de secuencia inicial del *host*. Este bit se establece sólo durante el comienzo inicial de la sesión TCP.
- FIN. Este indicador solicita que se cierre la conexión de forma normal.

Explicamos los bits ACK, SYN y FIN en la siguiente sección, “Comienzo y finalización de sesiones”.

Utilizamos el campo Tamaño de ventana en conjunción con el número de Acuse de recibo para proporcionar un control de flujo. Muestra la cantidad de espacio de *buffer* disponible en el *host* remitente para los paquetes entrantes de esta sesión, comenzando por el número de Acuse de recibo. Funciona así:

Después de configurada la sesión (como describimos en la siguiente sección), cherry tiene un número de Acuse de recibo de 0x00000020 y un Tamaño de ventana de 0x0040 (64 bytes). Es un número anormalmente pequeño, pero lo utilizamos por razones de claridad en el ejemplo. mango envía un paquete de 17 bytes de datos y un segundo paquete de 20 bytes. De alguna forma, el primer paquete se pierde en el camino hacia cherry. Cuando cherry envía el siguiente paquete, todavía tiene un número de Acuse de recibo de 0x00000020 (porque no ha visto los primeros 17 bytes de mango), pero disminuye su Tamaño de ventana a 0x0020c.

Entonces, mango envía un paquete de 10 bytes a cherry, que responde con un número de Acuse de recibo de 0x00000020 y un Tamaño de ventana de 0x0022. mango vuelve a enviar su primer paquete y cherry responde con un número de Acuse de recibo de 0x00000046 (que refleja todos los datos

que ha recibido) y un Tamaño de ventana de 0x000040 (vuelve a estar disponible todo su *buffer*).

Este método de control de flujo que utiliza un cambio en el tamaño de ventana se llama ventana deslizante.

La Suma de comprobación TCP es una suma de comprobación obligatoria que se computa sobre un pseudoencabezamiento, el encabezamiento TCP, y en los datos transportados por el paquete. El pseudoencabezamiento contiene campos del encabezamiento IP para verificar que el paquete ha sido recibido por el *host* correcto. Los campos utilizados son la dirección IP de origen de 4 bytes, la dirección de destino de 4 bytes, un relleno de 4 bytes (todos 0), el protocolo de transporte de 1 byte y el tamaño de transporte de 2 bytes. Mostramos un diagrama de este pseudoencabezamiento en la Figura 4.1.

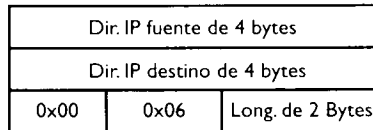


Figura 4.1. Un pseudoencabezamiento TCP.

Utilizamos el Puntero de datos urgentes para indicar el último byte de datos urgentes transportados en los datos TCP. Se utiliza para permitir la transmisión en banda de datos de emergencia. Si el datagrama contiene datos urgentes, también debe tener establecido un indicador URG. Estos datos urgentes representan frecuentemente una interrupción en el tráfico interactivo como el de Telnet, rlogin o FTP. Por ejemplo, la introducción de una interrupción para cambiar nuestras opciones de Telnet se haría con datos urgentes.

Comienzo y finalización de sesiones

Tanto el comienzo como la finalización de las sesiones imponen sobrecarga sobre la misma sesión. Esta sobrecarga permite que la sesión se administre de forma apropiada cuando haya comenzado y que se cierre apropiadamente cuando haya finalizado.

El protocolo de intercambio en tres pasos

El comienzo de una sesión TCP se llama protocolo de intercambio en tres pasos (o protocolo de intercambio). Se compone de los tres primeros segmentos TCP transmitidos. Durante este protocolo se intercambian y se acusa recibo de los números de secuencia iniciales y después las opciones de inicio.

Comienza cuando el cliente envía un datagrama al servidor. Este datagrama tiene establecido el indicador SYN y el valor inicial del número de secuencia del cliente. Si se van a utilizar una opción de escalado de ventana o MSS, estas opciones también deben establecerse en este datagrama. Normalmente a este datagrama se le llama datagrama o paquete SYN.

En el segundo paso del protocolo, el servidor envía un datagrama al cliente. Están establecidos tanto el indicador SYN como el ACK, se acusa recibo del número de secuencia del cliente (con un valor de número de secuencia incrementado en 1) y se establece el número de secuencia del servidor.

De nuevo, si se van a utilizar en esta sesión una opción de escalado de ventana o MSS, estas opciones se deben establecer en este datagrama. A este datagrama se le llama datagrama o paquete SYN-ACK.

El paso final del protocolo se dirige del cliente al servidor. En este datagrama, el cliente envía un indicador ACK, acusando recibo del número de secuencia del servidor (nuevamente con la cuenta incrementada en 1). Se le llama datagrama o paquete ACK.

Las herramientas netstat, nmap y ethereal (explicadas en los Capítulos 9, “Herramientas de resolución de problemas”, y 10, “Herramientas de revisión”) describen las sesiones TCP como si pasaran por varios estados. El protocolo de intercambio hace que la sesión TCP se traslade por varios de ellos. La siguiente sección revisa los pasos normales de un cliente y de un servidor. La Figura 4.2 muestra todos estos estados.

El cliente comienza en un estado CLOSED (cerrado) y pasa directamente al estado SYN_SENT (SYN enviado) cuando envía un paquete SYN. Cuando recibe el paquete SYN-ACK del servidor, pasa al estado ESTABLISHED (establecido), donde se mantiene hasta el cierre de la sesión.

El servidor comienza también en un estado CLOSED. Normalmente se une a un puerto y espera paquetes entrantes; a esto se le llama estado de LISTEN (escucha). Después de recibido el paquete SYN y enviado el paquete SYN-ACK, pasa al estado SYN_RCVD¹. Cuando el servidor recibe el paquete ACK del cliente, pasa al estado ESTABLISHED, donde se mantiene hasta el cierre de la sesión.

Cierre normal de la sesión

Para que una sesión se cierre normalmente se precisan cuatro paquetes. Cualquiera de los extremos de la conexión pueden solicitar el cierre de la sesión.

El cierre de la sesión comienza cuando uno de los equipos implicados en la misma envía un datagrama con los indicadores ACK y FIN establecidos. Normalmente se le llama paquete ACK-FIN. El segundo equipo responde con un datagrama en el que sólo está establecido el indicador ACK; se le llama paquete ACK. Como las sesiones TCP son en ambas direcciones, el segundo equipo podría continuar enviando paquetes, pero no es normal.

¹ Durante el estado SYN-RCVD, el servidor debe asignar un bloque de control de transmisión. Sólo un pequeño número de estas estructuras se envían aparte de la memoria, así que esto presenta una vulnerabilidad para el servidor. Los piratas que intentan desactivar un servidor atacan esta debilidad cuando intentan ejecutar un ataque de inundación SYN.

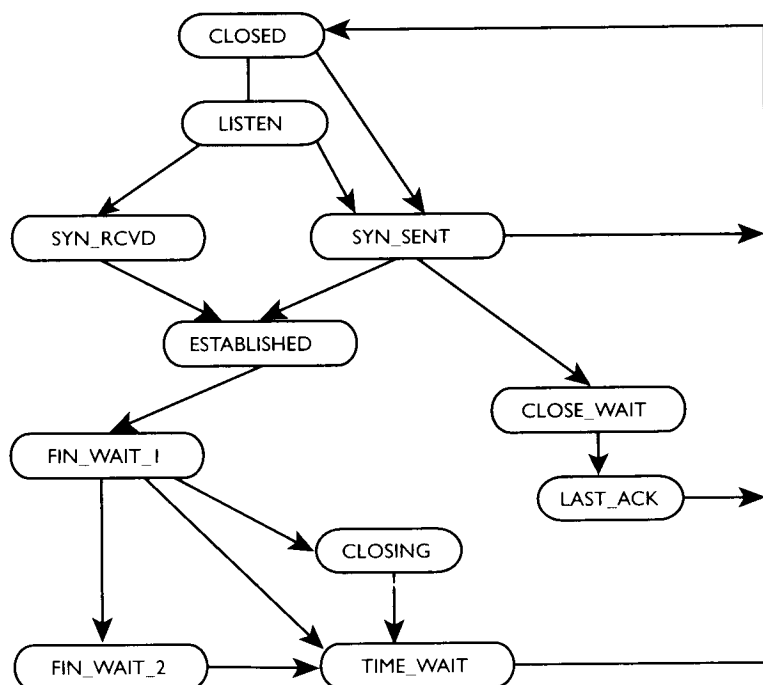


Figura 4.2. Diagrama de estado de una sesión TCP.

El cierre de la sesión queda completo cuando el segundo equipo envía un datagrama con los indicadores ACK y FIN establecidos y recibe la respuesta del primer equipo con el indicador ACK establecido. En este momento, ambas partes de la sesión están cerradas. Durante un cierre de sesión normal, la sesión TCP pasa por varios estados. Sigue una de dos rutas posibles por dichos estados, como describimos a continuación.

La primera ruta (y la más normal) de un cierre de sesión es la siguiente: en el equipo que cierra primero su parte de la sesión TCP, la sesión comienza en el estado ESTABLISHED. Cuando se envía el paquete ACK-FIN, pasa al estado FIN_WAIT_1. Cuando se recibe el paquete ACK, pasa al estado FIN_WAIT_2. Cuando recibe el ACK-FIN del equipo remoto, envía un paquete ACK y se introduce en el estado TIME_WAIT, donde permanece por el tiempo especificado por la implementación (el RFC 793 especifica 2 minutos, pero frecuentemente se utilizan otros valores).

Si el equipo remoto inicia el cierre normal de la sesión TCP, el equipo local pasa a un estado de CLOSE_WAIT después de recibir el paquete ACK-FIN y enviar su propio paquete ACK. Cuando el equipo local envía el paquete ACK-FIN, se introduce en el estado LAST_ACK y espera el paquete ACK del *host* remoto.

Es posible hacer algunas simplificaciones cerrando ambas partes de la sesión TCP al mismo tiempo. A esto se le llama cierre simultáneo. En este

caso, el primer *host* envía un paquete ACK-FIN, el segundo equipo responde con un paquete ACK-FIN y el primero envía un paquete ACK final. Esto reduce el número de datagramas necesarios para el cierre de la sesión de cuatro a tres y elimina el estado FIN_WAIT_2 de la serie de estados del primer equipo.

La Figura 4.2 muestra un diagrama de los estados por los que pasa una sesión TCP.

Interrupción de una sesión

Normalmente, las sesiones TCP se interrumpen (o abortan) por dos razones. O el cliente ha intentado conectar con un puerto inexistente o uno de los *host* implicados aborta manualmente la sesión.

Cuando el servidor recibe el paquete SYN del cliente que solicita una conexión con un puerto sin un proceso adjunto, el servidor responde con un datagrama que tiene establecido el indicador RST. El cliente debería entonces abandonar la sesión como si hubiera intentado crear una conexión con un puerto no disponible.

Cuando una de las dos partes de una conexión aborta una sesión, envía un datagrama con el indicador RST establecido. El otro equipo no envía respuesta; sencillamente aborta la sesión y notifica a la capa de aplicación que ha interrumpido la sesión.

