

Texto normal (P)		Texto cifrado (C)			Después del descifrado	
Simbólico	Numerico	P^3	$P^3 \pmod{33}$	C^7	$C^7 \pmod{33}$	Simbólico
S	19	6859	28	13492928512	19	S
U	21	9281	21	1801088541	21	U
Z	26	17576	20	12800000000	26	Z
A	01	1	1	1	1	A
N	14	2744	5	78125	14	N
N	14	2744	5	78125	14	N
E	05	125	26	8031810176	5	E

Figura 7-11. Ejemplo del algoritmo RSA.

hubiéramos seleccionado p y $q = 10^{100}$, podríamos tener $n = 10^{200}$, para que cada bloque pueda ser de hasta 664 bits ($2^{664} = 10^{200}$) u 83 caracteres de 8 bits, contra 8 caracteres para el DES.

Cabe señalar que el uso del RSA como lo hemos descrito es semejante a usar DES en modo ECB: el mismo bloque de entrada da el mismo bloque de salida. Por tanto, se requiere alguna forma de encadenamiento para el cifrado de datos. Sin embargo, en la práctica la mayoría de los sistemas basados en RSA usan criptografía de clave pública principalmente para distribuir claves de sesión de una sola vez para su uso con DES, IDEA u otros algoritmos semejantes. El RSA es demasiado lento para poder cifrar grandes volúmenes de datos.

Otros algoritmos de clave pública

Aunque el RSA se usa ampliamente, de ninguna manera es el único algoritmo de clave pública conocido. El primer algoritmo de clave pública fue el de la mochila (Merkle y Hellman, 1978). La idea aquí es que alguien es dueño de una gran cantidad de objetos, todos con pesos diferentes. El dueño cifra el mensaje seleccionando secretamente un subgrupo de los objetos y colocándolos en la mochila. El peso total de los objetos contenidos en la mochila se hace público, así como la lista de todos los objetos posibles. La lista de los objetos que se metieron en la mochila se mantiene en secreto. Con ciertas restricciones adicionales, el problema de determinar una lista posible de los objetos con el peso dado se consideró no computable, y formó la base del algoritmo de clave pública.

El inventor del algoritmo, Ralph Merkle, estaba bastante seguro de que este algoritmo no podía violarse, por lo que ofreció una recompensa de 100 dólares a cualquiera que pudiera descifrarlo. Adi Shamir (la "S" de RSA) pronto lo violó y cobró la recompensa. Sin desmotivarse, Merkle reforzó el algoritmo y ofreció una recompensa de 1000 dólares a quien pudiera violar el nuevo. Ron Rivest (la "R" de RSA) pronto lo descifró y cobró la recompensa. Merkle no se atrevió a ofrecer 10,000 dólares para la siguiente versión, por lo que "A" (Leonard Adleman) no tuvo suerte. Aunque se ha modificado nuevamente, el algoritmo de la mochila no se considera seguro y pocas veces se usa.

Otras esquemas de clave pública se basan en la dificultad para calcular logaritmos discretos (Rabin, 1979). El Gamal (1985) y Schnorr (1991) han inventado algoritmos basados en este principio.

Existen algunos otros esquemas, como los basados en curvas elípticas (Menezes y Vanstone, 1993), pero las tres categorías principales son las basadas en la dificultad para factorizar números grandes, calcular logaritmos discretos y determinar el contenido de una mochila a partir de su peso. Estos problemas se consideran verdaderamente difíciles de resolver porque los matemáticos han estado trabajando en ellos durante años sin adelantos importantes.

7.1.5. Protocolos de validación de identificación

La validación de identificación (*authentication*) es la técnica mediante la cual un proceso comprueba que su compañero de comunicación es quien se supone que es y no un impostor. La verificación de la identidad de un proceso remoto ante un intruso activo malicioso es sorprendentemente difícil y requiere protocolos complejos basados en criptografía. En esta sección estudiaremos algunos de los muchos procesos de validación de identificación que se usan en las redes de cómputo inseguras.

Como nota, alguna gente confunde la autorización con la validación de identificación. La validación de identificación se encarga del asunto de comprobar si realmente nos estamos comunicando con un proceso específico. La autorización se encarga de lo que puede hacer el proceso. Por ejemplo, un proceso cliente se comunica con un servidor de archivos y le dice: "soy el proceso de Sebastián y quiero borrar el archivo *recetas.old*". Desde el punto de vista del servidor de archivos, deben contestarse dos preguntas:

2. ¿Tiene permiso Sebastián de borrar el archivo *recetas.old* (autorización)?

Sólo tras contestar afirmativamente sin ambigüedades ambas preguntas puede llevarse a cabo la acción solicitada. La primera pregunta realmente es la clave. Una vez que el servidor de archivos sabe con quién está hablando, la comprobación de la autorización simplemente es cuestión de buscar entradas en las tablas locales. Por esta razón, nos concentraremos en la validación de identificación en esta sección.

El modelo general que usan todos los protocolos de validación de identificación es éste. Un usuario (en realidad, un proceso) iniciador, digamos Alicia, quiere establecer una conexión segura con un segundo usuario, Benjamín. Alicia y Benjamín a veces se llaman principales, los personajes centrales de nuestra historia. Benjamín es un banquero con el que Alicia quiere hacer negocios. Alicia comienza por enviar un mensaje a Benjamín o a un centro de distribución de claves (KDC) confiable, que siempre es honesto. Siguen varios otros intercambios de mensajes en diferentes direcciones. A medida que se envían estos mensajes, un intruso malicioso, Trudy,¹

Agradezco a Kaufman *et al.*²³ (1995) el que me revelaran su nombre.

puede interceptarlos, modificarlos o reproducirlos para engañar a Alicia y a Benjamín, o simplemente para estropear sus actividades.

No obstante, una vez que se ha completado el protocolo, Alicia está segura de que está hablando con Benjamín, y Benjamín está seguro de que está hablando con Alicia. Es más, en la mayoría de los protocolos, los dos también habrán establecido una clave de sesión secreta para usarla durante la conversación subsiguiente. En la práctica, por razones de desempeño, todo el tráfico de datos se cifra usando criptografía de clave secreta, aunque la criptografía de clave pública se usa ampliamente con los protocolos de validación de identificación mismos y para establecer la clave de sesión.

El objetivo de usar una clave de sesión nueva, seleccionada aleatoriamente, para cada nueva conexión es reducir al mínimo la cantidad de tráfico que es enviado con las claves secretas o públicas del usuario, reducir la cantidad de texto cifrado que puede obtener un intruso, y reducir al mínimo el daño provocado por la caída de un proceso, si su vaciado de memoria cae en las manos equivocadas. Con suerte, la única clave presente en ese momento será la clave de la sesión. Todas las claves permanentes debieron dejarse en blanco después de establecerse la sesión.

Validación de identificación basada en clave secreta compartida

Para nuestro primer protocolo de validación de identificación, supondremos que Alicia y Benjamín ya comparten una clave secreta, K_{AB} . (En los protocolos formales, abreviaremos Alicia como A y Benjamín como B , respectivamente.) Esta clave compartida podría haberse acordado telefónicamente o en persona pero, en cualquier caso, no a través de la (insegura) red.

Este protocolo se basa en un principio encontrado en muchos protocolos de validación de identificación: una parte envía un número aleatorio a la otra, que entonces lo transforma de una manera especial y devuelve el resultado. Tales protocolos se llaman protocolos **reto-respuesta** (*challenge-response*). En éste, y en los siguientes protocolos de validación de identificación, se usará la siguiente notación:

A, B son las identidades de Alicia y Benjamín

Las R_i son los retos, donde el subíndice identifica al retador

Las K_i son claves, donde i indica el dueño; K_S es la clave de la sesión

La secuencia de mensajes de nuestro primer protocolo de validación de identificación de clave compartida se muestra en la figura 7-12. En el mensaje 1, Alicia envía su identidad, A , a Benjamín de manera que la entienda Benjamín. Benjamín, por supuesto, no tiene manera de saber si este mensaje viene de Alicia o de Trudy, por lo que escoge un reto, un número aleatorio grande, R_B , y lo envía a "Alicia" como mensaje 2, en texto normal. Alicia entonces cifra el mensaje con la clave que comparte con Benjamín y envía el texto cifrado, $K_{AB}(R_B)$, en el mensaje 3. Cuando Benjamín ve este mensaje, de inmediato sabe que viene de Alicia porque Trudy no conoce K_{AB} y, por tanto, no pudo haberlo generado. Es más, dado que se seleccionó R_B al azar de un espacio grande (digamos, números aleatorios de 128 bits), es muy poco probable que Trudy haya visto R_B y su respuesta en una sesión previa.

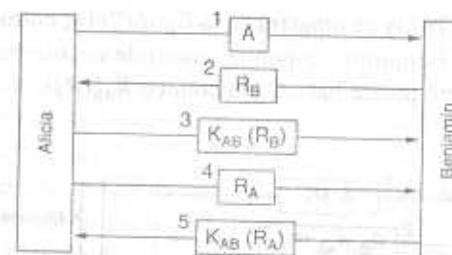


Figura 7-12. Validación de identificación de dos vías usando un protocolo de reto-respuesta.

En este punto, Benjamín está seguro de que está hablando con Alicia, pero Alicia no está segura de nada; hasta donde sabe, Trudy podría haber interceptado el mensaje 1 y enviado R_B como respuesta. Tal vez Benjamín murió anoche. Para saber a quién le está hablando, Alicia selecciona un número al azar, R_A , y lo envía a Benjamín como texto normal, en el mensaje 4. Cuando Benjamín responde con $K_{AB}(R_A)$, Alicia sabe que está hablando con Benjamín. Si desean establecer ahora una clave de sesión, Alicia puede seleccionar una, K_S , y enviársela a Benjamín cifrada con K_{AB} .

Aunque el protocolo de la figura 7-12 funciona, contiene mensajes de más. Estos pueden eliminarse combinando información, como se ilustra en la figura 7-13. Aquí, Alicia inicia el protocolo de reto-respuesta en lugar de esperar que Benjamín lo haga. De la misma manera, al tiempo que responde al reto de Alicia, Benjamín envía el suyo propio. El protocolo completo puede reducirse a tres mensajes en lugar de cinco.

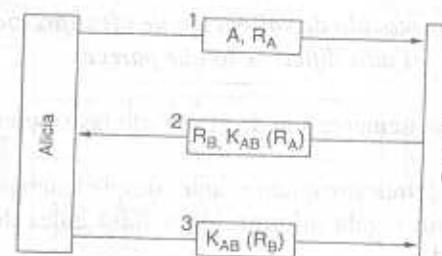


Figura 7-13. Protocolo de validación de identificación de dos vías más corto.

¿Es este nuevo protocolo una mejora sobre el original? En un sentido, sí: es más corto. Desgraciadamente, también está mal. En ciertas circunstancias, Trudy pudo burlar este protocolo usando lo que se conoce como **ataque por reflexión**. En particular, Trudy puede inutilizarlo si es posible abrir al mismo tiempo varias sesiones con Benjamín. Esta situación sería cierta si, por ejemplo, Benjamín es un banco y está preparado para aceptar muchas conexiones simultáneas de los cajeros automáticos.