Expiración del tiempo

Los temporizadores de sostenimiento no son parte de la especificación TCP, pero esto no hace que ciertas pilas de protocolo TCP/IP no los implementen. Generalmente, estos paquetes de sostenimiento se envían después de una espera de unas dos horas. Cuando se envía el paquete, pueden pasar tres cosas:

- El *host* remoto es inaccesible.
- El *host* remoto es accesible, pero la conexión está muerta.
- El *host* remoto es accesible y la conexión está todavía viva.

En el primer caso, el *host* local intentará paquetes de sostenimiento otras nueve veces, en intervalos de 75 segundos. Si no hay respuesta se cerrará la conexión. Observemos que el *host* remoto podría ser inaccesible porque se haya estropeado, se haya apagado, etc.; son causas razonables para cerrar una conexión. También podría ser inaccesible por un error transitorio de la red; ésta no es una razón demasiado buena para cerrar una conexión. El *host* local no tiene forma de determinar por qué no ha respondido el *host* remoto, así que a veces cierra paquetes que hubiera sido mejor dejar abiertos².

En el segundo caso, el *host* remoto responderá con un paquete RST y el local cerrará la conexión. Éste es un caso especial de puertos no disponibles (descritos anteriormente).

² Ésta es una de las razones por las que el exceso de tiempo no es parte de la especificación TCP.

En el tercer caso, el *host* remoto responderá con un paquete ACK y la conexión se mantendrá viva.

T/TCP

Como TCP impone una sobrecarga de siete paquetes para la configuración y el cierre de una sesión (y frecuentemente paquetes adicionales que llevan acuses de recibo), no está muy bien preparado para tráfico repetido y pequeño. En el RFC 1379 se define una solución que permite el tráfico transaccional. T/TCP (*Transactional TCP*, TCP Transaccional) permite que una única sesión TCP traslade múltiples transacciones. Lo hace añadiendo un campo CC (*Connection Count*, Cuenta de conexión) de 4 bytes al encabezamiento TCP. Cada transacción dentro de la sesión TCP utiliza una única CC, así que cada *host* debe hacer un seguimiento de la CC válida y actual del *host* remoto de la sesión.

TCP descodificado

El paquete capturado en la Figura 4.3 muestra un Puerto de origen 0x0413 (1043) y un Puerto de destino 0x1770 (6000). Los puertos de origen y de destino están en el intervalo disponible para los clientes, pero el de destino está en el puerto bien conocido (o registrado) del sistema X-11 y seguramente será el servidor de este intercambio. El equipo de origen tiene un número de Secuencia actual de 330802776 y está enviando un Acuse de recibo para el byte 327043426 (el siguiente byte que espera encontrar). El Tamaño de encabezamiento está establecido como 0x8, lo que representa 32 bytes de encabezamiento. A continuación está el relleno necesario y los indicadores como 0x18 (00011000 en binario), que representan ACK y PUSH. El Tamaño de ventana está establecido como 0x7900 (30976 bytes). La Suma de comprobación es 0x5e75.

Como el Tamaño de encabezamiento es de 32 bytes, sabemos que hay 12 bytes de opciones, 32 (tamaño del encabezamiento) - 20 (tamaño normal del encabezamiento) = 12 (bytes de encabezamiento). En este caso, las opciones son 0x00 (una NOP), 0x00 (otra NOP) y 0x010108a007b5cdb007b5cda (un sello temporal con un valor de 8084699 y una respuesta de sello temporal equivalente).

En la Figura 4.4 podemos ver el protocolo de intercambio en acción. Un usuario en cherry comienza la sesión Telnet con mango, mango replica y cherry responde a la réplica. Se intercambian estos tres paquetes antes de pasar cualquier dato Telnet (datos vivos o configurados).

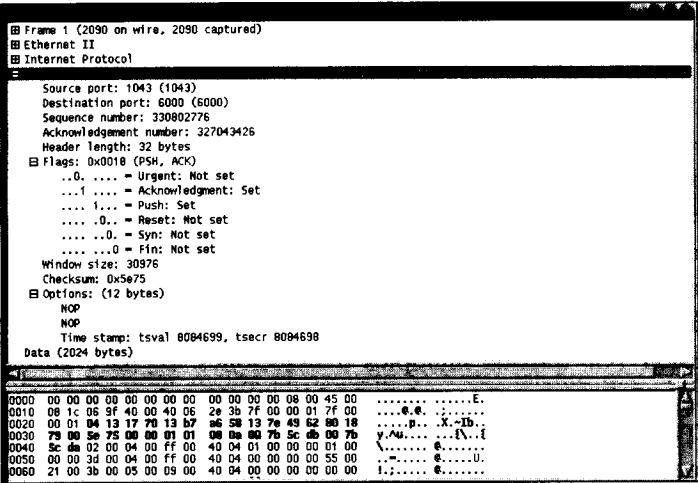


Figura 4.3. Un encabezamiento TCP.

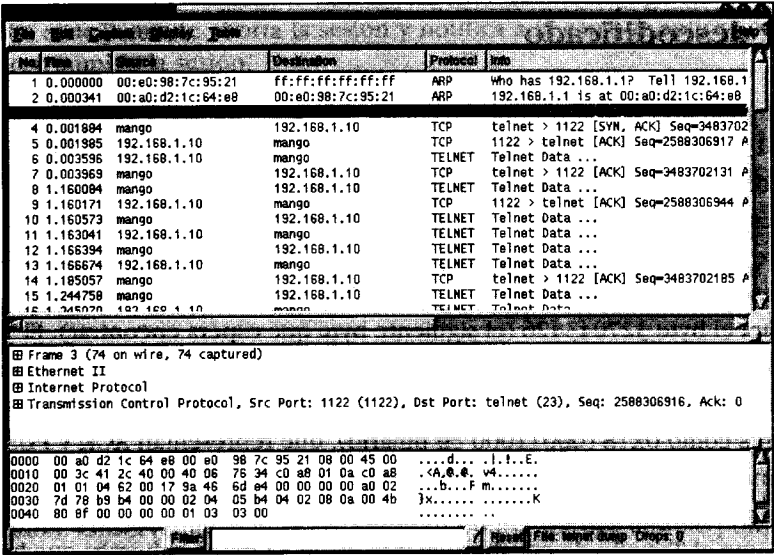


Figura 4.4. El protocolo de intercambio.

El primer paquete del protocolo de intercambio muestra a cherry estableciendo el bit SYN en el campo Opciones, con un número de Secuencia de 2588306916. En este momento no hay valor para el número de Acuse de recibo. Lo mostramos en la Figura 4.5.

El segundo paquete es de mango. Tiene establecidos tanto el bit ACK como el SYN. Tiene un número de Secuencia de 3483702130 y un número de Acuse de recibo de 2588306917. Observemos que el número de secuencia no tiene conexión con el número de secuencia utilizado por cherry; en

lugar de eso, es el número de acuse de recibo que refleja el número de secuencia del *host* remoto. mango utiliza un número de acuse de recibo 1 byte mayor que el número de secuencia de cherry, porque es el siguiente byte que espera recibir. Lo mostramos en la Figura 4.6.

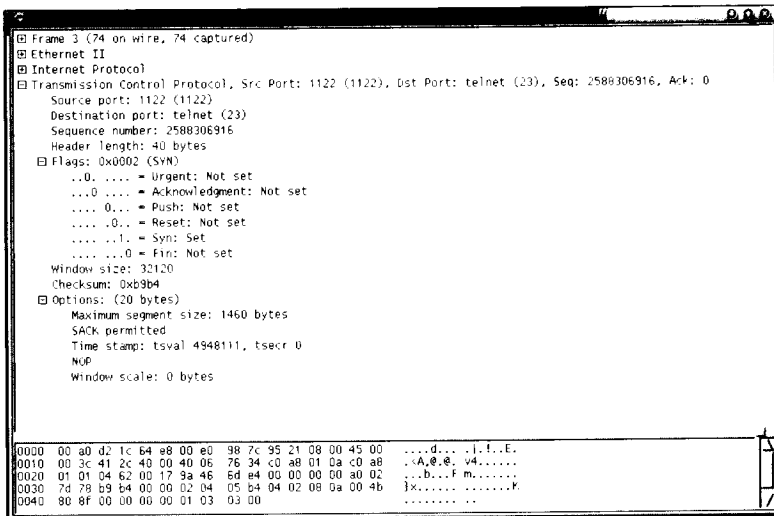


Figura 4.5. El paquete SYN del protocolo de intercambio.

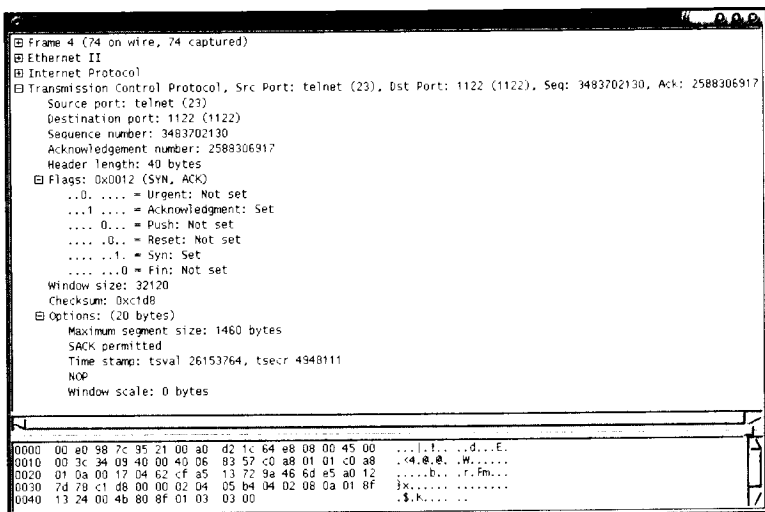


Figura 4.6. El paquete SYN-ACK del protocolo de intercambio.

En el paquete final del protocolo de intercambio (mostrado en la Figura 4.7), cherry ya no tiene establecido el bit SYN, pero tiene el ACK. Ahora ha incrementado su número de secuencia en 1, indicando que es el siguiente byte de datos que enviará. cherry establece su número de Acuse de recibo

como 3483702131, uno más que el número de secuencia de mango, mostrando el siguiente byte que espera encontrar.

Con el protocolo de intercambio apartado, los datos de la capa de aplicación pueden comenzar a pasar en los siguientes segmentos TCP. Esto continúa a lo largo de la captura de datos hasta que se termina la conexión.

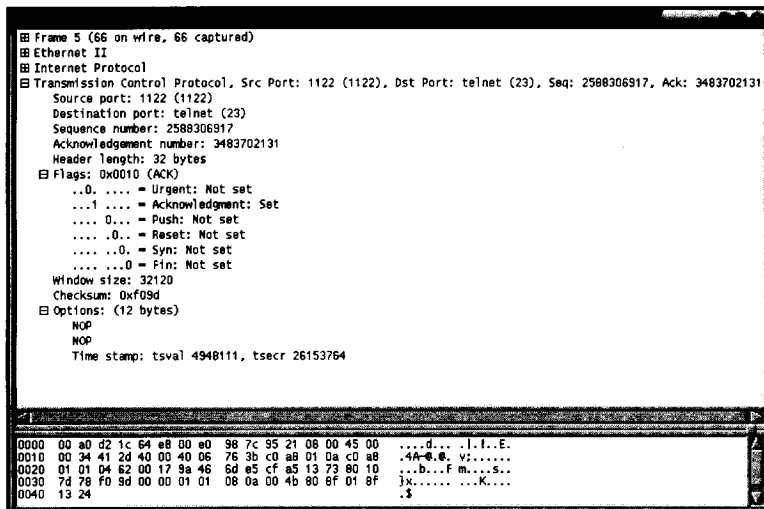


Figura 4.7. El paquete ACK del protocolo de intercambio.

En la Figura 4.8, que muestra una corriente TCP, podemos ver el comienzo de la finalización con el paquete resaltado. El cierre comienza cuando mango envía un paquete con los bits ACK y FIN establecidos. cherry responde con un paquete que tiene establecido el bit ACK. Después envía un paquete con los bits ACK y FIN establecidos.

La Figura 4.9 muestra la respuesta de mango, el paquete final con sólo el bit ACK establecido. Muestra también el primer paquete de cierre. Hemos resaltado los bits indicadores en esta captura de pantalla y podemos ver los bits ACK y FIN activados (el valor 0x11 corresponde a 0100001).

UDP

UDP (*User Datagram Protocol*, Protocolo de Datagrama de Usuario) está definido en el RFC 768. UDP proporciona un protocolo de transporte sencillo que utiliza datagramas para llevar los datos del origen al destino.

UDP ofrece una ventaja de velocidad frente a TCP al coste de algo de robustez. No hay sesión UDP, así que no hay comienzo de sesión y no hay sobrecarga de seguimiento de una ventana de datos. El encabezamiento tiene la mitad de tamaño que el encabezamiento de TCP. Como UDP no está orientado a la corriente como TCP, cada datagrama lleva un grupo de datos discreto para la aplicación de nivel superior (o la mayor parte de un grupo de datos discreto permitida por la ruta MTU).

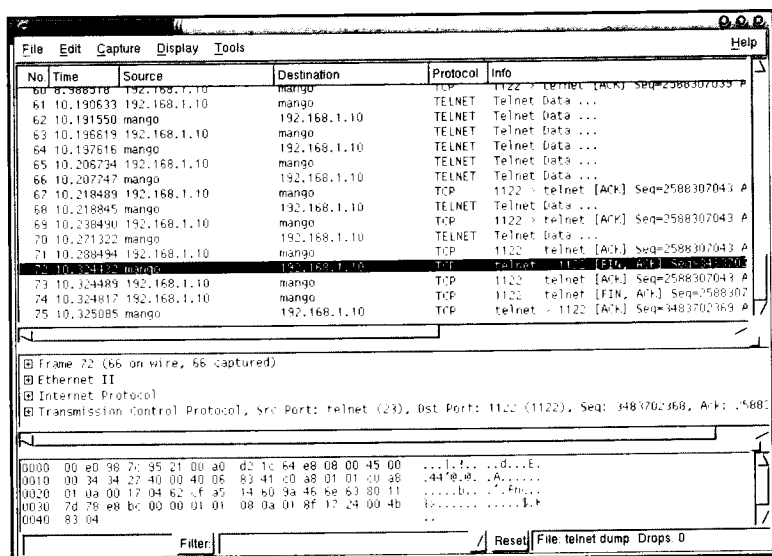


Figura 4.8. Un cierre de sesión de cuatro pasos.

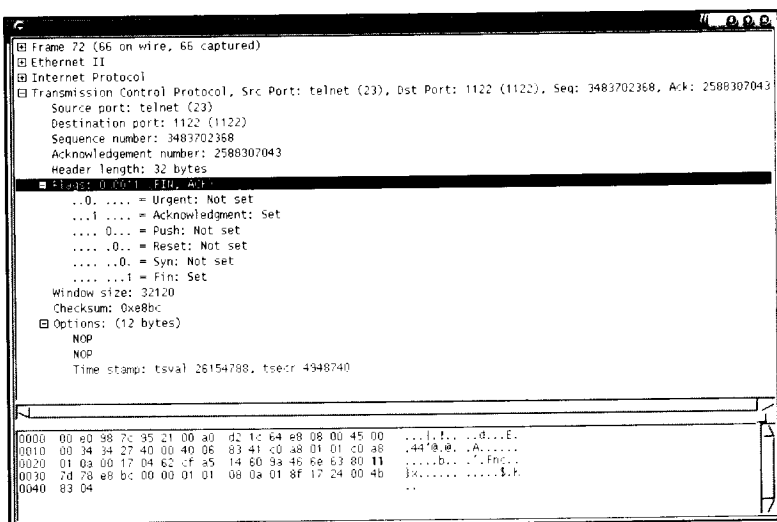


Figura 4.9. El primer paquete ACK-FIN.

Un estudio de UDP

El encabezamiento UDP está compuesto de cuatro campos, cada uno de 2 bytes: el Puerto de origen UDP, el Puerto objetivo UDP, el Tamaño del mensaje y la Suma de comprobación. Los puertos de Origen y Objetivo se utilizan para indicar el proceso utilizado en cada extremo de la conexión, como describimos anteriormente. El Tamaño del mensaje indica el número de bytes

del encabezamiento UDP y los datos trasladados por el datagrama: el valor mínimo es de 8 bytes.

La Suma de comprobación de UDP es opcional. Si no la calcula el remitente, se envía como un campo de todo ceros (0x0000). Cuando la utilizamos, se computa para los datos y para el pseudoencabezamiento construido como el pseudoencabezamiento TCP explicado en la sección anterior. Mostramos el pseudoencabezamiento en la Figura 4.10.

Dir. IP fuente de 4 bytes		
Dir. IP destino de 4 bytes		
0x00	0x11	Long. de 2 bytes

Figura 4.10. Pseudoencabezamiento UDP para la computación de la suma de comprobación.

Se precisa un relleno en al menos un lugar (y posiblemente dos) de los datos de la suma de comprobación. Hay un relleno de 1 byte entre la dirección IP de destino y el campo de Protocolo. Si hay un número extraño de bytes en la porción de datos del datagrama UDP, se necesita otro relleno de 1 byte para hacer que los datos de la suma de comprobación se ajusten a unos límites de 2 bytes.

UDP descodificado

El paquete UDP de la Figura 4.11 es una petición DNS. El Puerto de origen es 0x0402 (1042) y el Puerto de destino es 0x0035 (53 o DNS) El Tamaño es 0x0032 (50 bytes). La Suma de comprobación está en uso y tiene un valor de 0x590b.

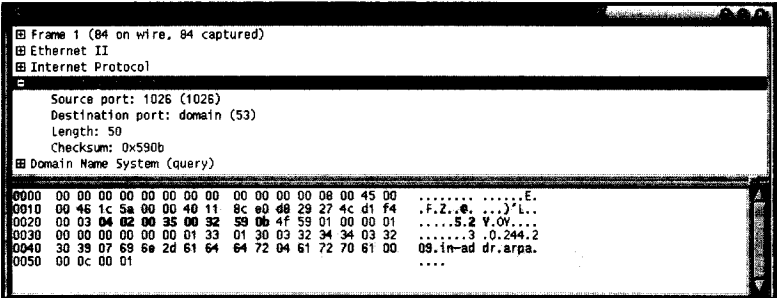


Figura 4.11. Un encabezamiento UDP.

Aunque UDP no tiene sesiones como TCP, es posible que aplicaciones basadas en UDP lleven a cabo transacciones. La red de la Figura 4.12 muestra un buen ejemplo. Observemos que el primer paquete es una petición genérica y que recibe una respuesta vacía. Las dos peticiones siguientes son para nombres de *host* distintos y reciben respuestas distintas.

ICMP

Mientras que TCP y UDP trasladan datos, el ICMP (*Internet Control Message Protocol*, Protocolo de mensajes de control de Internet) ofrece capacidades de informe de errores y diagnóstico. Debido a esta funcionalidad, ICMP es una importante herramienta para la comprensión de nuestras redes.

Revisión del protocolo

Los paquetes ICMP pueden generarse a partir de *hosts* IP cuando se encuentran ciertos errores o por una aplicación. Estas dos causas de generación de un paquete ICMP nos ayudan a dividir dichos paquetes en dos clases: errores ICMP y peticiones-respuestas ICMP.

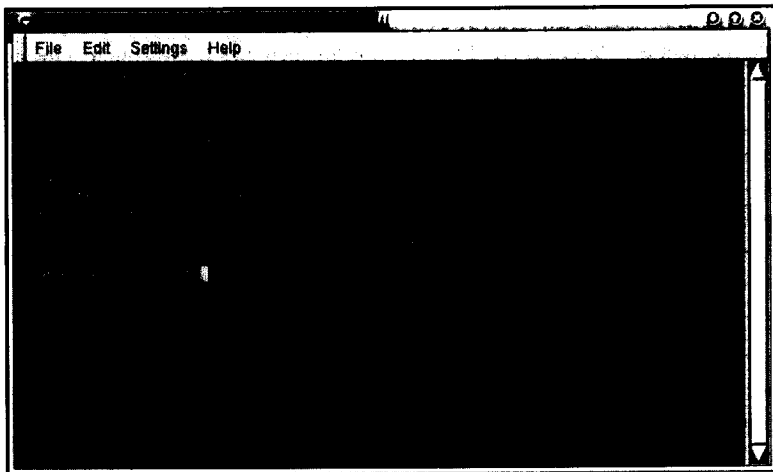


Figura 4.12. Un rastro de transacciones DNS en UDP.

Los errores ICMP generalmente se deben a un fallo en la entrega de un datagrama IP. Como los datagramas IP no siguen una ruta fijada a través de la red, los mensajes de error ICMP se devuelven sólo al *host* de origen, no a los encaminadores intermedios. (Esto nos permite construir herramientas de diagnóstico como *traceroute*, que explicaremos en el Capítulo 9.)

Todos los paquetes ICMP siguen la misma estructura básica: un campo de Tipo de 1 byte, un campo de Código de 1 byte y un campo de Suma de comprobación de 2 bytes, seguidos de una cierta variedad de campos, dependiendo del tipo de paquete. Como la suma de comprobación del encabezamiento IP, la de ICMP utiliza un valor de 0x0000 para su cálculo. A diferencia de ella, que utiliza sólo el encabezamiento, para el cálculo se utiliza todo el paquete ICMP.

Veremos la estructura de varios tipos de paquetes ICMP paso a paso. La Tabla 4.4 muestra los diferentes tipos de paquetes ICMP resumidos.

Tabla 4.4. Paquetes ICMP con Tipo y Código

Tipo	Código	Función
0/8	0	Petición-respuesta eco.
3	0-15	Destino inaccesible.
4	0	Apagado de origen.
5	0-3	Redireccionamiento.
9/10	0	Solicitud-aviso de encaminador (no mostrado).
11	0-1	TTL excedido.
12	0-1	Error de parámetro.
13/14	0	Petición-respuesta de marca de tiempo.
15/16	0	Obsoleto (no mostrado).
17/18	0	Petición-respuesta de máscara de subred (no mostrado).