El ataque por reflexión de Trudy se muestra en la figura 7-14; comienza cuando Trudy indica que ella es Alicia y envía R_T . Benjamín responde, como de costumbre, con su propio reto, R_B . Ahora Trudy está atorada. ¿Qué puede hacer? No conoce $K_{AB}(R_B)$.

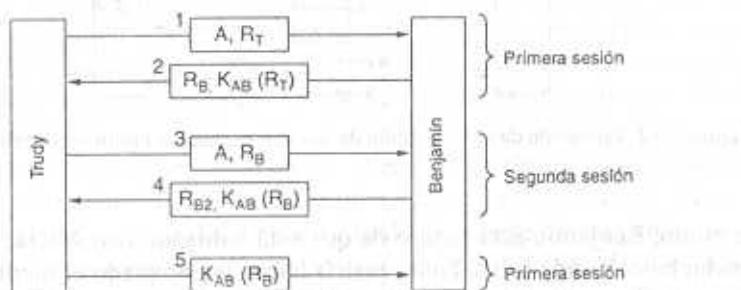


Figura 7-14. Ataque por reflexión.

Trudy puede abrir una segunda sesión con el mensaje 3, proporcionando como su reto el R_B tomado del mensaje 2. Benjamín lo cifra sin chistar y envía de regreso $K_{AB}(R_B)$ en el mensaje 4. Ahora Trudy tiene la información faltante, por lo que puede completar la primera sesión y abortar la segunda. Benjamín ahora está convencido de que Trudy es Alicia, por lo que, cuando ella solicita el saldo de su cuenta bancaria, él se lo da sin más. Después, cuando ella solicita la transferencia de todo a una cuenta secreta de un banco suizo, él lo hace sin dudarlo.

La moraleja de esta historia es:

El diseño de un protocolo de validación de identificación correcto es más difícil de lo que parece.

Tres reglas generales que frecuentemente son de ayuda son las siguientes:

1. Hacer que el iniciador demuestre quién es antes de que lo tenga que hacer el respondedor. En este caso, Benjamín regala información valiosa antes de que Trudy dé cualquier prueba de su identidad.
2. Hacer que el iniciador y el respondedor usen diferentes claves para comprobación, incluso si esto significa tener dos claves compartidas, K_{AB} y K'_{AB} .
3. Hacer que el iniciador y el respondedor tomen sus retos de diferentes conjuntos. Por ejemplo, el iniciador debe usar números pares y el respondedor números nones.

Las tres reglas se violaron aquí, con resultados desastrosos. Nótese que nuestro primer protocolo de validación de identificación (de cinco mensajes) requiere que Alicia compruebe primero su identidad, por lo que ese protocolo no está a merced del ataque por reflexión.

Establecimiento de una clave compartida: intercambio de claves Diffie-Hellman

Hasta ahora hemos supuesto que Alicia y Benjamín comparten una clave secreta. ¿Qué sucede si no es así? ¿Cómo pueden establecer una? Una manera sería hacer que Alicia llame a Benjamín y le dé su clave por teléfono, pero él podría comenzar por decir: "¿Cómo sé que tú eres Alicia y no Trudy?" Ellos podrían intentar concertar una cita, trayendo cada uno un pasaporte, su licencia de conducir y tres tarjetas de crédito bancarias, pero siendo gente ocupada, tal vez no encuentren una fecha aceptable sino hasta dentro de varios meses. Afortunadamente, aunque suene increíble, hay una manera de que completos desconocidos establezcan una clave secreta a plena luz del día, aun con Trudy registrando cuidadosamente cada mensaje.

El protocolo que permite que dos extraños establezcan una clave secreta compartida se llama intercambio de claves Diffie-Hellman (Diffie y Hellman, 1976), y funciona como sigue. Alicia y Benjamín tienen que acordar dos números primos grandes, n y g , donde $(n - 1)/2$ también es primo, y g debe cumplir ciertas condiciones. Estos números pueden ser públicos, por lo que cualquiera de ellos puede escoger n y g y decírselo al otro abiertamente. Ahora, Alicia escoge un número grande (digamos de 512 bits), x , y lo mantiene en secreto. De la misma manera, Benjamín escoge un número secreto grande, y .

Alicia inicia el protocolo de intercambio de claves enviando a Benjamín un mensaje que contiene $(n, g, g^x \bmod n)$, como se muestra en la figura 7-15. Benjamín responde enviando a Alicia un mensaje que contiene $g^y \bmod n$. Ahora Alicia toma el número que Benjamín le envió y lo eleva a la potencia x para obtener $(g^y \bmod n)^x$. Benjamín lleva a cabo una tarea parecida para obtener $(g^x \bmod n)^y$. Por las leyes de la aritmética modular, ambos cálculos arrojan $g^{xy} \bmod n$. Ahora, Alicia y Benjamín comparten una clave secreta, $g^{xy} \bmod n$.

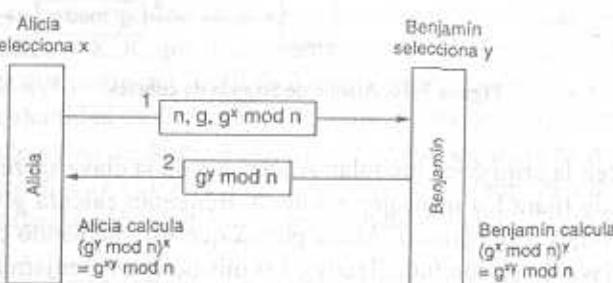


Figura 7-15. Intercambio de claves Diffie-Hellman.

Por supuesto, Trudy ha visto ambos mensajes, así que conoce g y n del mensaje 1. Si Trudy pudiera calcular x y y , podría averiguar la clave secreta. El problema es que, dado sólo $g^x \bmod n$, no es posible encontrar x . No se conoce un algoritmo práctico para calcular logaritmos discretos módulo un número primo muy grande.

Para hacer más concreto el ejemplo anterior, usaremos los valores (completamente irreales) de $n = 47$ y $g = 3$. Alicia selecciona $x = 8$ y Benjamín selecciona $y = 10$. Ambos se mantienen en

secreto. El mensaje de Alicia a Benjamín es $(47, 3, 28)$ porque $3^8 \bmod 47$ es 28. El mensaje de Benjamín a Alicia es (17) . Alicia calcula $17^8 \bmod 47$, que es 4. Benjamín calcula $28^{10} \bmod 47$, que es 4. Alicia y Benjamín han determinado independientemente que la clave secreta ahora es 4. Trudy tiene que resolver la ecuación $3^x \bmod 47 = 28$, lo que puede lograrse mediante una búsqueda exhaustiva en el caso de números pequeños como éste, pero no cuando todos los números tienen cientos de bits de longitud. Todos los algoritmos actualmente conocidos simplemente tardan demasiado, aun usando una supercomputadora masivamente paralela.

A pesar de la elegancia del algoritmo Diffie-Hellman, hay un problema: cuando Benjamín recibe la tripleta $(47, 3, 28)$, ¿cómo sabe que es de Alicia y no de Trudy? No hay manera de saberlo. Desgraciadamente, Trudy puede explotar este hecho para engañar tanto a Alicia como a Benjamín, como se ilustra en la figura 7-16. Aquí, mientras Alicia y Benjamín seleccionan x y y respectivamente, Trudy selecciona su propio número aleatorio, z . Alicia envía el mensaje 1 destinado a Benjamín. Trudy lo intercepta y envía el mensaje 2 a Benjamín, usando el g y el n correctos (que de todas maneras son públicos) pero con su propia z en lugar de x . Trudy también envía el mensaje 3 de regreso a Alicia. Después, Benjamín envía el mensaje 4 a Alicia, que es interceptado y guardado también por Trudy.



Figura 7-16. Ataque de brigada de cubetas.

Ahora todos hacen la aritmética modular. Alicia calcula la clave secreta como $g^{xz} \bmod n$, y también lo hace Trudy (para los mensajes a Alicia). Benjamín calcula $g^{yz} \bmod n$, al igual que Trudy (para los mensajes a Benjamín). Alicia piensa que está hablando con Benjamín, por lo que establece una clave de sesión (con Trudy). Lo mismo hace Benjamín. Cada mensaje que Alicia envía durante la sesión cifrada es capturado por Trudy, almacenado, modificado si ella así lo desea, y pasado (opcionalmente) a Benjamín. Lo mismo ocurre en la otra dirección. Trudy ve todo y puede modificar todos los mensajes si lo desea, mientras que tanto Alicia como Benjamín están pensando equivocadamente que tienen un canal seguro entre ambos. Este ataque se conoce como **ataque de brigada de cubetas**, porque se parece vagamente a las brigadas de bomberos voluntarios antiguas en las que se pasaban las cubetas de agua en fila desde el carro de bomberos hasta el incendio. También se llama **ataque del hombre (mujer) en medio**, lo que no debe confundirse con el encuentro a la mitad de los cifrados de bloque. Afortunadamente, con la ayuda de algoritmos más complicados se puede frustrar este ataque.

Validación de identificación usando un centro de distribución de claves

El establecimiento de un secreto compartido con un extraño casi funcionó, pero no por completo. Por otra parte, probablemente no valía la pena desde el principio (ataque de uvas agrias). Para hablarle a n personas de esta manera se requerirían n claves. Para la gente muy popular, la administración de claves se volvería una verdadera carga, especialmente si cada clave tuviera que almacenarse en el *chip* de una tarjeta de plástico individual.

Un enfoque diferente es introducir un centro de distribución de claves confiable (KDC) (*key distribution center*). En este modelo, cada usuario tiene una sola clave compartida con el KDC. La validación de identificación y la administración de claves de sesión ahora pasan a través del KDC. El protocolo de validación de identificación KDC más simple conocido es la **rana de boca amplia** (Burrows *et al.*, 1990) (llamado así por el apodo que se consiguió su inventor [Burrows] durante sus días en la universidad). Este protocolo se ilustra en la figura 7-17.

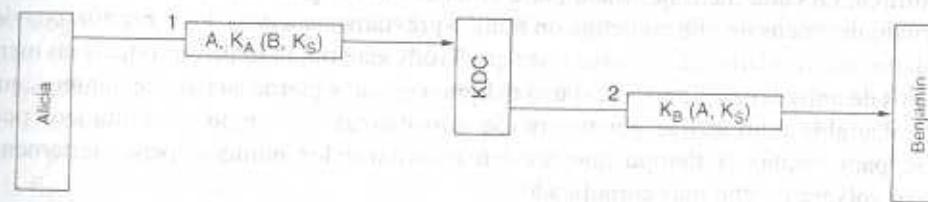


Figura 7-17. Protocolo de validación de identificación de rana de boca amplia.

El concepto en que se basa el protocolo de rana de boca amplia es sencillo: Alicia escoge una clave de sesión, K_S , e indica al KDC que desea hablar con Benjamín usando K_S . Este mensaje se cifra con la clave secreta que comparte Alicia (solamente) con el KDC, K_A . El KDC descifra este mensaje, extrayendo la identidad de Benjamín y la clave de sesión; luego construye un mensaje nuevo que contiene la identidad de Alicia y la clave de sesión y envía este mensaje a Benjamín. Este cifrado se hace con K_B , la clave secreta que Benjamín comparte con el KDC. Cuando Benjamín descifra el mensaje, sabe que Alicia quiere hablar con él, y también conoce la clave que ella desea usar.