Nota: no vamos a describir en el texto todos los mensajes ICMP enumerados en esta tabla. Hemos intentado describir sólo los más comunes e importantes.

- Hay cinco tipos de errores que no producen mensajes ICMP:
- Un fallo que implique a un mensaje de error ICMP.
- Un datagrama IP con una dirección de destino de difusión.
- Un marco de capa de enlace con una dirección de destino de difusión.
- Cualquier fragmento de un datagrama IP distinto del primero.
- Cualquier datagrama cuya dirección de origen no sea una dirección uniconversión.

Si se enviaran mensajes de error ICMP para estos tipos de errores, el resultado podría ser un gran periodo de tráfico llamado tormenta de difusión³.

Petición-respuesta eco

Los mensajes de petición y respuesta eco ICMP son la base del comando */bin/ping* explicado en el Capítulo 9. El *host* origen lleva a cabo una petición eco ICMP (Tipo 8) y el *host* destino devuelve una respuesta eco ICMP (Tipo 0).

El mensaje de petición/respuesta eco ICMP está organizado así: un campo Tipo de 1 byte, un campo Código de 1 byte (siempre 0x00), un campo Suma de comprobación de 2 bytes, un campo Número ID de 2 bytes, un campo Número de secuencia de 2 bytes y un campo Datos de tamaño variable. El Número ID es único para comando *ping* que se lleve a cabo. El Número de secuencia se incrementa de forma separada en cada comando *ping*. Si ejecu-

³ Una tormenta de difusión es un periodo de tráfico muy pesado (que a veces consume todo el ancho de banda disponible), caracterizada por el tráfico enviado a direcciones que no son uniconversión.

tamos un guión como el siguiente, observaremos que cada *ping* tiene un Número ID diferente pero el mismo Número de secuencia:

```
for count in 1 2 3; do
    /bin/ping -c1 192.168.1.1
done
```

Si ejecutamos este comando, veremos que cada *ping* tiene el mismo Número ID pero que se incrementan los Números de secuencia:

```
/bin/ping -c3 192.168.1.1
```

Destino inaccesible

Los mensajes de Destino inaccesible de ICMP se envían cuando no se puede entregar un paquete en el destino. Estos mensajes están organizados así: un campo Tipo de 1 byte (0x03), un campo Código de 1 byte (véase la Tabla 4.6), un campo Suma de comprobación de 2 bytes, un campo Reservado de 4 bytes (este campo debe ser 0x00000000), un campo Encabezamiento IP fallido de 20 bytes y un campo Datagrama IP fallido de 8 bytes. Los 20 bytes del campo Encabezamiento IP fallido ofrecen todos los datos necesarios para identificar el paquete IP de origen de la condición de error. Los 8 bytes del campo Datos IP fallidos muestran los primeros 8 bytes del encabezamiento de la capa de transporte; esto ofrece suficiente información para identificar la aplicación que ha causado la condición de error. Mostramos la organización de este tipo de paquete ICMP en la Tabla 4.5.

Tabla 4.5. Campos del mensaje Destino inaccesible de ICMP

Tamaño	Contenido	Datos de ejemplo
1 byte	Tipo	03
1 byte	Código	00
2 bytes	Suma de comprobación	0a7b
4 bytes	Reservado	00 00 00 00
20 bytes	Encabezamiento IP fallido	45 ...
8 bytes	Datos IP fallidos	05 b3 ...

La Tabla 4.6 muestra varios valores de Código que se pueden enviar en los mensajes Destino inaccesible de ICMP.

Tabla 4.6. Códigos de Destino inaccesible de ICMP

Código	Significado
0	La red es inaccesible
1	El <i>host</i> es inaccesible

(continúa)

Tabla 4.6. Códigos de Destino inaccesible de ICMP. (*Continuación*)

Código	Significado
2	El protocolo es inaccesible.
3	El puerto es inaccesible.
4	Se necesita fragmentación pero está desactivada.
5	Ruta de origen fallida.
6	La red de destino es desconocida.
7	El <i>host</i> de destino es desconocido.
8	Obsoleto.
9	La red de destino está prohibida.
10	El <i>host</i> de destino está prohibido.
11	La red es inaccesible para TOS.
12	El <i>host</i> es inaccesible para TOS.
13	La comunicación está prohibida por un filtro.
14	Ha tenido lugar una violación de precedencia de <i>host</i> .
15	El corte de precedencia está en efecto.

Los códigos del 0 al 3 son los valores más normales, aunque el 4 se utiliza al determinar la ruta MTU de una conexión de red.

Apagado de origen

Los mensajes de Apagado de origen ICMP se utilizan cuando un encaminador está cercano al límite de su capacidad de *buffer*. El encaminador envía su mensaje al *host* de origen del datagrama que disparó el evento. Se supone que el sistema que recibe el mensaje de apagado de origen reduce su índice de transmisión hasta que deja de recibir este tipo de mensajes.

El mensaje de apagado de origen contiene los siguientes campos: un campo Tipo de 1 byte (siempre 0x04), un campo Código de 1 byte, un campo Suma de comprobación de 2 bytes, un campo Reservado de 4 bytes (este campo debería ser 0x00000000), un campo Encabezamiento IP fallido de 20 bytes y un campo Datos IP fallidos de 8 bytes. La Tabla 4.7 muestra una comparación de los campos de los mensajes de apagado de origen.

Tabla 4.7. Campos del mensaje Apagado de origen de ICMP

Tamaño	Contenido	Datos de ejemplo
1 byte	Tipo	04
1 byte	Código	00
2 bytes	Suma de comprobación	0a7b
4 bytes	Reservado	00 00 00 00
20 bytes	Encabezamiento IP fallido	45 ...
8 bytes	Datos IP fallidos	05 b3 ...

Redireccionamiento

Los mensajes de redireccionamiento de ICMP se envían cuando un *host* intenta utilizar un encaminador inapropiado. La Figura 4.13 muestra una red en la que podría ocurrir.

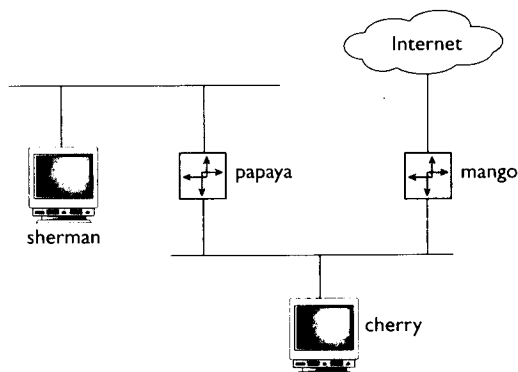


Figura 4.13. Una red con dos encaminadores.

cherry utiliza a mango como su encaminador predeterminado, para la siguiente tabla de enrutamiento:

Tabla de encaminamiento IP de núcleo

Destination	Gateway	Genmask	Flags	MSS Window	irtt	Iface
192.168.1.10	0.0.0.0	255.255.255.255	UH	0 0	0	eth0
192.168.1.0	0.0.0.0	255.255.255.0	U	0 0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0	0	lo
0.0.0.0	192.168.1.1	0.0.0.0	UG	0 0	0	eth0

mango necesita enviar tráfico a sherman (que reside en una red tras papaya). cherry envía el primer paquete a mango, que se lo pasa a papaya y envía un mensaje de redireccionamiento ICMP (mensaje ICMP Tipo 5 Código 0) a cherry. Entonces cherry debería actualizar su tabla de encaminamiento para que tuviera la siguiente apariencia:

Tabla de encaminamiento IP de núcleo

Destination	Gateway	Genmask	Flags	MSS Window	irtt	Iface
192.168.1.10	0.0.0.0	255.255.255.255	UH	0 0	0	eth0
192.168.1.0	0.0.0.0	255.255.255.0	U	0 0	0	eth0
192.168.2.0	192.168.1.11	255.255.255.0	UD	0 0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0 0	0	lo
0.0.0.0	192.168.1.1	0.0.0.0	UG	0 0	0	eth0

Todo el tráfico posterior a la red 192.168.2.0/24 debería ir ahora a través de papaya.

Un mensaje de redireccionamiento ICMP está organizado de la misma forma que el mensaje de apagado de origen descrito anteriormente. Los campos son un campo Tipo de 1 byte, un campo Código de 1 byte, un campo Suma de comprobación de 2 bytes, un campo Reservado de 4 bytes (este campo

debería ser 0x00000000), un campo Encabezamiento IP fallido de 20 bytes y un campo Datos IP fallidos de 8 bytes. Mostramos la organización del mensaje y los posibles valores del campo Código en las Tablas 4.8 y 4.9.

Tabla 4.8. Campos del mensaje Redireccionamiento de ICMP

Tamaño	Contenido	Datos de ejemplo
1 byte	Tipo	05
1 byte	Código	00
2 bytes	Suma de comprobación	0a7b
4 bytes	Reservado	00 00 00 00
20 bytes	Encabezamiento IP fallido	45 ...
8 bytes	Datos IP fallidos	05 b3 ...

Tabla 4.9. Códigos de Redireccionamiento de ICMP

Código	Significado
0	Redireccionamiento para red
1	Redireccionamiento para <i>host</i>
2	Redireccionamiento para red y TOS
3	Redireccionamiento para <i>host</i> y TOS

TTL excedido

El mensaje TTL excedido de ICMP es la base del comando `/usr/sbin/traceroute`. Se genera cuando la cuenta de saltos de un paquete excede su TTL. (Por favor, observemos que también hay un código que representa un exceso de tiempo mientras esperamos la desfragmentación.) Los mensajes TTL contienen los siguientes campos: un campo Tipo de 1 byte (siempre 0x011), un campo Código de 1 byte, un campo Suma de comprobación de 2 bytes, un campo Reservado de 4 bytes (este campo debería ser 0x00000000), un campo Encabezamiento IP fallido de 20 bytes y un campo Datos IP fallidos de 8 bytes. Mostramos este mensaje y los Códigos válidos en las Tablas 4.10 y 4.11.

Tabla 4.10. Mensaje TTL excedido de ICMP

Tamaño	Contenido	Datos de ejemplo
1 byte	Tipo	0b
1 byte	Código	00
2 bytes	Suma de comprobación	0a7b
4 bytes	Reservado	00 00 00 00
20 bytes	Encabezamiento IP fallido	45 ...
8 bytes	Datos IP fallidos	05 b3 ...

Tabla 4.11. Códigos de TTL excedido de ICMP

Código	Significado
0	TTL igual a 0 durante el tránsito.
1	TTL igual a 0 durante la reconstrucción.

Error de parámetro

Cuando se detecta un error en el encabezamiento IP de un paquete, se envía un mensaje error de parámetro de ICMP al *host* que lo origina. Este mensaje indica que uno de los campos se ha perdido o es incorrecto. La organización del mensaje de error de parámetro es similar a la de los otros mensajes de error de ICMP: un campo Tipo de 1 byte (siempre 0x0c), un campo Código de 1 byte, un campo Suma de comprobación de 2 bytes, un campo Puntero a los datos erróneos de 1 byte, un campo Reservado de 3 bytes (este campo debería ser 0x00000000), un campo Encabezamiento IP fallido de 20 bytes y un campo Datos IP fallidos de 8 bytes. Podemos ver dicha organización en la Tabla 4.12. Observemos que el campo Puntero a los datos erróneos es 0 (lo que indica que el error está en el primer byte).

La Tabla 4.13 muestra los campos Código válidos para el mensaje de parámetro de ICMP.

Tabla 4.12. Mensaje Error de parámetro de ICMP

Tamaño	Contenido	Datos de ejemplo
1 byte	Tipo	0b
1 byte	Código	00
2 bytes	Suma de comprobación	0a7b
1 byte	Puntero a los datos erróneos	00
3 bytes	Reservado	00 00 00 00
20 bytes	Encabezamiento IP fallido	40 ...
8 bytes	Datos IP fallidos	05 b3 ...

Tabla 4.13. Códigos de Error de parámetro de ICMP

Código	Significado
0	El encabezamiento IP es erróneo.
1	Se ha perdido una opción necesaria.

5

Protocolos de la capa de aplicación

Desde que comenzamos con la capa de enlace en el Capítulo 2, “Protocolos de la capa de enlace”, hemos recorrido la pila de protocolos. Ahora hemos llegado a la cima. Ahora, en las cumbres de la capa de aplicación, vamos a ver mucha más variedad que en cualquier otra capa que hayamos explicado. En este capítulo, explicaremos tres protocolos: RIP, TFTP y HTTP. Los dos primeros están contruidos sobre UDP; el tercero utiliza TCP.

RIP

Utilizamos el RIP (*Routing Information Protocol*, Protocolo de información de encaminamiento) para enviar información de encaminamiento entre *hosts* en red¹. RIP se especificó formalmente en el RFC 1058, que fue escrito varios años después de la implementación del protocolo en Berkeley UNIX. Actualmente, la versión más común de RIP con más uso es la versión 2, o RIP-2, especificada en el RFC 1388. Los equipos más antiguos que ejecuten RIP pueden entender los paquetes RIP-2 porque utiliza campos reservados de RIP para enviar información adicional y RIP ignora dichos paquetes. En el resto de esta sección, vamos a hablar de la versión 2 de RIP; cuando utilicemos el término RIP, nos referiremos a RIP-2.

¹ Para obtener más información sobre encaminamiento, véase la sección “Encaminamiento” del Capítulo 3, “Protocolos de la capa de red”.

RIP sólo proporciona un esquema de encaminamiento muy sencillo: confía en la cuenta de saltos como única métrica para la determinación de la ruta a escoger. Cada *host* participante retransmite un mensaje que contiene su lista de redes conocidas con la correspondiente cuenta de saltos. Como las actualizaciones se retransmiten sólo de forma periódica (generalmente cada 30 o 90 segundos), a los encaminadores les puede llevar mucho tiempo propagar una red. RIP tiene una forma de ayudar a reducir esto al mínimo, cualquier cosa que requiera más de 15 saltos se considera inaccesible.

La Figura 5.1 muestra los encaminadores utilizados para conectar varias redes pequeñas. La Tabla 5.1 muestra las interfaces y direcciones IP de dichos encaminadores (damos el nombre en el formato nombre.interfaz). Utilizaremos esta red para mostrar el funcionamiento normal de RIP y para ilustrar un inconveniente de RIP junto con su solución.

Tabla 5.1. Encaminadores, interfaces y direcciones

Nombre e interfaz	Dirección IP y máscara de subred
aaron.0	192.168.1.1/24
aaron.1	192.168.2.1/24
aaron.2	192.168.3.1/24
alan.0	192.168.4.1/24
alan.1	192.168.5.1/24
alan.2	192.168.6.1/24
bob.0	192.168.6.2/24
bill.0	192.168.5.2/24
bill.1	192.168.12.1/24
bill.2	192.168.13.1/24
bruce.0	192.168.3.2/24
bruce.1	192.168.14.1
brandon.0	192.168.2.2/24
brandon.1	192.168.15.1/24
chuck.0	192.168.13.2/24
chuck.1	192.168.21.1/24
caleb.0	192.168.12.2/24
caleb.1	192.168.22.1/24
caleb.2	192.168.23.1/24
carl.0	192.168.14.2/24
carl.1	192.168.15.2/24
carl.2	192.168.24.1/24
doug.0	192.168.22.2/24
doug.1	192.168.24.2/24
don.0	192.168.21.2/24
don.1	192.168.23/24

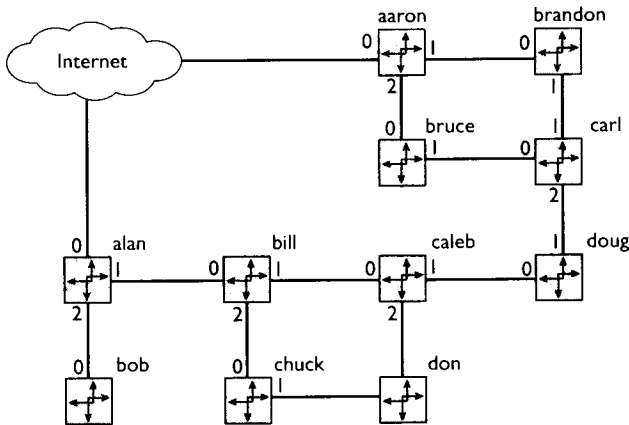


Figura 5.1. Los encaminadores de una pequeña interred.

Cada encaminador de la Tabla 5.1 tiene una visión distinta de la red. alan y aaron tienen una conexión directa a Internet y pueden actuar como pasarelas predeterminadas para las redes que hay tras ellas. doug y don están en la parte de atrás de la red y ambas tienen una cierta variedad de rutas para llegar a la parte de delante. Comencemos observando el punto de vista del mundo de alan, bill, caleb y doug, además de la información de encaminamiento que enviarán.

alan tiene una entrada predeterminada de 0.0.0.0/0 a través de la interfaz 0. No estará configurada para enviar información de encaminamiento fuera de la interfaz, para evitar direcciones de difusión internas (RFC 1918). También tiene caminos de un salto hacia las redes 192.168.4.0/24, 192.168.5.0/24 y 192.168.6.0/24². Como bill ya tiene un camino de un salto hacia la red 192.168.5.0/24, ignora esta entrada. Las demás se añaden a su tabla de encaminamiento. En su siguiente difusión, bill envía información sobre todas las entradas de uno y dos saltos de su tabla de encaminamiento. (Ahora, también alan conoce la red 192.168.13.0/24.)

BUCLES DE ENCAMINAMIENTO Y HORIZONTE DIVIDIDO

Aquí es donde nos encontramos nuestro problema más importante con RIP. ¿Qué ocurriría si alan.3 fallara? bill enviaría una difusión RIP que mostraría una entrada de dos saltos hacia la red 192.168.6.0/24. Normalmente, alan las ignoraría, porque tienen una cuenta de saltos mayor que sus propias entradas, pero, sin la interfaz eth3, los caminos de bill pasan a ser los mejores disponibles, excepto en que implican un salto hacia atrás hasta alan.

RIP no lleva suficiente información para detener este tipo de bucles de encaminamiento. En lugar de ello, confía en un método llamado horizonte dividido. Ningún participante RIP retransmitirá fuera de la interfaz un camino que haya recibido en ella. Si bill sigue esta norma, alan nunca recibe los caminos erróneos de bob y evitamos el bucle de encaminamiento.

² Éstos son todos caminos de dos saltos porque bill ha añadido uno a la cuenta de saltos para cada camino, indicando que debe pasar por alan.

Continuando con nuestro ejemplo, caleb tiene caminos de un salto a las redes 192.168.12.0/24, 192.168.22.0/24 y 192.168.23.0/24. Cuando recibe la difusión de bill, caleb aprende caminos de dos saltos a las redes 192.168.13.0/24 y 192.168.5.0/24 y de tres a las redes 192.168.4.0/24 y 192.168.6.0/24. Retransmitirá su propia tabla de encaminamiento y bill y alan conocerán las redes 192.168.22.0/24 y 192.168.23.0/24 (con dos y tres saltos respectivamente).

doug comienza con caminos de un salto a las redes 192.168.22.0/24 y 192.168.24.0/24. Tras recibir las actualizaciones de caleb, doug también tendrá caminos de saltos a las redes 192.168.23.0/24 y 192.168.12.0/24, de tres a las 192.168.13.0/24 y 192.168.5.0/24 y de cuatro a las 192.168.4.0/24 y 192.168.6.0/24.

En las descripciones anteriores hemos visto el flujo de los datos RIP en una sola dirección. En realidad, fluye en ambas direcciones. Después de que la red lleve conectada un rato, los caminos se propagan a través de ella. En este momento, alan tendrá una tabla de encaminamiento como la mostrada en la Tabla 5.2.

Tabla 5.2. Tabla de encaminamiento de alan

Red	Salto siguiente	Cuenta de saltos
192.168.4.0/24	192.168.4.1	1
192.168.5.0/24	192.168.5.1	1
192.168.6.0/24	192.168.6.1	1
192.168.12.0/24	192.168.6.2	2
192.168.13.0/24	192.168.5.2	2
192.168.21.0/24	192.168.5.2	3
192.168.22.0/24	192.168.5.2	3
192.168.23.0/24	192.168.5.2	3
192.168.24.0/24	192.168.5.2	4
192.168.14.0/24	192.168.5.2	5
192.168.15.0/24	192.168.5.2	5
192.168.3.0/24	192.168.5.2	6
192.168.2.0/24	192.168.5.2	6
192.168.1.0/24	192.168.5.2	7

Rastro del protocolo RIP-2

La Figura 5.2 muestra una petición RIP-2. El paquete está dirigido a la dirección de difusión IP. Tanto el puerto de origen como el de destino se envían a 520. Dentro del datagrama RIP, el campo Comando está establecido como 1 (Petición). El campo Versión de RIP está establecido como 2.

TFTP

El TFTP (*Trivial File Transfer Protocol*, Protocolo de transferencia de archivos trivial) es un protocolo basado en el UDP (*User Datagram Protocol*, Protocolo de datagrama de usuario) para el intercambio de archivos entre sistemas. Fue diseñado para acelerar y facilitar la implementación de un *host* con recursos limitados. TFTP está definido en el RFC 1350.

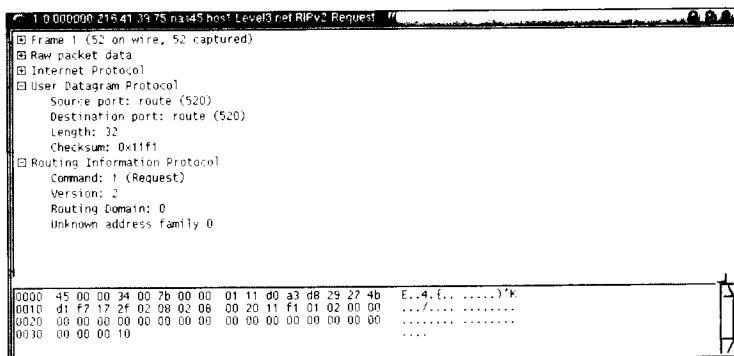


Figura 5.2. Un aviso de camino RIP

Revisión del protocolo

TFTP está implementado con cinco tipos de paquetes:

- Peticiones de lectura.
- Peticiones de escritura.
- Datos de archivo.
- Acuses de recibo.
- Manipulación de errores.