La validación de identificación aquí es gratuita. El KDC sabe que el mensaje 1 debe haber venido de Alicia, puesto que nadie más habría sido capaz de cifrarlo con la clave secreta de Alicia. De la misma manera, Benjamín sabe que el mensaje 2 debe haber venido del KDC, en quien confía, puesto que nadie más sabe su clave secreta.

Desgraciadamente, este protocolo tiene una falla seria. Trudy necesita dinero, por lo que piensa en algún servicio legítimo que puede prestar a Alicia, hace una oferta atractiva, y consigue el trabajo. Después de hacerlo, Trudy solicita cortésmente a Alicia que le pague mediante una transferencia bancaria. Alicia entonces establece una clave de sesión con su banquero, Benjamín, y le envía un mensaje solicitando la transferencia del dinero a la cuenta de Trudy.

Mientras tanto, Trudy ha regresado a su viejo hábito, espiar la red; copia tanto el mensaje 2 de la figura 7-17 como la solicitud de transferencia de dinero que le sigue. Posteriormente, Trudy

reproduce ambos para Benjamín. Benjamín los recibe y piensa: "Alicia debió haber contratado nuevamente a Trudy. Aparentemente trabaja bien". Benjamín entonces transfiere una cantidad igual de dinero de la cuenta de Alicia a la de Trudy. Algun tiempo después del par de mensajes número 50, Benjamín sale a buscar a Trudy para ofrecerle un préstamo grande para que pueda expandir su negocio, que evidentemente tiene éxito. Este problema se llama **ataque por repetición** (*replay attack*).

Son posibles varias soluciones al ataque por repetición. La primera es incluir una marca de tiempo en cada mensaje. Entonces, si alguien recibe un mensaje obsoleto, puede descartarlo. El problema con este enfoque es que los relojes nunca están perfectamente sincronizados en toda una red, por lo que tiene que haber un intervalo durante el cual la marca de tiempo sea válida. Trudy puede repetir el mensaje durante este intervalo y salirse con la suya.

La segunda solución es poner un número de mensaje único, de una sola ocasión, a veces llamado **númico**, en cada mensaje. Cada parte entonces tiene que recordar todos los númicos y rechazar cualquier mensaje que contenga un númico previamente usado. Pero los númicos tienen que recordarse indefinidamente, no vaya a ser que Trudy esté tratando de reproducir un mensaje de cinco años de antigüedad. También, si una máquina se cae y pierde su lista de númicos, nuevamente es vulnerable a un ataque por repetición. Las marcas de tiempo y los númicos pueden combinarse para limitar el tiempo que pueden recordarse los númicos, pero ciertamente el protocolo se volverá mucho más complicado.

Un enfoque más refinado para la validación de identificación es usar un protocolo multisentido de reto-respuesta. Un ejemplo bien conocido de tal protocolo es el de **validación de identificación Needham-Schroeder** (Needham y Schroeder, 1978), del cual se muestra una variante en la figura 7-18.

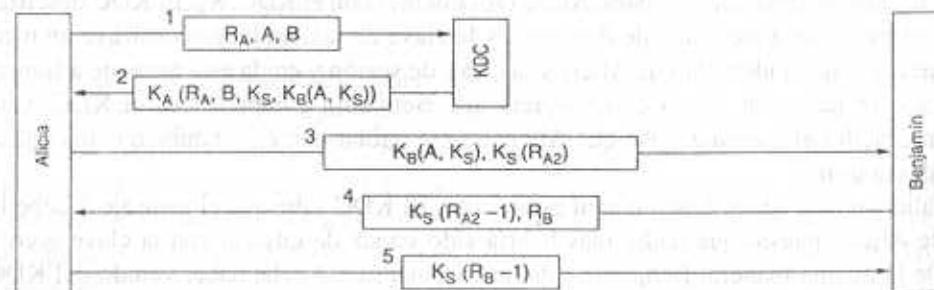


Figura 7-18. Protocolo de validación de identificación Needham-Schroeder.

El protocolo comienza cuando Alicia indica al KDC que quiere hablar con Benjamín. Este mensaje contiene un número aleatorio grande, R_A , como númico. El KDC devuelve el mensaje 2 que contiene el número aleatorio de Alicia, una clave de sesión, y un billete que puede enviar a Benjamín. El objeto del número aleatorio, R_A , es asegurar a Alicia que el mensaje 2 es reciente, y no una repetición. La identidad de Benjamín también se incluye por si a Trudy se le ocurre

reemplazar B del mensaje 1 por su propia identidad, para que el KDC cifre el billete al final del mensaje 2 con K_T en lugar de K_B . El billete cifrado con K_B se incluye dentro del mensaje cifrado para evitar que Trudy lo reemplace por otra cosa en su camino hacia Alicia.

Alicia ahora envía el billete a Benjamín, junto con un nuevo número aleatorio, R_{A2} , cifrado con la clave de la sesión, K_S . En el mensaje 4, Benjamín devuelve $K_S(R_{A2} - 1)$ para demostrar a Alicia que está hablando con el verdadero Benjamín. El envío de regreso de $K_S(R_{A2})$ no habría funcionado, puesto que Trudy lo podría haber robado del mensaje 3.

Tras recibir el mensaje 4, Alicia está convencida de que está hablando con Benjamín, y de que no se pudieron haber usado repeticiones hasta el momento. A fin de cuentas, Alicia acaba de generar R_{A2} hace unos cuantos milisegundos. El propósito del mensaje 5 es convencer a Benjamín de que realmente está hablando con Alicia, y de que tampoco se han usado repeticiones aquí. Al hacer que cada parte genere un reto y responda a éste, se elimina la posibilidad de un ataque por repetición.

Aunque este protocolo parece bastante sólido, tiene una ligera debilidad. Si Trudy llega a obtener una clave de sesión vieja en texto normal, puede iniciar una nueva sesión con Benjamín repitiendo el mensaje 3 correspondiente a la clave obtenida y convencerlo de que ella es Alicia (Denning y Sacco, 1981). Esta vez Trudy puede desfalcar la cuenta de Alicia sin tener que llevar a cabo el servicio legítimo ni siquiera una sola vez.

Needham y Schroeder publicaron posteriormente un protocolo que corrige este problema (Needham y Schroeder, 1987). En el mismo número de la misma publicación, Otway y Rees (1987) publicaron un protocolo que también resuelve el problema pero de una manera más corta. En la figura 7-19 se muestra un protocolo Otway-Rees ligeramente modificado.

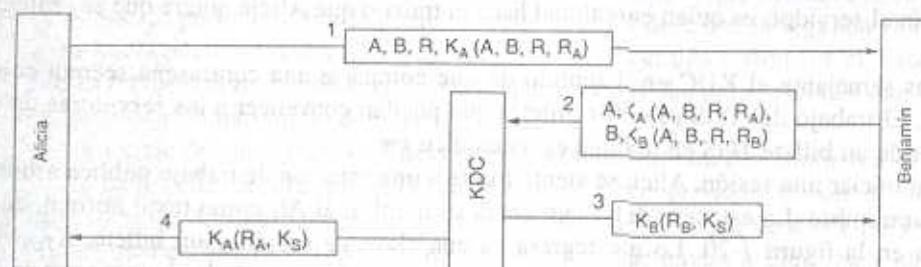


Figura 7-19. Protocolo de validación de identificación Otway-Rees (ligeramente simplificado).

En el protocolo Otway-Rees, Alicia comienza por generar un par de números aleatorios: R , que se usará como identificador común, y R_A , que Alicia usará para retar a Benjamín. Cuando Benjamín recibe este mensaje, construye un mensaje nuevo a partir de la parte cifrada del mensaje de Alicia, y uno análogo propio. Ambas partes cifradas con K_A y K_B identifican a Alicia y a Benjamín, contienen el identificador común y contienen un reto.

El KDC comprueba que el R de ambas partes es igual. Los R podrían no serlo porque Trudy alteró el R del mensaje 1 y reemplazó parte del mensaje 2. Si los dos R son iguales, el KDC se

convence de que el mensaje de solicitud de Benjamín es válido, y genera una clave de sesión y la cifra dos veces, una para Alicia y otra para Benjamín. Cada mensaje contiene el número aleatorio del receptor como prueba de que el KDC, y no Trudy, generó el mensaje. En este punto, tanto Alicia como Benjamín poseen la misma clave de sesión y pueden comenzar a comunicarse. La primera vez que intercambien mensajes de datos, ambos podrán ver que el otro tiene una copia idéntica de K_S , por lo que la validación de identificación está completa.

Validación de identificación usando Kerberos

Un protocolo de validación de identificación usado en muchos sistemas reales es Kerberos, basado en una variación del de Needham-Schroeder; se llama así por un perro de varias cabezas de la mitología griega que solía cuidar la entrada al ayerbo (supuestamente para mantener fuera a los indeseables). Kerberos se diseñó en el M.I.T. para permitir a los usuarios de estaciones de trabajo el acceso a recursos de una manera segura; su principal diferencia respecto del protocolo de Needham-Schroeder es el supuesto de que todos los relojes están bastante bien sincronizados. El protocolo ha pasado por varias iteraciones. La V4 es la versión de uso más difundido en la industria, por lo que la describiremos. Después diremos algunas palabras sobre su sucesor, V5. Para mayor información, véase (Neuman y Ts'o, 1994, y Steiner *et al.*, 1988).

Kerberos comprende tres servidores además de Alicia (una estación de trabajo cliente):

Servidor de validación de identificación (AS): verifica los usuarios durante el establecimiento de la sesión

Servidor de otorgamiento de billetes (TGS): emite "billetes de identidad comprobada"

Benjamín el servidor: es quien en realidad hace el trabajo que Alicia quiere que se realice

El AS es semejante al KDC en el sentido de que comparte una contraseña secreta con cada usuario. El trabajo del TGS es emitir billetes que puedan convencer a los servidores de que el portador de un billete TGS en realidad es quien dice ser.

Para iniciar una sesión, Alicia se sienta frente a una estación de trabajo pública arbitraria y teclea su nombre. La estación de trabajo envía su nombre al AS como texto normal, como se muestra en la figura 7-20. Lo que regresa es una clave de sesión y un billete, $K_{TGS}(A, K_S)$, dirigido al TGS. Estos elementos se empacan juntos y se cifran usando la clave secreta de Alicia, de modo que sólo Alicia puede descifrarlos. No es sino hasta que llega el mensaje 2 que la estación de trabajo solicita la contraseña de Alicia. La contraseña entonces se usa para generar K_A , a fin de descifrar el mensaje 2 y obtener la clave de la sesión y el billete TGS contenido en ella. En este punto, la estación de trabajo sobrescribe la contraseña de Alicia para asegurarse que solamente esté en la estación de trabajo durante unos cuantos milisegundos a lo sumo. Si Trudy intenta establecer una sesión haciéndose pasar por Alicia, la contraseña que ella escriba estará equivocada y la estación de trabajo detectará esto porque la parte estándar del mensaje 2 será incorrecta.