Cada uno de estos tipos de paquetes (excepto las peticiones de lectura y de escritura) tiene una estructura de paquete ligeramente diferente. Echaremos un vistazo a cada una de ellas por orden.

Las peticiones de lectura y escritura de paquetes tienen cinco campos. El primero es de 2 bytes y es el campo Código Op; las peticiones de lectura son 0x0001 y las de escritura 0x0002. El siguiente campo es de tamaño variable y es el nombre del archivo a leer. A continuación hay un separador de 1 byte, que siempre es 0x00. El cuarto campo es el Modo de datos y puede tener uno de estos tres valores:

- Netascii.
- Binario.
- La dirección de e-mail del remitente (sólo en las peticiones de escritura).

El cuarto campo es de tamaño variable. El quinto es de 1 byte y es un marcador de EOF (*End of File*, Final de archivo); siempre es 0x00.

Los paquetes de datos sólo tienen tres campos. El primero es el campo Código Op y tiene 2 bytes; siempre es 0x0003. El segundo es el Número de

bloque y tiene 2 bytes. El resto del paquete (con una longitud máxima de 512 bytes) son los datos. Cuando TFTP transfiere datos, utiliza un mecanismo sencillo para ver dónde está dentro de los datos a ser transferidos. Los datos están divididos en bloques de 512 bytes y cada uno se envía y se acusa recibo de forma individual³. El último paquete de una transferencia es un paquete de datos con un campo de datos menor de 512 bytes. Si los datos se dividen exactamente en bloques de 512 bytes, se enviará un paquete vacío en último lugar (con 0 bytes, que es menor que 512).

Cada bloque de datos se responde con un paquete de acuse de recibo. Este paquete tiene 4 bytes y dos campos. El primer campo es el Código Op de 2 bytes (0x0004). El segundo es el número de bloque de 2 bytes del bloque al que se está dando acuse de recibo.

Como nada es perfecto, TFTP tiene un sistema de manipulación de errores sencillo construido en un paquete de manipulación de errores. Estos paquetes tienen cuatro campos y son de tamaño variable. El primer campo es el Código Op; tiene 2 bytes y es siempre 0x0005. El segundo es de 2 bytes y lleva el código de error. El tercero es el texto del error, de tamaño variable. En la Tabla 5.3 mostramos los códigos de error válidos y sus correspondientes textos de error.

Tabla 5.3. Códigos de error TFTP.

Código de error	Texto de error
0x0000	Error indefinido.
0x0001	Archivo no encontrado.
0x0002	Violación de acceso.
0x0003	Espacio de disco excedido.
0x0004	Operación ilegal.
0x0005	ID de transferencia desconocida.
0x0006	El archivo existe.
0x0007	No existe dicho usuario.

Rastro del protocolo TFTP

La Figura 5.3 muestra una petición de lectura TFTP; el campo Código Op es 0x0001. El siguiente campo es el nombre del archivo a leer, testout. A continuación hay un marcador EOF, 0x00. El siguiente campo es el modo de datos, netascii. El último es otro marcador EOF, 0x00.

³ Nos puede llevar a una complicación poco afortunada. El remitente guarda un temporizador en los datos que envía. Si el receptor no acusa recibo de un bloque de datos antes de que el temporizador expire, el remitente enviará otra copia. Si el receptor ha tardado, enviará un acuse de recibo del bloque y después otro cuando reciba la segunda copia. Si continúa ocurriendo, crea un efecto de bola de nieve que congestiona la red. A esto frecuentemente se le llama el fallo del Aprendiz de brujo, debido a la escena de Fantasía en la que Mickey Mouse conjura un ejército de gamuzas para limpiar la torre del brujo (las gamuzas se le van rápidamente de las manos, como el proceso TFTP que sufre este error).

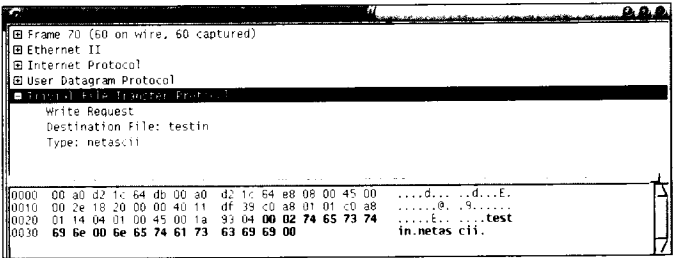


Figura 5.3. Una petición de lectura TFTP.

La Figura 5.4 muestra una petición de escritura TFTP; el campo Código Op es 0x0002. El nombre del archivo a escribir es testin, seguido de un 0x00 (el marcador EOF). El modo de datos es netascii y también está seguido por un marcador EOF

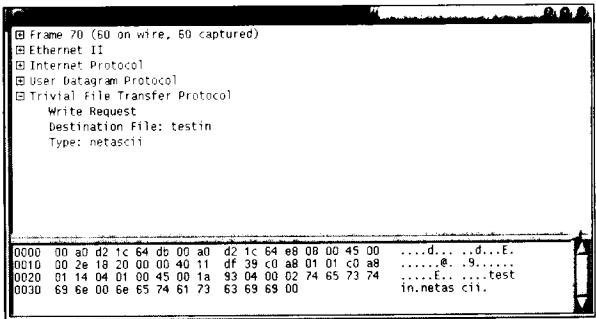


Figura 5.4. Una petición de escritura TFTP.

En la Figura 5.5 mostramos el paquete de datos TFTP. El campo Código Op es 0x0003; el número de bloque es 0x0001. El resto del paquete es un gran campo Datos. Tiene 512 bytes, así que sabemos que quedan más paquetes por llegar.

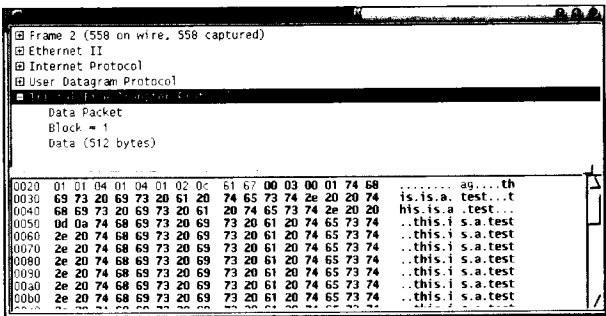


Figura 5.5. Un paquete de datos TFTP.

La Figura 5.6 muestra un paquete de Acuse de recibo TFTP. Su campo Código Op es 0x0004 y su número de bloque es 0x0001. El resto del paquete está relleno para alcanzar el límite mínimo de tamaño de datos de un paquete de Ethernet II.

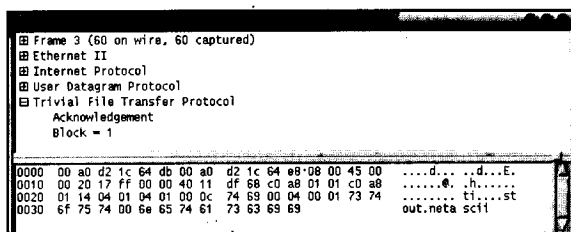


Figura 5.6. Un paquete de acuse de recibo TFTP.

En la Figura 5.7 mostramos el último tipo de paquete TFTP, un paquete de error TFTP. El campo Código Op es 0x0005. El código de error es 0x0001 y el texto de error es “*File not found*” (Archivo no encontrado).

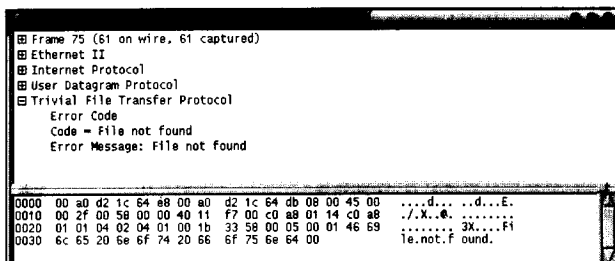


Figura 5.7. Un paquete de error TFTP.

Configuración de un servidor para TFTP

La configuración de un equipo Linux para servir a TFTP es una tarea bastante sencilla. TFTP es un hueco de seguridad notorio, pero también puede ser una parte importante de nuestra red. Los sistemas sin disco y otros tipos de hardware utilizan frecuentemente TFTP para recuperar archivos de configuración de un servidor específico.

Hay dos problemas importantes de seguridad en TFTP: el primero, permite el acceso de escritura y lectura anónimo en un puerto bien conocido. El segundo, es una aplicación basada en UDP, así que es difícil de manipular con un cortafuegos (casi el bloqueo de todo).

Como ocurre en la mayoría de los protocolos, si no necesitamos TFTP, mejor no ejecutarlo. Si necesitamos un servidor TFTP, lo ejecutamos en un solo equipo y hacemos todo lo posible por mantenerlo tan seguro como podamos. Los siguientes párrafos explican cómo hacerlo.

La configuración de un servidor TFTP es tan sencilla como modificar nuestro archivo `/etc/inetd.conf` y enviar a `inetd` la señal HUP⁴. La sección que queremos cambiar tiene la apariencia de la que mostramos en el Ejemplo 5.1 (al menos en sistemas basados en Red Hat 6.2).

Ejemplo 5.1. Un `inetd.conf` sin TFTP activado.

```
# El servicio TFTP se proporciona principalmente para el arranque.
# La mayoría de los sitios
# ejecutan esto sólo en equipos que actúan como "servidores de
# arranque." No deje de comentar
# esto a menos que lo *necesite*.
#
#tftp dgram udp wait root /usr/sbin/tcpd in.tftpd
#bootps dgram udp wait root /usr/sbin/tcpd bootpd
#
```

Tenemos que cambiarla hasta que tenga la apariencia del Ejemplo 5.2.

Ejemplo 5.2. Un `inetd.conf` con TFTP activado.

```
# El servicio TFTP se proporciona principalmente para el arranque.
# La mayoría de los sitios
# ejecutan esto sólo en equipos que actúan como "servidores de
# arranque." No deje de comentar
# esto a menos que lo *necesite*.
#
#tftp dgram udp wait root /usr/sbin/tcpd in.tftpd
#bootps dgram udp wait root /usr/sbin/tcpd bootpd
#
```

Después de hecho este cambio, podemos suprimir el servidor `inetd`. Mostramos el proceso en el Ejemplo 5.3.

Ejemplo 5.3. HUP de `inetd`.

```
[root@cherry /root]# ps -aux | grep inetd
root      482  0.0  0.0 1148  0 ?        SW   Aug08   0:00 [inetd]
root     2791  0.0  0.8 1360 508 pts/1    R    04:31   0:00 grep inetd
[root@cherry /root]# ps -ef | grep inetd
root      482      1  0 Aug08 ?          00:00:00 [inetd]
[root@cherry /root]# kill -1 482
[root@cherry /root]# ps -aux | grep inetd
root      482  0.0  0.3 1148 240 ?        S    Aug08   0:00 inetd
root     2795  0.0  0.8 1364 524 pts/1    S    04:32   0:00 grep inetd
[root@cherry /root]#
```

⁴ Un HUP o Colgado (*Hang UP*) se envía con el comando `kill -1 [pid]`.