Tras establecer la sesión, Alicia puede indicar a la estación de trabajo que quiere hacer contacto con Benjamín, el servidor de archivos. La estación de trabajo envía entonces el mensaje

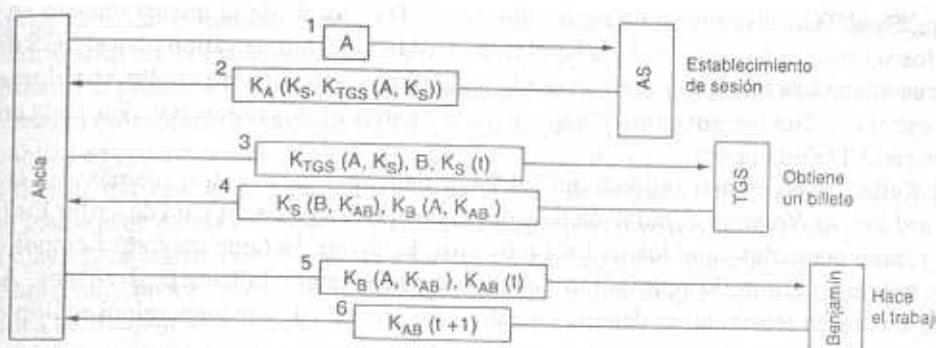


Figura 7-20. Funcionamiento del Kerberos V4.

3 al TGS solicitando un billete para usarlo con Benjamín. El elemento clave de esta solicitud es $K_{TGS}(A, K_S)$, que se cifra con la clave secreta del TGS y sirve como prueba de que el transmisor en realidad es Alicia. El TGS responde creando una clave de sesión, K_{AB} , para que Alicia la use con Benjamín. Se devuelven dos versiones de esta clave. La primera se cifra sólo con K_S , para que Alicia pueda leerla. La segunda se cifra con la clave de Benjamín, K_B , para que Benjamín pueda leerla.

Trudy puede copiar el mensaje 3 e intentar usarlo de nuevo, pero se verá frustrada por la marca de tiempo cifrada, t , que se envía junto con él. Trudy no puede reemplazar la marca de tiempo por una más reciente, pues no conoce K_S , la clave de sesión usada por Alicia para hablar con el TGS. Aun si Trudy repite el mensaje 3 rápidamente, todo lo que obtendrá es otra copia del mensaje 4, que no pudo descifrar la primera vez y no podrá descifrar la segunda vez tampoco.

Ahora Alicia puede enviar K_{AB} a Benjamín para establecer una sesión con él. Este intercambio también recibe una marca de tiempo. La respuesta es prueba para Alicia de que en realidad está hablando con Benjamín, no con Trudy.

Tras esta serie de intercambios, Alicia puede comunicarse con Benjamín con la protección de K_{AB} . Si luego ella decide que necesita hablar con otro servidor, Carolina, simplemente repetirá el mensaje 3 al TGS, sólo que ahora especificará C en lugar de B . El TGS responderá rápidamente con un billete cifrado con K_C que Alicia puede enviar a Carolina y que Carolina aceptará como prueba de que vino de Alicia.

El objeto de todo este trabajo es que ahora Alicia puede acceder a los servidores de toda la red de una manera segura, y que su contraseña nunca ha tenido que atravesar la red. De hecho, la contraseña sólo tuvo que estar en su propia estación de trabajo durante algunos milisegundos. Sin embargo, nótense que cada servidor efectúa su propia autorización. Cuando Alicia presenta su billete a Benjamín, éste simplemente indica a Benjamín quién lo envió. Lo que Alicia tenga permiso de hacer lo determinará Benjamín.

Dado que los diseñadores del Kerberos no esperaban que todo el mundo confiara en un solo servidor de validación de identificación, tomaron las providencias necesarias para tener varios reinos, cada uno con su propio AS y su propio TGS. Si quiere obtener un billete para un servidor de un reino distante, Alicia pedirá a su propio TGS un billete aceptado por el TGS del reino

distante. Si el TGS distante se ha registrado con el TGS local (de la misma manera en que lo hacen los servidores locales), el TGS local le dará a Alicia un billete válido para el TGS distante. Entonces ella podrá trabajar por allá, por ejemplo obteniendo billetes para los servidores remotos de ese reino. Nótese, sin embargo, que para que partes de dos reinos trabajen, cada una debe confiar en el TGS de la otra.

El Kerberos V5 es más refinado que el V4, y tiene más carga extra, además, usa la ASN.1 (*Abstract Syntax Notation 1*, notación de sintaxis abstracta 1) del OSI para describir los tipos de datos y tiene pequeños cambios en los protocolos. Esta versión tiene mayores tiempos de vida de los billetes, permite la renovación de éstos y puede emitir billetes posfechados. Además, cuando menos en teoría, no es dependiente del DES, como V4, y maneja reinos múltiples.

Validación de identificación usando criptografía de clave pública

La validación de identificación mutua también puede hacerse usando criptografía de clave pública. Para comenzar, supongamos que Alicia y Benjamín ya conocen las claves públicas del otro (un asunto nada trivial). Ellos quieren establecer una sesión y luego usar criptografía de clave secreta en esa sesión, ya que típicamente es de 100 a 1000 veces más rápida que la criptografía de clave pública. El propósito de este intercambio inicial entonces es validar la identificación de ambos y acordar una clave de sesión compartida.

Este procedimiento puede hacerse de varias maneras. Una típica se muestra en la figura 7-21. Aquí, Alicia comienza por cifrar su identidad y un número al azar, R_A , usando la clave pública (o de cifrado) de Benjamín, E_B . Cuando Benjamín recibe este mensaje, no sabe si vino de Alicia o de Trudy, pero entra en el juego y devuelve a Alicia un mensaje que contiene el R_A de Alicia, su propio número aleatorio R_B , y la clave de sesión propuesta, K_S .

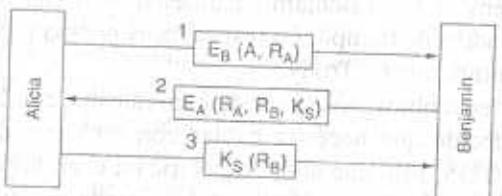


Figura 7-21. Validación de identificación mutua usando criptografía de clave pública.

Cuando Alicia recibe el mensaje 2, lo descifra usando su clave privada y encuentra R_A en él, lo que la hace sentirse cómoda. El mensaje debió haber venido de Benjamín, puesto que Trudy no tiene manera de determinar R_A . Es más, el mensaje debe ser reciente y no una repetición, puesto que ella acaba de enviar R_A a Benjamín. Alicia acepta la sesión devolviendo el mensaje 3. Cuando Benjamín ve R_B cifrado con la clave de sesión que él acaba de generar, sabe que Alicia recibió el mensaje 2 y comprobó R_A .

¿Qué puede hacer Trudy para sabotear este protocolo? Puede crear un mensaje 1 falso y engañar a Benjamín para que sondee a Alicia, pero Alicia verá un R_A que no envió y no continuará.

Trudy no pudo falsificar el mensaje 3 de manera convincente, puesto que no conoce R_B ni K_S y no puede determinarlos sin la clave privada de Alicia. Mala suerte.

No obstante, el protocolo sí tiene una debilidad: supone que Alicia y Benjamín ya conocen la clave pública del otro. Supongamos que no es así. Alicia podría simplemente enviar a Benjamín su clave pública en el primer mensaje y pedir a Benjamín que devuelva la suya en el siguiente. El problema de este enfoque es que está sujeto a un ataque de brigada de cubetas. Trudy puede capturar el mensaje de Alicia a Benjamín y devolver su propia clave a Alicia. Alicia pensará que tiene una clave para hablar con Benjamín cuando, de hecho, tiene una clave para hablar con Trudy. Ahora Trudy puede leer todos los mensajes cifrados con lo que Alicia piensa que es la clave pública de Benjamín.

El intercambio inicial de claves públicas puede evitarse almacenando todas las claves públicas en una base de datos pública. Así, Alicia y Benjamín pueden obtener la clave pública del otro de la base de datos. Desgraciadamente, Trudy aún puede poner en práctica el ataque de brigada de cubetas interceptando las solicitudes a la base de datos y enviando respuestas simuladas que contengan su propia clave. A fin de cuentas, ¿cómo pueden saber Alicia y Benjamín que las respuestas vinieron de la base de datos, y no de Trudy?

Rivest y Shamir (1984) han diseñado un protocolo que frustra el ataque de brigada de cubetas de Trudy. En su **protocolo de interbloqueo**, tras el intercambio de claves públicas, Alicia envía sólo la mitad de su mensaje a Benjamín, digamos, sólo los bits pares (después del cifrado). Benjamín entonces responde con sus bits pares. Tras recibir los bits pares de Benjamín, Alicia envía sus bits noes, y luego lo hace Benjamín.

El truco aquí es que, cuando Trudy recibe los bits pares de Alicia, no puede descifrar el mensaje, aunque tiene la clave privada. En consecuencia, Trudy es incapaz de volver a cifrar los bits pares usando la clave pública de Benjamín. Si Trudy envía basura a Benjamín, el protocolo continuará, pero Benjamín pronto se dará cuenta de que el mensaje reensamblado no tiene sentido y de que ha sido engañado.

7.1.6. Firmas digitales

La validación de identificación de muchos documentos legales, financieros y de otros tipos se determina por la presencia o ausencia de una firma manuscrita autorizada. Las fotocopias no cuentan. Para que los sistemas computarizados de mensajes reemplacen el transporte físico de papel y tinta, debe encontrarse una solución a estos problemas.

El problema de inventar un reemplazo para las firmas manuscritas es difícil. Básicamente, lo que se requiere es un sistema mediante el cual una parte pueda enviar un mensaje "firmado" a otra parte de modo que

1. El receptor pueda verificar la identidad proclamada del transmisor.
2. El transmisor no pueda repudiar después el contenido del mensaje.
3. El receptor no haya podido confeccionar el mensaje él mismo.

El primer requisito es necesario, por ejemplo, en los sistemas financieros. Cuando la computadora de un cliente ordena a la computadora de un banco que compre una tonelada de oro, la

computadora del banco necesita asegurarse de que la computadora que da la orden realmente pertenece a la compañía a la que se le aplicará el débito.

El segundo requisito es necesario para proteger al banco contra fraudes. Supongamos que el banco compra la tonelada de oro, e inmediatamente después cue marca rápidamente el precio del oro. Un cliente deshonesto podría demandar al banco, alegando que nunca emitió una orden para comprar el oro. Cuando el banco presenta un mensaje en la corte, el cliente niega haberlo enviado.

El tercer requisito es necesario para proteger al cliente en el caso de que el precio del oro suba mucho y que el banco trate de falsificar un mensaje firmado en el que el cliente solicitó un lingote de oro en lugar de una tonelada.

Firmas de clave secreta

Un enfoque de las firmas digitales sería tener una autoridad central que sepa todo y en quien todos confíen, digamos el *Big Brother* (*BB*). Cada usuario escoge entonces una clave secreta y la lleva personalmente a las oficinas de *BB*. Por tanto, sólo Alicia y *BB* conocen la clave secreta de Alicia, K_A , etcétera.

Cuando Alicia quiere enviar un mensaje de texto normal firmado, P , a su banquero, Benjamín, genera $K_A(B, R_A, t, P)$ y lo envía como se muestra en la figura 7-22. *BB* ve que el mensaje es de Alicia, lo descifra, y envía un mensaje a Benjamín como se muestra. El mensaje a Benjamín contiene el texto normal del mensaje de Alicia y también el mensaje firmado $K_{BB}(A, t, P)$, donde t es una marca de tiempo. Ahora, Benjamín atiende la solicitud de Alicia.