Cuando terminemos, tendremos un servidor TFTP en ejecución. La gran pregunta ahora es, “¿Qué hará?” De forma predeterminada, `tftpd` sirve archivos fuera del directorio `/tftpboot`. Puede que queramos cambiar esto y convertirlo en algo en su propio sistema de archivos para ayudar a mitigar ataques de *stuffing* de sistemas de archivos⁵. Podemos ajustar el directorio del que `tftpd` sirve los archivos ajustando así nuestro archivo `/etc/inetd.conf`:

```
tftp dgram udp wait root /usr/sbin/tcpd in.tftpd /mnt/tftpserve
```

Tras crear este archivo y hacer HUP de `inetd`, `tftpd` proporcionará servicios del sistema de archivos `/mnt/tftpserve`.

Necesitaremos colocar los archivos que queramos servir en este directorio. Tanto éstos como el directorio necesitarán ser legibles para todo el mundo (modo 666 para archivos y 777 para directorios), porque `tftpd` no utiliza autenticación para las transferencias de archivos. `tftpd` permitirá acceso de escritura sólo a archivos que existan y que sean de escritura para todo el mundo (modo 555).

Cómo utilizar TFTP

El cliente del protocolo TFTP está implementado como el ejecutable `/usr/bin/tftp`. Este comando no está instalado en todas las distribuciones de Linux.

Opciones de línea de comandos

`/usr/bin/tftp` nos permite sólo una opción de línea de comandos: el nombre o dirección IP del *host* remoto. Utilizamos el comando como mostramos en el Ejemplo 5.4.

Ejemplo 5.4. Inicio de TFTP.

```
[pate@cherry pate] tftp crashtestdummy
tftp?
```

El comando del Ejemplo 5.4 comienza una sesión TFTP con el *host* `crashtestdummy`. La parte del *host* remoto del comando es opcional y se puede llevar a cabo de forma tan fácil como en el Ejemplo 5.5.

Ejemplo 5.5. Otra forma de iniciar TFTP.

```
[pate@router pate]$ tftp
tftp?
[pate@router pate]$
```

⁵ Es posible escribir archivos increíblemente grandes en un sistema utilizando TFTP, rellenando eventualmente el sistema de archivos (si es el sistema de archivos raíz) y cerrando el sistema.

Comandos interactivos

TFTP permite un cierto número de comandos interactivos. Los más importantes son éstos:

- Connect
- Get
- Mode
- Put
- Status
- Trace
- Verbose
- ?

Explicaremos cada uno de ellos en esta sección. Comenzaremos con `trace` y `status` porque nos ofrecerán mucha luz en posteriores explicaciones.

El comando `status` nos muestra el estado actual de cada paquete TFTP. Lo mostramos en el Ejemplo 5.6.

Ejemplo 5.6. El comando `status` de TFTP

```
tftp? status
Not connected.
Mode: netascii Verbose: off Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp?
```

El comando `trace` envía un informe a la terminal por cada paquete TFTP. Lo mostramos en el Ejemplo 5.7.

Ejemplo 5.7. El comando `trace` de TFTP

```
tftp? trace
tftp*rang; get testin
tftp> Packet tracing on.
tftp> sent RRQ ?file=testin, mode=netascii?
sent RRQ ?file=testin, mode=netascii?
sent RRQ ?file=testin, mode=netascii?
sent RRQ ?file=testin, mode=netascii?
sent RRQ ?file=testin, mode=netascii?
Transfer timed out.
```

Si no especificamos un *host* remoto, el comando `connect` nos permite especificar uno. Podemos utilizar `connect` con un *host* nuevo incluso si ya estamos conectados con uno. Utilizamos este comando como mostramos en el Ejemplo 5.8.

Ejemplo 5.8. El comando connect de TFTP.

```
tftp? 192.168.1.20
tftp? status
Connected to 192.168.1.20.
Mode: netascii Verbose: off Tracing: off
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds
tftp?
```

Para recuperar un archivo, utilizamos el comando `get`. Podemos especificar nombres de archivo de varias formas:

- `get filename`
- `get remotefile localfile`
- `get file1 file2 ... fileN`
- `get`

La segunda y la tercera opción pueden producir algo de confusión: si introducimos sólo dos nombres de archivo, leemos en el primero y escribimos en un archivo local llamado como el segundo. Si introducimos más de dos, cada uno de ellos se leerá desde el servidor y se escribirá en archivos del mismo nombre en el cliente. Si utilizamos el cuarto formato, se nos pide el archivo o los archivos que queremos descargar; el formato es el mismo que en los tres primeros de `get`. También podemos especificar un nombre de *host* con el comando `get`; el formato es `get host:file`. El Ejemplo 5.9 muestra varios métodos para la obtención de archivos con este comando.

Ejemplo 5.9. El comando get de TFTP.

```
tftp? trace
tftp? get testout
sent RRQ ?file=testout, mode=netascii?
received DATA ?block=1, 17 bytes?
Received 17 bytes in 0.1 seconds
tftp? get
(files) testin testout
sent RRQ ?file=testin, mode=netascii?
received DATA ?block=1, 448 bytes?
Received 448 bytes in 0.1 seconds
tftp? get
(files) testin testout foo
sent RRQ ?file=testin, mode=netascii?
received DATA ?block=1, 448 bytes?
Received 448 bytes in 0.0 seconds
sent RRQ ?file=testout, mode=netascii?
received DATA ?block=1, 17 bytes?
Received 17 bytes in 0.0 seconds
sent RRQ ?file=foo, mode=netascii?
received ERROR ?code=1, msg=File not found?
Error code 1: File not found
tftp?
```

Utilizamos el comando `mode` para establecer el modo de los datos para todas las transferencias siguientes. Podemos utilizarlo repetidamente para cambiar de `netascii` a binario según lo necesitemos. `mode` espera un solo argumento que describa el modo de los datos que queremos utilizar. Si no damos un modo de datos, TFTP responde con el modo actual. Mostramos el comando `mode` en el Ejemplo 5.10.

Ejemplo 5.10. El comando `mode` de TFTP

```
tftp? mode
Using netascii mode to transfer files.
tftp? mode binary
tftp? mode
Using octet mode to transfer files.
tftp? mode netascii
tftp?
```

Podemos escribir un archivo en el *host* remoto con el comando `put`. Su sintaxis es como la de `get`, explicada anteriormente.

Para hacer una rápido repaso de todos los comandos (y de los demás que no hemos explicado), podemos utilizar el comando `?`. Devuelve una lista de comandos con una pequeña descripción, como mostramos en el Ejemplo 5.11.

Ejemplo 5.11. El comando `?` de TFTP

```
tftp? ?
Los comandos pueden abreviarse. Los comandos son:
```

<code>connect</code>	connect to remote tftp
<code>mode</code>	set file transfer mode
<code>put</code>	send file
<code>get</code>	receive file
<code>quit</code>	exit tftp
<code>verbose</code>	toggle verbose mode
<code>trace</code>	toggle packet tracing
<code>status</code>	show current status
<code>binary</code>	set mode to octet
<code>ascii</code>	set mode to netascii
<code>rexmt</code>	set per-packet retransmission timeout
<code>timeout</code>	set total retransmission timeout
<code>?</code>	print help information
<code>tftp?</code>	

HTTP

El HTTP (*Hypertext Transfer Protocol*, Protocolo de transferencia de hipertexto), descrito en el RFC 1945 y en el 2616, ofrece un protocolo de

capa de aplicación para la distribución de datos a través de la red. Permite varios métodos de petición y envío de datos. HTTP proporciona un método para la introducción de datos que se están transmitiendo.

A diferencia de los dos protocolos explicados anteriormente, está implementado sobre el protocolo de capa de transmisión TCP. El protocolo TCP ofrece información de sesión que permite a HTTP evitar hacer algo del trabajo que un protocolo basado en UDP necesitaría hacer.

HTTP es similar en muchas cosas al protocolo TFTP: ambos están diseñados para enviar datos entre dos *hosts* conectados por red. HTTP no necesita preocuparse por la división de datos en bloques de tamaño explícito ni de la señalización de una condición de Fin de Datos. Sin embargo, tiene que cargar la gran cantidad de datos de la configuración de la sesión de TCP.

La descripción más sencilla de HTTP es que consiste en un proceso de cuatro pasos⁶:

1. El cliente y el servidor establecen una sesión TCP. El servidor normalmente reside en el puerto TCP 80.
2. El cliente envía una petición HTTP. Normalmente es una petición GET y tiene esta apariencia: GET /file.html HTTP/1.0.
3. El servidor responde enviando los datos solicitados al cliente.
4. Los *hosts* cierran su sesión TCP.

Después de configurada la sesión, el cliente envía una petición HTTP (potencialmente) multilínea al servidor. Mostramos las distintas partes de esta petición en el Ejemplo 5.12.

Ejemplo 5.12. El formato de petición HTTP.

```
method absolute-path (HTTP/1.0|HTTP/1.1)
(directivas opcionales)
```

El método al que hacemos referencia en el Ejemplo 5.12 puede ser de varios tipos; los explicamos en la Tabla 5.4

Tabla 5.4. Métodos de HTTP seleccionados.

Método	Descripción
DELETE	El método DELETE solicita que el servidor elimine la información a la que se hace referencia.
GET	El método GET recupera la información a la que se hace referencia.
HEAD	El método HEAD es similar al método GET, excepto en que la respuesta incluye sólo el encabezamiento, no la información a la que se hace referencia.

(continúa)

⁶ En la práctica, hay muchas variaciones a estos cuatro pasos. La más normal es utilizar pasos adicionales para **multiplex** peticiones de archivos en una sola sesión TCP.

Tabla 5.4. Métodos de HTTP seleccionados. *(Continuación)*

Método	Descripción
POST	El método POST indica que los datos se están transfiriendo al servidor para utilizarse con la información a la que se hace referencia.
PUT	El método PUT solicita al servidor que almacene la información que lo acompaña en la ubicación dada.

La respuesta HTTP es similar a la petición. Mostramos el formato de la respuesta en el Ejemplo 5.13.

Ejemplo 5.13. La respuesta HTTP.

```
[HTTP/1.0|HTTP/1.1] status-code description
(líneas de información adicional)
[data]
```

El código de estado mencionado en el Ejemplo 5.13 puede tener muchos valores posibles. Todos ellos se ajustan a una clasificación global, mostrada en la Tabla 5.5 y mostramos algunos de los códigos de estado más importantes en la Tabla 5.6.

Tabla 5.5. Valores base de códigos de estado de HTTP.

Valor	Descripción
1yx	Informativo.
2yx	Éxito.
3yx	Redirección.
4yx	Error de cliente.
5yx	Error de servidor.

Tabla 5.6. Códigos de estado de HTTP seleccionados.

Código	Descripción
100	Continuar.
200	OK.
204	Sin contenido.
206	Contenido parcial.
301	Trasladado permanentemente.
304	No modificado.

(continúa)

Tabla 5.6. Códigos de estado de HTTP seleccionados. (*Continuación*)

Código	Descripción
400	Petición errónea.
401	No autorizado.
403	Prohibido.
404	No encontrado.
405	Método no permitido.
408	Petición expirada.
500	Error interno de servidor.
501	No implementado.
505	Versión HTTP no soportada.

HTTP sobre SSL (HTTPS, *HTTP over SSL*) es una extensión segura de HTTP. Envía tráfico HTTP a través de la SSL (*Secure Socket Layer*, Capa de *socket* segura) para encriptarlo y proporcionar algo de autenticación para los sistemas finales. HTTPS utiliza el puerto 443 en lugar del típico puerto 80 de HTTP.

La CGI (*Common Gateway Interface*, Interfaz de pasarela común) es una interfaz muy utilizada para ofrecer contenido dinámico a los sitios web. Los programas CGI pueden escribirse en cualquier lenguaje que esté disponible pero normalmente se escriben en Perl y en C. Podemos enviar los parámetros de los programas CGI al *host* como parte del URL con un GET, o de una manera más secreta con un POST. Mostramos un programa CGI sencillo en el Ejemplo 5.14. El ejemplo muestra una página HTML que contiene una lista de todos los parámetros enviados. (No es nada seguro, pero es útil para la comprobación de enlaces a programas CGI para asegurarnos de que estamos pasando los parámetros y valores correctos.)

Ejemplo 5.14. Un programa ejemplo de CGI en Perl.

```
#!/usr/bin/perl
use CGI qw(:standard);

# imprime un encabezamiento http, la sección de html <head>, un
# titular se y un encabezamiento de impresión de norma horizontal
print header
    start_html('Un ejemplo sencillo'),
    h1('Un ejemplo sencillo'),
    hr;

# obtiene una lista de todos los parámetros
@param = param();

# para cada parámetro, imprime su valor
foreach $key (@param) {
    $param = param($key);
    print p,"$key is $param\n";
}
```

Rastro del protocolo HTTP

En esta sección veremos HTTP de tres formas diferentes. Primero veremos un rastro de una sesión de una página web que se está descargando. En segundo lugar veremos una representación ASCII de la descarga de un archivo desde dicha página. Por último, veremos la decodificación de un paquete de una petición GET de HTTP.

El rastro de la sesión de la Figura 5.8 comienza con una búsqueda DNS del servidor web en los paquetes 1 y 2. Los paquetes 3, 5 y 6 son los paquetes que conforman el protocolo de intercambio. Los paquetes del 7 al 15 contienen la primera petición GET y la respuesta que la acompaña. Los paquetes del 17 al 19 representan el protocolo de intercambio de la siguiente petición. Podemos ver el comienzo de una segunda petición GET y la respuesta que la acompaña, pero el resto de la sesión está truncada.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	ppp225-88, thecia.net	resolver1.level3.net	DNS	Standard query request to thecia.net
2	1.693903	resolver1.level3.net	ppp225-88, thecia.net	DNS	Standard query response to 225.218.151.20
3	1.741406	ppp225-88, thecia.net	theCIA.net	TCP	1093 → www [FIN, ACK] Seq=1676612090 Win=0 Len=0
4	1.886567	ppp225-88, thecia.net	resolver1.level3.net	DNS	Standard query PTR 88.225.209.53.in-addr.arpa
5	2.033756	theCIA.net	ppp225-88, thecia.net	TCP	www 1093 [SYN, ACK] Seq=1676612079 Win=0 Len=0
6	2.039921	ppp225-88, thecia.net	theCIA.net	TCP	1093 → www [ACK] Seq=3027813818 Win=0 Len=0
7	2.041298	ppp225-88, thecia.net	theCIA.net	HTTP	GET /users/pate/ HTTP/1.0\r\n\r\n
8	2.193653	resolver1.level3.net	ppp225-88, thecia.net	DNS	Standard query response PTR ppp225-88, thecia.net
9	2.200635	ppp225-88, thecia.net	resolver1.level3.net	DNS	Standard query PTR 53.209.244.209.in-addr.arpa
10	2.377828	theCIA.net	ppp225-88, thecia.net	TCP	www 1093 [ACK] Seq=1676612090 Win=0 Len=0
11	3.167801	theCIA.net	ppp225-88, thecia.net	HTTP	HTTP/1.1 200 OK\r\n\r\n
12	3.167914	ppp225-88, thecia.net	theCIA.net	TCP	1093 → www [ACK] Seq=3027814109 Win=0 Len=0
13	4.847911	theCIA.net	ppp225-88, thecia.net	HTTP	Continuation
14	4.867896	ppp225-88, thecia.net	theCIA.net	TCP	1093 → www [ACK] Seq=3027814109 Win=0 Len=0
15	4.867930	ppp225-88, thecia.net	ppp225-88, thecia.net	DNS	Standard query response PTR resolver1.level3.net
16	4.991702	ppp225-88, thecia.net	nr.lun.net	TCP	1090 → www [FIN, ACK] Seq=1752841057 Win=0 Len=0
17	4.992395	ppp225-88, thecia.net	theCIA.net	TCP	1094 → www [SYN] Seq=3033470986 Win=0 Len=0
18	5.218163	theCIA.net	ppp225-88, thecia.net	TCP	www 1094 [SYN, ACK] Seq=1677283506 Win=0 Len=0
19	5.218254	ppp225-88, thecia.net	theCIA.net	TCP	1094 → www [ACK] Seq=3033470986 Win=0 Len=0
20	5.219238	ppp225-88, thecia.net	theCIA.net	HTTP	GET /users/pate/sflogo-hammer1.jpg HTTP/1.0\r\n\r\n
21	5.287873	nr.lun.net	ppp225-88, thecia.net	TCP	www 1090 [RST] Seq=1582079283 Win=0 Len=0
22	5.587919	theCIA.net	ppp225-88, thecia.net	TCP	www 1094 [ACK] Seq=1677283507 Win=0 Len=0

Figura 5.8. Descarga de una página web.

La Figura 5.9 muestra una representación ASCII de la primera petición GET y su respuesta de la Figura 5.8. El primer bloque representa la petición GET y las directivas pasadas al servidor desde el cliente. El segundo bloque de datos representa el encabezamiento de la respuesta del servidor. El bloque final (el que hemos truncado) muestra los datos reales transferidos entre los dos *hosts*.

La Figura 5.10 muestra el mensaje HTTP que contiene la primera petición HTTP de la Figura 5.8. Es una petición HTTP 1.0; solicita contenido Inglés (*English*) e intenta establecer una conexión.

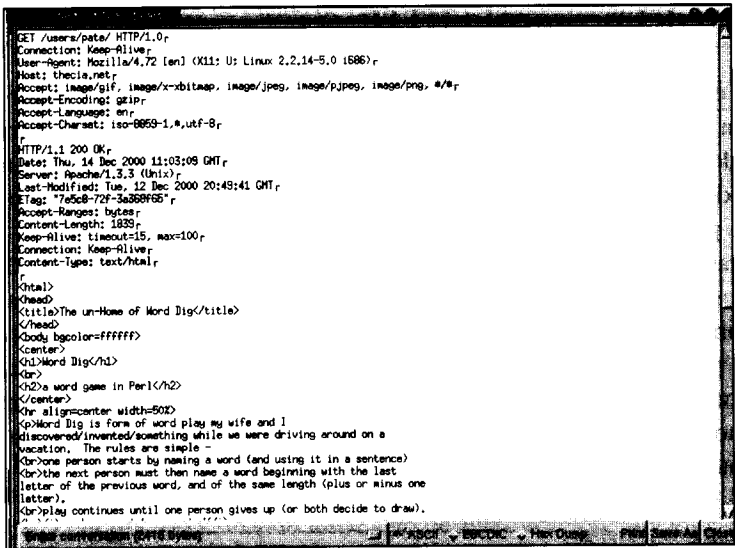


Figura 5.9. Un rastro ASCII de una sesión HTTP.

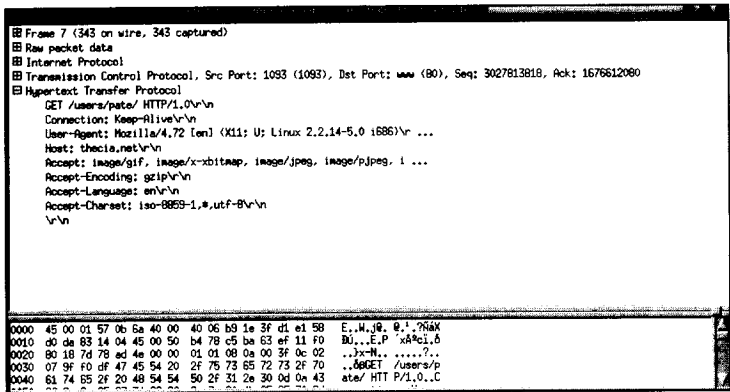


Figura 5.10. Una petición GET.

Cómo utilizar HTTP desde la línea de comandos

Uno de los métodos de interacción (o comprobación) raramente utiliza- dos con un servidor HTTP es una conexión de línea de comandos. De esta forma la conexión con el servidor nos ofrece un buen grado de control sobre la sesión. También nos suministra retroalimentación inmediata sobre los datos que se están devolviendo. En el Ejemplo 5.15 mostramos un ejemplo de una comprobación de una conexión.

Ejemplo 5.15. Cómo utilizar HTTP desde la línea de comandos.

```
[pate@cherry figures]$ telnet thecia.net 80
Trying 208.218.131.20...
Connected to thecia.net.
Escape character is '^]'.
GET /users/pate/ HTTP/1.1
HOST: thecia.net
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Thu, 14 Dec 2000 11:39:54 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Tue, 12 Dec 2000 20:49:41 GMT
ETag: "7e5c8-72f-3a368f65"
Accept-Ranges: bytes
Content-Length: 1839
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<html>
<head>
<title>The un-Home of Word Dig</title>
</head>
<body bgcolor=ffffff>
<center>
<h1>Word Dig</h1>
<br>
<h2>a word game in Perl</h2>
</center>
<hr align=center width=50%>
[resto de salida truncado]
```

En el Ejemplo 5.15 podemos ver tres bloques distintos: la petición HTTP generada por el usuario, la respuesta HTTP generada por el servidor y los datos HTML que se están transfiriendo desde el servidor. En la petición HTTP, utilizamos el parámetro HOST (que es necesario para HTTP/1.1) y el CONNECTION.

La respuesta HTTP contiene cierto número de parámetros interesantes. Los servidores *proxy* HTTP que guardan contenido temporalmente pueden utilizar el parámetro LAST_MODIFIED. El servidor sabe entonces cuándo reemplazar su contenido local por una descarga reciente del servidor HTTP. El parámetro Content-Length permite al proceso cliente saber cuántos datos esperar (se utiliza para actualizar la barra de estado en la mayoría de los navegadores web). El parámetro Keep-Alive tiene dos valores importantes: el valor timeout es el número de segundos que estará abierta la sesión para continuar transmisiones y el valor max es el número total de transferencias de archivos que se pueden tramitar en esta sesión.