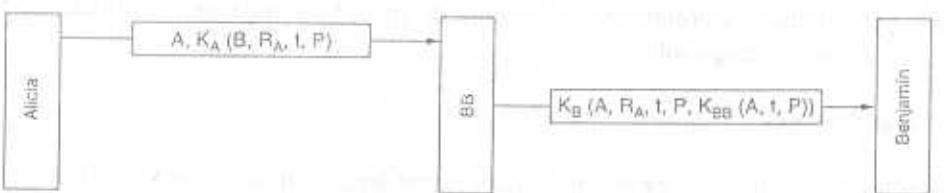


Figura 7-22. Firmas digitales con el *Big Brother*.

¿Qué ocurre si Alicia luego niega el envío del mensaje? El paso 1 es que todos demandan a todos (cuando menos en Estados Unidos). Por último, cuando el caso llega a la corte y Alicia niega haber enviado a Benjamín el mensaje en disputa, el juez pregunta a Benjamín por qué está tan seguro de que el mensaje en disputa vino de Alicia y no de Trudy. Benjamín primero indica que *BB* no aceptaría un mensaje de Alicia a menos que estuviera cifrado con K_A , por lo que no hay posibilidad de que Trudy enviara a *BB* un mensaje falso de Alicia.

Benjamín entonces presenta la prueba $A, K_{BB}(A, t, P)$. Benjamín dice que éste es un mensaje firmado por *BB* que comprueba que Alicia envió P a Benjamín. El juez entonces pide a *BB* (en quien todo el mundo confía) descifrar la prueba A . Cuando *BB* testifica que Benjamín dice la verdad, el juez se pronuncia a favor de Benjamín. Caso cerrado.

Un problema potencial del protocolo de firma de la figura 7-22 es que Trudy repita cualquiera de los dos mensajes. Para minimizar este problema, se usan en todos los intercambios marcas de tiempo. Es más, Benjamín puede revisar todos los mensajes recientes para ver si se usó R_A en cualquiera de ellos. De ser así, el mensaje se descarta como repetición. Nótese que Benjamín rechazará los mensajes muy viejos con base en la marca de tiempo. Para protegerse contra ataques de repetición instantánea, Benjamín simplemente examina el R_A de cada mensaje de entrada para ver si un mensaje igual se recibió de Alicia durante la hora pasada. Si no, Benjamín puede suponer con seguridad que ésta es una solicitud nueva.

Firmas de clave pública

Un problema estructural del uso de la criptografía de clave secreta para las firmas digitales es que todos tienen que confiar en el *Big Brother*. Es más, el *Big Brother* lee todos los mensajes firmados. Los candidatos más lógicos para operar el servidor del *Big Brother* son el gobierno, los bancos y los abogados. Estas organizaciones no inspiran confianza completa a todos los ciudadanos. Por tanto, sería bueno si la firma de documentos no requiriese una autoridad confiable.

Afortunadamente, la criptografía de clave pública puede hacer una contribución importante aquí. Supongamos que los algoritmos públicos de cifrado y descifrado tienen la propiedad de que $E(D(P)) = P$ además de la propiedad normal de $D(E(P)) = P$. (El RSA tiene esta propiedad, por lo que el supuesto es razonable.) Suponiendo que éste es el caso, Alicia puede enviar un mensaje de texto normal firmado, P , a Benjamín transmitiendo $E_B(D_A(P))$. Nótese que Alicia conoce su propia clave de descifrado (privada), D_A , así como la clave pública de Benjamín, E_B , por lo que la construcción de este mensaje es algo que Alicia puede hacer.

Cuando Benjamín recibe el mensaje, lo transforma usando su clave privada, como es normal, produciendo $D_A(P)$, como se muestra en la figura 7-23. Benjamín almacena este texto en un lugar seguro y lo descifra usando E_A para obtener el texto normal original.

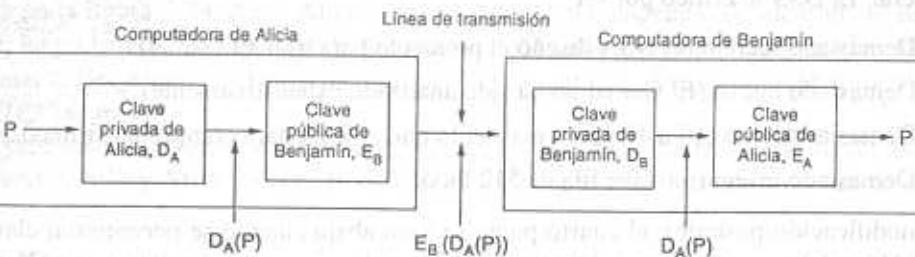


Figura 7-23. Firmas digitales usando criptografía de clave pública.

Para ver cómo funciona la propiedad de firma, supongamos que Alicia posteriormente niega haber enviado el mensaje P a Benjamín. Cuando se presenta el caso en la corte, Benjamín puede presentar tanto P como $D_A(P)$. El juez puede comprobar fácilmente que Benjamín tiene un mensaje válido cifrado por D_A con solo aplicarle E_A . Puesto que Benjamín no conoce la clave

privada de Alicia, la única forma en que Benjamín pudo haber adquirido un mensaje cifrado con ella sería que Alicia en efecto lo hubiera enviado. Mientras está en la cárcel por perjurio y fraude, Alicia tendrá tiempo suficiente para diseñar nuevos e interesantes algoritmos de clave pública.

Aunque el uso de la criptografía de clave pública para las firmas digitales es un esquema elegante, hay problemas relacionados con el entorno en el que opera más que con el algoritmo básico. Por una parte, Benjamín puede demostrar que un mensaje fue enviado por Alicia siempre y cuando D_A permanezca en secreto. Si Alicia divulga su clave secreta, el argumento ya no se mantiene, puesto que cualquiera pudo haber enviado el mensaje, incluido el mismo Benjamín.

El problema podría surgir, por ejemplo, si Benjamín es el corredor de bolsa de Alicia. Alicia le indica a Benjamín que compre ciertas acciones o bonos. Inmediatamente después, el precio cae en picada. Para repudiar su mensaje a Benjamín, Alicia corre a la policía para informarles que robaron su casa y que también sustrajeron su clave. Dependiendo de las leyes de su estado o país, puede o no ser responsable legalmente, especialmente si indica no haber descubierto el robo hasta después de llegar del trabajo, varias horas después.

Otro problema con el esquema de firmas es qué ocurre si Alicia decide cambiar su clave. Hacerlo ciertamente es legal, y probablemente es una buena idea cambiar la clave periódicamente. Si luego surge un caso en la corte, como se describió antes, el juez aplicará la E_A actual a $D_A(P)$ y descubrirá que no produce P . Benjamín quedará en ridículo en ese momento. En consecuencia, parece que sí se requiere alguna autoridad para registrar todos los cambios de clave y sus fechas.

En principio, cualquier algoritmo de clave pública puede usarse para firmas digitales. El estándar *de facto* de la industria es el algoritmo RSA, y muchos productos de seguridad lo usan. Sin embargo, en 1991, el NIST (*National Institute of Standards and Technology*) propuso el uso de una variación del algoritmo de clave pública de El Gamal para su nuevo **Estándar de firmas digitales (DSS)**. La seguridad de El Gamal radica en la dificultad para calcular logaritmos discretos, en lugar de la dificultad para factorizar números grandes.

Como es normal cuando el gobierno intenta dictar estándares criptográficos, hubo una protesta general. El DSS se criticó por ser

1. Demasiado secreto (el NSA diseñó el protocolo para usar El Gamal).
2. Demasiado nuevo (El Gamal no ha sido analizado exhaustivamente).
3. Demasiado lento (10 a 40 veces más lento que el RSA para comprobar firmas).
4. Demasiado inseguro (clave fija de 512 bits).

En una modificación posterior, el cuarto punto se vino abajo cuando se permitieron claves de hasta 1024 bits. Aún no está claro si el DSS echará raíces. Para mayores detalles, véase (Kaufman *et al.*, 1995, Schneier, 1996, y Stinson, 1995).

Compendios de mensaje

Una crítica a los métodos de firma es que con frecuencia reúnen dos funciones: validación de identificación y secreto. En muchos casos se requiere la validación de identificación, pero no del

secreto. Puesto que la criptografía es lenta, a menudo es deseable poder mandar documentos de texto normal firmados. A continuación describiremos un esquema de validación de identificación que no requiere el cifrado del mensaje completo (De Jonge y Chaum, 1987).

Este esquema se basa en la idea de una función de dispersión unidireccional que toma una parte arbitrariamente grande de texto común y a partir de ella calcula una cadena de bits de longitud fija. Esta función de dispersión, llamada **compendio de mensaje** (*message digest*), tiene tres propiedades importantes:

1. Dado P , es fácil calcular $MD(P)$.
2. Dado $MD(P)$, es imposible encontrar P .
3. Nadie puede generar dos mensajes que tengan el mismo compendio de mensaje.

Para cumplir el criterio 3, la dispersión debe ser de cuando menos 128 bits de longitud, y de preferencia mayor.

El cálculo de un compendio de mensaje a partir de un trozo de texto normal es mucho más rápido que el cifrado de ese texto normal con un algoritmo de clave pública, por lo que los compendios de mensaje pueden usarse para acelerar los algoritmos de firma digital. Para ver su funcionamiento, considere nuevamente el protocolo de firma de la figura 7-22. En lugar de firmar P con $K_{BB}(A, t, P)$, BB ahora calcula el compendio de mensaje aplicando MD a P para producir $MD(P)$. BB entonces incluye $K_{BB}(A, t, MD(P))$ como quinto elemento de la lista cifrada con K_B que se envía a Benjamín, en lugar de $K_{BB}(A, t, P)$.

Si surge una disputa, Benjamín puede presentar tanto P como $K_{BB}(A, t, MD(P))$. Una vez que el *Big Brother* lo ha descifrado para el juez, Benjamín tiene $MD(P)$, que está garantizado que es genuino, y el P supuesto. Dado que es prácticamente imposible que Benjamín encuentre otro mensaje que dé esta parcialización, el juez se convencerá fácilmente de que Benjamín dice la verdad. Este uso de compendios de mensaje ahorra tanto tiempo de cifrado como costos de transporte y almacenamiento de mensajes.

Los compendios de mensaje funcionan también en los criptosistemas de clave pública, como se muestra en la figura 7-24. Aquí, Alicia primero calcula el compendio de mensaje de su texto normal; luego firma el compendio de mensaje y envía tanto el compendio firmado como el texto normal a Benjamín. Si Trudy reemplaza P en el camino, Benjamín verá esto cuando calcule $MD(P)$ él mismo.

Se ha propuesto una variedad de funciones de compendio de mensaje. Las de mayor uso son MD5 (Rivest, 1992) y SHA (NIST, 1993). MD5 es la quinta de una serie de funciones de

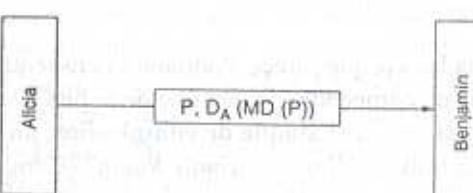


Figura 7-24. Firmas digitales usando compendios de mensaje.

dispersión diseñadas por Ron Rivest; opera alterando los bits de una manera tan complicada que cada bit de salida es afectado por cada bit de entrada. Muy brevemente, comienza por llenar el mensaje a una longitud de 448 bits (módulo 512). Después se agrega la longitud original del mensaje como entero de 64 bits para dar una entrada total cuya longitud es un múltiplo de 512 bits. El último paso de precálculo es la inicialización de un *buffer* de 128 bits a un valor fijo.

Ahora comienza el cálculo. Cada ronda toma un bloque de 512 bits de entrada y lo mezcla por completo con el *buffer* de 128 bits. Por si fuera poco, se introduce también una tabla construida a partir de la función seno. El objetivo de usar una función conocida como el seno no es porque sea más aleatoria que un generador de números aleatorios, sino para evitar cualquier sospecha de que el diseñador construyó una puerta secreta ingeniosa por la que sólo él puede entrar. La negativa de IBM a hacer públicos los principios en que se basó el diseño de las cajas S del DES dio pie a una gran cantidad de especulación sobre las puertas secretas. Se hacen cuatro rondas por cada bloque de entrada. Este proceso continúa hasta que todos los bloques de entrada se han consumido. El contenido del *buffer* de 128 bits forma el compendio de mensaje. El algoritmo ha sido optimizado para su implementación en máquinas de 32 bits. Como consecuencia, podría no ser lo bastante rápido para redes futuras de alta velocidad (Touch, 1995).

La otra función importante de compendio de mensaje es el SHA (*Secure Hash Algorithm*, algoritmo seguro de dispersión), desarrollado por la NSA y bendecido por el NIST. Al igual que el MD5, SHA procesa datos de entrada en bloques de 512 bits pero, a diferencia del MD5, genera un compendio de mensaje de 160 bits. El SHA comienza por llenar el mensaje, y luego agrega una cadena de 64 bits para obtener un múltiplo de 512 bits; por último, inicializa su *buffer* de salida de 160 bits.

Por cada bloque de entrada, el *buffer* de salida se actualiza usando al bloque de entrada de 512 bits. No se usa ninguna tabla de números aleatorios (ni de valores senoidales), pero por cada bloque se calculan 80 rondas, resultando en una mezcla exhaustiva. Cada grupo de 20 rondas usa diferentes funciones de mezcla. Puesto que el código de dispersión de SHA es 32 bits más largo que el de MD5, siendo todo lo demás igual, el primero es 2^{32} más seguro que el MD5; sin embargo, también es más lento que el MD5, y tener un código de dispersión que no es una potencia de dos a veces puede ser un inconveniente. Por lo demás, los dos algoritmos son técnicamente semejantes. Políticamente, el MD5 se define en un RFC y se usa de manera intensiva en Internet. El SHA es un estándar del gobierno, y lo usan compañías que tienen que usarlo porque el gobierno les dice que lo hagan, o aquellas que quieren seguridad extra. Una versión modificada, el SHA-1, ha sido aprobada como estándar por el NIST.

El ataque de cumpleaños

En el mundo de la criptografía, nada es lo que parece. Podríamos pensar que se requieren del orden de 2^m operaciones para subvertir un compendio de mensaje de m bits. De hecho, con frecuencia $2^{m/2}$ operaciones son suficientes si se usa el ataque de cumpleaños, un enfoque publicado por Yuval (1979) en su ahora clásico trabajo "How to Swindle Rabin" (Cómo estafar a Rabin).

La idea de este ataque proviene de una técnica que con frecuencia usan los profesores de matemáticas en sus cursos de probabilidad. La pregunta es: ¿Cuántos estudiantes se necesitan en

un grupo antes de que la probabilidad de tener dos personas con el mismo cumpleaños exceda $1/2$? Muchos estudiantes suponen que la respuesta debe ser mucho mayor que 100. De hecho, la teoría de la probabilidad indica que es de apenas 23. Sin hacer un análisis riguroso, intuitivamente, con 23 personas, podemos formar $(23 \times 22)/2 = 253$ pares diferentes, cada uno de los cuales tiene una probabilidad de $1/365$ de cumplir el requisito. Bajo esta luz, la respuesta ya no es en realidad tan sorprendente.

En términos más generales, si hay alguna correspondencia entre las entradas y las salidas con n entradas (gente, mensajes, etc.) y k salidas posibles (cumpleaños, compendios de mensaje, etc.) hay $n(n - 1)/2$ pares de entradas. Si $n(n - 1)/2 > k$, la posibilidad de que cuando menos una coincida es bastante buena. Por lo tanto, en términos aproximados, es probable una igualación para $n > \sqrt{k}$. Este resultado significa que un compendio de mensaje de 64 bits probablemente puede violarse generando unos 2^{32} mensajes y buscando dos con el mismo compendio de mensaje.

Veamos ahora un ejemplo práctico. El Departamento de Informática de la Universidad Estatal tiene una cátedra para un miembro facultativo y tiene dos candidatos, Tomás y Daniel. Tomás fue contratado dos años antes que Daniel, por lo que su candidatura será considerada antes. Si Tomás obtiene el puesto, mala suerte para Daniel. Tomás sabe que la jefa del departamento, Marilyn, tiene buen concepto de su trabajo, por lo que le pide que escriba una carta de recomendación para el rector, quien decidirá el caso de Tomás. Una vez enviadas, todas las cartas se vuelven confidenciales.

Marilyn le dice a su secretaria, Elena, que escriba una carta al rector, delineando lo que quiere en ella. Cuando está lista, Marilyn la revisará, calculará y firmará el compendio de 64 bits y lo enviará al rector. Elena puede mandar la carta después por correo electrónico.

Desafortunadamente para Tomás, Elena está relacionada románticamente con Daniel y le gustaría dejar fuera a Tomás, por lo que escribe la carta siguiente con las 32 opciones entre corchetes.

Estimado rector Sánchez,

Esta [carta | mensaje] es para dar mi opinión [franca | honesta] sobre el profesor Tomás Guerrero, que [ahora | este año] [es candidato a | está en espera de] una cátedra. He [conocido a | trabajado con] el profesor Guerrero durante [unos | casi] seis años. Es un investigador [sobresaliente | excelente] de gran [talento | habilidad] conocido [mundialmente | internacionalmente] por sus [brillantes | creativas] investigaciones sobre [muchos | una gran variedad de] problemas [difíciles | desafiantes].

Él es también un [profesor | educador] [altamente | grandemente] [respetado | admirado]. Sus estudiantes han hecho evaluaciones [sobresalientes | espectaculares] de sus [clases | cursos]; es el [profesor | instructor] [más admirado | más querido] [del departamento | por nosotros].

[Además | Por otra parte], el profesor Guerrero es [hábil | diligente] para obtener financiamiento. Sus [subvenciones | contratos] han aportado una cantidad [importante | sustancial] de dinero [al | a nuestro] departamento. [Este dinero ha | Estos fondos han] [posibilitado | permitido] que [emprendamos | pongamos en práctica] muchos programas [especiales | importantes], [como | por ejemplo] su programa Estado 2000. Sin estos fondos [seríamos incapaces de | no podríamos] continuar este programa, que es tan [importante | esencial] para ambos. Recomiendo encarecidamente que se le otorgue la cátedra.

Desgraciadamente para Tomás, tan pronto como Elena termina de redactar y mecanografiar esta carta, también escribe una segunda:

Esta [carta | mensaje] es para dar mi opinión [franca | honesta] sobre el profesor Tomás Guerrero, que [ahora | este año] [es candidato a | está en espera de] una cátedra. He [conocido a | trabajado con] el profesor Guerrero durante [unos | casi] seis años. Es un investigador [malo | mediocre] poco conocido en su [campo | área]. Sus investigaciones [casi nunca | pocas veces] muestran [entendimiento | comprensión] de los problemas [clave | principales] [del día | de nuestros tiempos].

Es más, no es un [profesor | educador] [respetado | admirado]. Sus estudiantes han hecho evaluaciones [pobres | malas] de sus [clases | cursos]; es el [maestro | instructor] menos querido [del departamento | por nosotros], conocido [principalmente | más] en [el | nuestro] departamento por su [tendencia | propensión] a [ridiculizar | avergonzar] a los estudiantes lo bastante [tontos | imprudentes] como para hacer preguntas durante su clase.

[Además | Por otra parte], el profesor Guerrero no es [hábil | diligente] para obtener financiamiento. Sus [subvenciones | contratos] han aportado una cantidad [insustancial | insignificante] de dinero [al | a nuestro] departamento. A menos que [se recolecte dinero nuevo | se consigan fondos nuevos] pronto tendremos que cancelar algunos programas esenciales, como su programa Estado 2000. Desgraciadamente, en estas [condiciones | circunstancias] no puedo recomendarlo de buena [conciencia | fe] a usted para [la cátedra | un puesto permanente].

Ahora Elena prepara su computadora para calcular los 2^{32} compendios de mensaje para cada carta durante la noche. Con suerte, un compendio de la primera carta será igual a un compendio de la segunda. De no serlo puede agregar algunas opciones más e intentar de nuevo durante el fin de semana. Supongamos que encuentra una correspondencia. Llamémos a la carta "buena" A y a la "mala" B.

Ahora Elena envía la carta A a Marilyn para su aprobación. Marilyn, por supuesto, la aprueba, calcula su compendio de mensaje de 64 bits, firma el compendio y manda por correo electrónico el compendio firmado al rector Sánchez. Independientemente, Elena envía la carta B al rector.

Al recibir la carta y el compendio de mensaje firmado, el rector ejecuta el algoritmo de compendio de mensaje con la carta B, ve que coincide con lo que Marilyn le envió, y despidie a Tomás. (Final opcional: Elena le dice a Daniel lo que hizo. Daniel se horroriza y rompe con ella. Elena se pone furiosa y confiesa su falta a Marilyn. Marilyn llama al rector. Tomás consigue la cátedra a fin de cuentas.) Con el MD5, el ataque de cumpleaños es infactible porque aun a una velocidad de mil millones de compendios por segundo, se requerirían más de 500 años para calcular los 2^{64} compendios de dos cartas con 64 variantes cada una, e incluso entonces no se garantiza una equivalencia.

7.1.7. Aspectos sociales

Las implicaciones de la seguridad de las redes para la confidencialidad individual y social en general son aterradoras. A continuación mencionaremos unos cuantos de los temas sobresalientes.

A los gobiernos no les gusta que los ciudadanos guarden secretos. En algunos países (por ejemplo, Francia), toda la criptografía no gubernamental simplemente está prohibida a menos que el gobierno tenga todas las claves empleadas. Como apuntan Kahn (1980) y Selfridge y Schwartz (1980), la práctica gubernamental de escuchar secretamente se practica en una escala mucho más masiva de lo que podría soñar la mayoría de la gente, y el gobierno quiere más que simplemente una pila de bits indescifrables a cambio de sus esfuerzos.

El gobierno de Estados Unidos ha propuesto un esquema de cifrado para los teléfonos digitales futuros que incluye una característica especial para permitir a la policía la intervención y descifrado de todas las llamadas telefónicas hechas en ese país. El gobierno promete no usar esta característica sin una orden judicial, pero muchos aún se acuerdan de la manera en que el director de FBI, J. Edgar Hoover, intervino ilegalmente los teléfonos de Martin Luther King, Jr. y otros en un intento por neutralizarlos. La policía dice que necesita este poder para atrapar criminales. El debate entre ambos lados es airoso, por llamarlo de alguna manera. En (Kaufman et al., 1995) se presenta un estudio de la tecnología en cuestión (Clipper). Una manera de sortear esta tecnología y enviar mensajes que el gobierno no pueda leer se describe en (Blaze, 1994, y Schneier, 1996). En (Hoffman, 1995) se presentan las posturas de todas las partes del debate.

Estados Unidos tiene una ley (22 U.S.C. 2778) que prohíbe a los ciudadanos la exportación de material de guerra, como tanques y aviones de combate, sin autorización del Departamento de la Defensa. Para los fines de esta ley, el software criptográfico se clasifica material de guerra. Phil Zimmermann, quien escribió PGP (*Pretty Good Privacy*, confidencialidad bastante buena), un programa de protección de correo electrónico, ha sido acusado de violar esta ley, aunque el gobierno reconoce que Zimmermann no exportó el programa (pero lo dio a un amigo que lo puso en Internet, donde los extranjeros lo podían obtener). Muchos consideraron este incidente como una violación grave de los derechos de un ciudadano estadounidense que trabaja para proteger la confidencialidad de la gente.

No ser ciudadano estadounidense tampoco ayuda. El 9 de julio de 1986 tres investigadores del Instituto Weizmann de Israel solicitaron una patente en Estados Unidos para un nuevo esquema de firma digital de su invención. Durante los siguientes seis meses los investigadores presentaron su trabajo en conferencias por todo el mundo. El 6 de enero de 1987, la oficina de patentes les indicó que notificaran a todos los estadounidenses que conocían sus resultados que la divulgación de las investigaciones podría hacerlos acreedores a dos años de prisión, una multa de 10,000 dólares, o ambas cosas. La oficina de patentes también quería una lista de todos los extranjeros que sabían algo sobre las investigaciones. Para conocer el final de esta historia, véase (Landau, 1988).

Las patentes son otro tema candente. Casi todos los algoritmos de clave pública están patentados. La protección de patentes dura 17 años. La patente del RSA, por ejemplo, vence el 20 de septiembre del 2000.

La seguridad de las redes tiene más implicaciones políticas que casi ningún otro tema técnico, y con justa razón, puesto que se relaciona con la diferencia entre una democracia y un estado policiaco en la era digital. Las ediciones de marzo de 1993 y noviembre de 1994 de *Communications of the ACM* incluyen amplias secciones sobre la seguridad telefónica y de redes, respectivamente, con argumentos que explican y defienden muchos puntos de vista. El capítulo

25 del libro de seguridad de Schneier trata la política de la criptografía (Schneier, 1996) lo mismo que el capítulo 8 de su libro de correo electrónico (Schneier, 1995). La confidencialidad y las computadoras también se estudian en (Adam, 1995). Estas referencias son muy recomendables para los lectores que deseen ampliar sus estudios sobre este tema.

7.2. DNS — SISTEMA DE NOMBRES DE DOMINIO

Los programas pocas veces hacen referencia a los *hosts*, buzones de correo y otros recursos por sus direcciones binarias de red. En lugar de números binarios, los programas usan cadenas ASCII, como *tana@art.ucsb.edu*. Sin embargo, la red misma sólo entiende direcciones binarias, por lo que se requiere algún mecanismo para convertir las cadenas ASCII en direcciones de red. En las siguientes secciones estudiaremos la manera en que se logra esta correspondencia en Internet.

Hace mucho, en los tiempos del ARPANET, simplemente había un archivo, *hosts.txt*, en el que se listaban todos los *hosts* y sus direcciones IP. Cada noche, todos los *hosts* obtenían este archivo de la instalación en la que se mantenía. En una red de unas cuantas máquinas grandes de tiempo compartido, este enfoque funcionaba razonablemente bien.

Sin embargo, cuando miles de estaciones de trabajo se conectaron a la red, todos se dieron cuenta de que este enfoque no podría continuar funcionando eternamente. Por una parte, el archivo se volvería demasiado grande. Un problema aún más importante era que ocurrirían conflictos constantes con los nombres de los *hosts* a menos de que los nombres se administraran centralmente, algo impensable en una red internacional enorme. Para resolver estos problemas, se inventó el DNS (*Domain Name System*, sistema de nombres de dominio).

La esencia del DNS es la invención de un esquema de nombres jerárquico basado en dominio y una base de datos distribuida para implementar este esquema de nombres. El DNS se usa principalmente para relacionar las direcciones de *host* y destinos de correo electrónico con las direcciones IP, pero también puede usarse con otros fines. El DNS se define en los RFC 1034 y 1035.

Muy brevemente, el modo de usar el DNS es el siguiente. Para relacionar un nombre con una dirección IP, un programa de aplicación llama a un procedimiento de biblioteca llamado resovedor, pasándole el nombre como parámetro. El resovedor envía un paquete UDP a un servidor DNS local, que entonces busca el nombre y devuelve la dirección IP al resovedor, que entonces lo devuelve al solicitante. Con la dirección IP, el programa puede entonces establecer una conexión TCP con el destino, o enviarle paquetes UDP.

7.2.1. El espacio de nombres del DNS

La administración de un grupo grande y continuamente cambiante de nombres es un problema nada sencillo. En el sistema postal, la administración de nombres se hace requiriendo letras para especificar (implícita o explícitamente) el país, estado o provincia, ciudad y calle, y dirección del destinatario. Con este tipo de direccionamiento jerárquico, no hay confusión entre el Marvin Anderson de Main St., en White Plains, N.Y. y el Marvin Anderson de Main St., en Austin, Texas. El DNS funciona de la misma manera.

Conceptualmente, la Internet se divide en varios cientos de dominios de nivel superior, cada uno de los cuales abarca muchos *hosts*. Cada dominio se divide en subdominios, y éstos se dividen nuevamente, etc. Todos estos dominios pueden representarse mediante un árbol, como se muestra en la figura 7-25. Las hojas del árbol representan los dominios que no tienen subdominios (pero que, por supuesto, contienen máquinas). Un dominio de hoja puede contener un solo *host*, o puede representar a una compañía y contener miles de *hosts*.

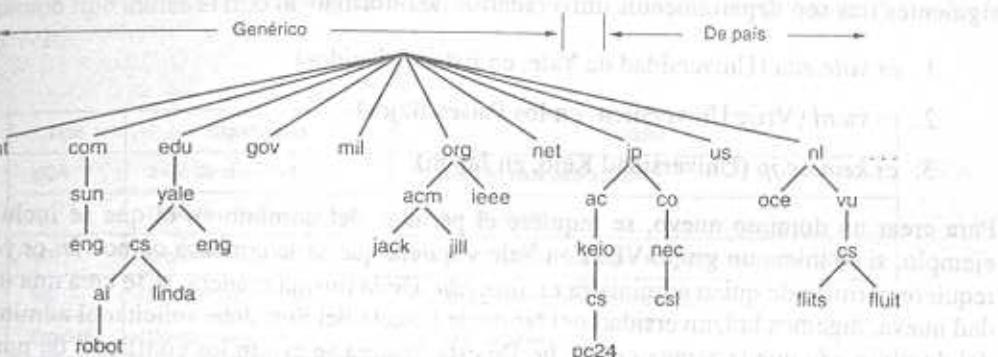


Figura 7-25. Parte del espacio de nombres de dominio de Internet.

Los dominios de nivel superior vienen en dos sabores: genéricos y de país. Los dominios genéricos son *com* (comercial), *edu* (instituciones educativas), *gov* (el gobierno federal de Estados Unidos), *int* (ciertas organizaciones internacionales), *mil* (las fuerzas armadas de Estados Unidos), *net* (proveedores de red) y *org* (organizaciones no lucrativas). Los dominios de país incluyen una entrada para cada país, como se define en el ISO 3166.

Cada dominio se nombra por la trayectoria hacia arriba desde él a la raíz (sin nombre). Los componentes se separan por puntos. Por tanto, el departamento de ingeniería de Sun Microsystems podría ser *eng.sun.com*, a diferencia de un nombre tipo UNIX como */com/sun/eng*. Nótese que este nombramiento jerárquico significa que *eng.sun.com* no entra en conflicto con un uso potencial de *eng* en *eng.yale.edu*, que podría usarse en el departamento de inglés de Yale.

Los nombres de dominio pueden ser absolutos o relativos. Un nombre de dominio absoluto termina con un punto (por ejemplo, *eng.sun.com.*), y uno relativo no. Los nombres relativos tienen que interpretarse en algún contexto para determinar de manera única su significado verdadero. En ambos casos, un dominio nombrado se refiere a un nodo específico del árbol y a todos los nodos por debajo de él.

Los nombres de dominio no hacen distinción entre las mayúsculas y las minúsculas, por lo que *edu* y *EDU* significan la misma cosa. Los nombres de componentes pueden ser de hasta 63 caracteres de longitud, y los nombres de trayectoria completa no deben exceder los 255 caracteres.

En principio, los dominios pueden introducirse en el árbol de tres maneras diferentes. Por ejemplo, *cs.yale.edu* podría estar listado también bajo el dominio de país *us* como *cs.yale.ct.us*.

En la práctica, casi todas las organizaciones de Estados Unidos están bajo un dominio genérico, y casi todas las de fuera de Estados Unidos están bajo el dominio de su país. No hay ninguna regla contra el registro bajo dos dominios de nivel superior, pero hacerlo podría causar confusión, por lo que pocas organizaciones lo hacen.

Cada dominio controla el modo de asignación de los dominios que están debajo de él. Por ejemplo, Japón tiene los dominios *ac.jp* y *co.jp* que son espejos de *edu* y *com*. Los Países Bajos no hacen esta distinción y ponen a todas las organizaciones directamente bajo *nl*. Por tanto, los siguientes tres son departamentos universitarios de informática:

1. *cs.yale.edu* (Universidad de Yale, en Estados Unidos)
2. *cs.vu.nl* (Vrije Universiteit, en los Países Bajos)
3. *cs.keio.ac.jp* (Universidad Keio, en Japón)

Para crear un dominio nuevo, se requiere el permiso del dominio en el que se incluirá. Por ejemplo, si se inicia un grupo VLSI en Yale y quiere que se le conozca como *vlsi.cs.yale.edu*, requiere permiso de quien administra *cs.yale.edu*. De la misma manera, si se crea una universidad nueva, digamos la Universidad del Norte de Dakota del Sur, debe solicitar al administrador del dominio *edu* que le asigne *unsd.edu*. De esta manera se evitan los conflictos de nombres y cada dominio puede llevar el registro de todos sus subdominios, como *cs.unsd.edu*, sin obtener el permiso de nadie más arriba en el árbol.

Los nombres reflejan los límites organizacionales, no las redes físicas. Por ejemplo, si los departamentos de informática e ingeniería electrónica se ubican en el mismo edificio y comparten la misma LAN, de todas maneras pueden tener dominios diferentes. De la misma manera, si el departamento de informática está dividido entre el edificio Babbage y el edificio Turing, todos los *hosts* de ambos edificios pertenecerán, normalmente, al mismo dominio.

7.2.2. Registros de recursos

Cada dominio, sea un *host* individual o un dominio de nivel superior, puede tener un grupo de registros de recursos asociados a él. En un *host* individual, el registro de recursos más común es simplemente su dirección IP, pero también existen muchos otros tipos de registros de recursos. Cuando un resolvelor da un nombre de dominio al DNS, lo que recibe son los registros de recursos asociados a ese nombre. Por tanto, la función real del DNS es relacionar los dominios de nombres con los registros de recursos.

Un registro de recursos tiene cinco tuplas. Aunque éstas se codifican en binario por cuestión de eficiencia, en la mayoría de las presentaciones los registros de recursos se dan como texto ASCII, una línea por registro de recurso. El formato que usaremos es el siguiente:

Nombre_dominio Tiempo_de_vida Tipo Clase Valor

El *nombre_dominio* indica el dominio al que pertenece este registro. Normalmente existen muchos registros por dominio y cada copia de la base de datos contiene información de muchos

dominios. Por tanto, este campo es la llave primaria de búsqueda usada para atender las consultas. El orden de los registros en la base de datos no es significativo. Cuando se hace una consulta relativa a un dominio, se devuelven todos los registros coincidentes de la clase solicitada.

El campo de *tiempo_de_vida* es una indicación de la estabilidad del registro. La información altamente estable recibe un valor grande, como 86400 (la cantidad de segundos en un día). La información altamente volátil recibe un valor pequeño, como 60 (1 minuto). Regresaremos a este punto después de haber estudiado el proceso de caché.

El campo *tipo* indica el tipo de registro de qué se trata. Los tipos más importantes se listan en la figura 7-26.

Tipo	Significado	Valor
SOA	Inicio de autoridad	Parámetros para esta zona
A	Dirección de IP de un host	Entero de 32 bits
MX	Intercambio de correo	Prioridad, dominio dispuesto a aceptar correo electrónico
NS	Servidor de nombres	Nombre de un servidor para este dominio
CNAME	Nombre canónico	Nombre de dominio
PTR	Apuntador	Alias de una dirección IP
HINFO	Descripción del host	CPU y SO en ASCII
TXT	Texto	Texto ASCII no interpretado

Figura 7-26. Tipos principales de registro de recurso DNS.

Un registro *SOA* proporciona el nombre de la fuente primaria de información sobre la zona del servidor de nombres (que se describe más adelante), la dirección de correo electrónico de su administrador, un número de serie único y varias banderas y temporizadores.

El tipo de registro más importante es el registro *A* (dirección) que contiene una dirección IP de 32 bits de algún *host*. Cada *host* de Internet debe tener cuando menos una dirección IP, para que otras máquinas puedan comunicarse con él. Algunos *hosts* tienen dos o más conexiones de red, en cuyo caso tendrán un registro de recurso tipo *A* por cada conexión de red (y, por tanto, por cada dirección IP).

El siguiente tipo de registro más importante es el registro *MX*, que especifica el nombre del dominio que está preparado para aceptar correo electrónico del dominio especificado. Un uso común de este registro es permitir que una máquina que no está en Internet reciba correo electrónico de las instalaciones Internet. La entrega se logra haciendo que la instalación no Internet establezca un arreglo con alguna instalación Internet para que acepte correo electrónico dirigido a ella y lo reenvíe usando cualquier protocolo acordado por las dos.

Por ejemplo, supóngase que Cathy se graduó en informática en la UCLA. Tras recibir su título en informática, ella inicia una compañía, la Electrobrain Corporation, para comercializar sus ideas. No puede aún darse el lujo de una conexión con Internet, por lo que llega a un arreglo

con UCLA para recibir su correo electrónico ahí. Varias veces al día puede llamar y consultar su correo.

Acto seguido, Cathy registra su compañía con el dominio *com* y se le asigna el dominio *electrobrain.com*. Luego ella podría pedir al administrador del dominio *com* que agregue un registro *MX* a la base de datos *com*, a saber:

```
electrobrain.com 86400 IN MX 1 mailserver.cs.ucla.edu
```

De esta manera, el correo se reenviará a UCLA donde Cathy puede recogerlo estableciendo una conexión. Como alternativa, la UCLA podría llamarla y transferir el correo electrónico mediante cualquier protocolo acordado mutuamente.

Los registros *NS* especifican servidores de nombres. Por ejemplo, cada base de datos DNS normalmente tiene un registro *NS* por cada dominio de nivel superior, de modo que el correo electrónico pueda enviarse a partes alejadas del árbol de nombres. Regresaremos a este punto más adelante.

Los registros *CNAME* permiten la creación de alias. Por ejemplo, una persona familiarizada con los nombres de Internet en general que quiere enviar un mensaje a alguien cuya clave de acceso es *paul* y está en el departamento de informática del M.I.T. podría adivinar que *paul@cs.mit.edu* funcionará. De hecho, esta dirección no funciona, puesto que el dominio del departamento de informática del M.I.T. es *lcs.mit.edu*. Sin embargo, como servicio para la gente que no sabe esto, el M.I.T. podría crear una entrada *CNAME* para encaminar a la gente y a los programas en la dirección correcta. La entrada podría ser como ésta:

```
cs.mit.edu 86400 IN CNAME lcs.mit.edu
```

Al igual que *CNAME*, *PTR* apunta a otro nombre. Sin embargo, a diferencia de *CNAME*, que en realidad es sólo una definición de macro, *PTR* es un tipo de datos DNS normal, cuya interpretación depende del contexto en que se encontró. En la práctica, *PTR* casi siempre se usa para asociar un nombre a una dirección IP a fin de permitir búsquedas de la dirección IP y devolver el nombre de la máquina correspondiente.

Los registros *HINFO* permiten que la gente conozca el tipo de máquina y sistema operativo al que corresponde un dominio. Por último, los registros *TXT* permiten a los dominios identificarse de modos arbitrarios. Ambos tipos de registro son para el provecho de los usuarios. Ninguno es obligatorio, por lo que los programas no pueden contar con que los recibirán (y probablemente no puedan manejarlos si los obtienen).

Regresando a la estructura general de los registros de recurso, el cuarto campo de cada registro de recurso es la *clase*. Para la información de Internet, es siempre *IN*. Para la información que no pertenece a Internet, pueden utilizarse otros códigos.

Por último, llegamos al campo de *valor*. Este campo puede ser un número, un nombre de dominio o una cadena ASCII. La semántica depende del tipo de registro. En la figura 7-26 se presenta una descripción corta de los campos de *valor* para cada uno de los tipos principales de registro.

Como ejemplo del tipo de información que podría encontrarse en la base de datos de un dominio, véase la figura 7-27. En esta figura se ilustra una parte de una base de datos (semihipotética)

; Datos autorizados correspondientes a cs.vu.nl			
cs.vu.nl.	86400	IN SOA	star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.	86400	IN TXT	"Faculteit Wijskunde en Informatica."
cs.vu.nl.	86400	IN TXT	"Vrije Universiteit Amsterdam."
cs.vu.nl.	86400	IN MX	1 zephyr.cs.vu.nl.
cs.vu.nl.	86400	IN MX	2 top.cs.vu.nl.
flits.cs.vu.nl.	86400	IN HINFO	Sun Unix
flits.cs.vu.nl.	86400	IN A	130.37.16.112
flits.cs.vu.nl.	86400	IN A	192.31.231.165
flits.cs.vu.nl.	86400	IN MX	1 flits.cs.vu.nl.
flits.cs.vu.nl.	86400	IN MX	2 zephyr.cs.vu.nl.
flits.cs.vu.nl.	86400	IN MX	3 top.cs.vu.nl.
www.cs.vu.nl.	86400	IN CNAME	star.cs.vu.nl
ftp.cs.vu.nl.	86400	IN CNAME	zephyr.cs.vu.nl
rowboat		IN A	130.37.56.201
		IN MX	1 rowboat
		IN MX	2 zephyr
		IN HINFO	Sun Unix
little-sister		IN A	130.37.62.23
		IN HINFO	Mac Mac OS
laserjet		IN A	192.31.231.218
		IN HINFO	"HP Laserjet IIISI" Proprietary

Figura 7-27. Parte de una posible base de datos DNS para *cs.vu.nl*.

correspondiente al dominio *cs.vu.nl* mostrado en la figura 7-25. La base de datos contiene siete tipos de registros de recurso.

La primera línea no de comentario de la figura 7-27 da un poco de información básica sobre el dominio, de lo que ya no nos ocuparemos. Las dos siguientes líneas dan información textual sobre la localización del dominio. Luego vienen dos entradas que dan el primer y segundo lugar a donde se intentará entregar correo electrónico dirigido a *person@cs.vu.nl*. La *zephyr* (una máquina específica) debe intentarse primero. Si falla, debe intentarse la *top* después.

Tras la línea en blanco, que se agregó para hacer más clara la lectura, siguen líneas que indican que la *flits* es una estación de trabajo Sun que ejecuta UNIX, y dan sus dos direcciones IP. Después se indican tres posibilidades para manejar el correo electrónico enviado a *flits.cs.vu.nl*. La primera opción naturalmente es la *flits* misma, pero si está inactiva, la *zephyr* y la *top* son la segunda y tercera opciones. Luego viene un alias, *www.cs.vu.nl*, para que esta dirección pueda usarse sin designar una máquina específica. La creación de este alias permite a *cs.vu.nl* cambiar su servidor del World Wide Web sin invalidar la dirección que la gente usa para llegar a él. Un argumento parecido es válido para *ftp.cs.vu.nl*.

Las siguientes cuatro líneas contienen una entrada típica de una estación de trabajo, en este caso, *rowboat.cs.vu.nl*. La información proporcionada contiene la dirección IP, los destinos de correo primarios y secundarios, e información sobre la máquina. Luego viene una entrada para un sistema no UNIX incapaz de recibir correo por sí mismo, seguida de una entrada para la impresora láser.

Lo que no se muestra (y no está en este archivo) son las direcciones IP a usar para buscar los dominios de nivel superior. Éstas se requieren para buscar *hosts* distantes pero, dado que no son parte del dominio *cs.vu.nl*, no están en este archivo. Tales direcciones son suministradas por los servidores raíz, cuyas direcciones IP están presentes en un archivo de configuración del sistema y se cargan en el caché del DNS al iniciar el servidor DNS. Tienen temporizaciones muy grandes por lo que, una vez cargadas, nunca se purgan del caché.

7.2.3. Servidores de nombres

Cuando menos en teoría, un solo servidor de nombres podría contener la base de datos DNS completa y responder a todas las consultas sobre ella. En la práctica, este servidor estaría tan sobrecargado que sería inservible. Es más, si llegara a caerse, la Internet completa se vendría abajo.

Para evitar los problemas asociados a tener una sola fuente de información, el espacio de nombres DNS se divide en zonas no traslapantes. Una manera posible de dividir el espacio de nombres de la figura 7-25 se muestra en la figura 7-28. Cada zona contiene una parte del árbol y también contiene servidores de nombres que tienen la información de autorización correspondiente a esa zona. Normalmente, una zona tendrá un servidor de nombres primario, que obtiene su información de un archivo en su disco, y uno o más servidores secundarios, que obtienen su información del servidor de nombres primario. Para mejorar la confiabilidad, algunos servidores de cierta zona pueden situarse fuera de la zona.

El lugar donde se colocan los límites de las zonas dentro de una zona es responsabilidad del administrador de esa zona. Esta decisión se toma en gran medida basada en la cantidad de

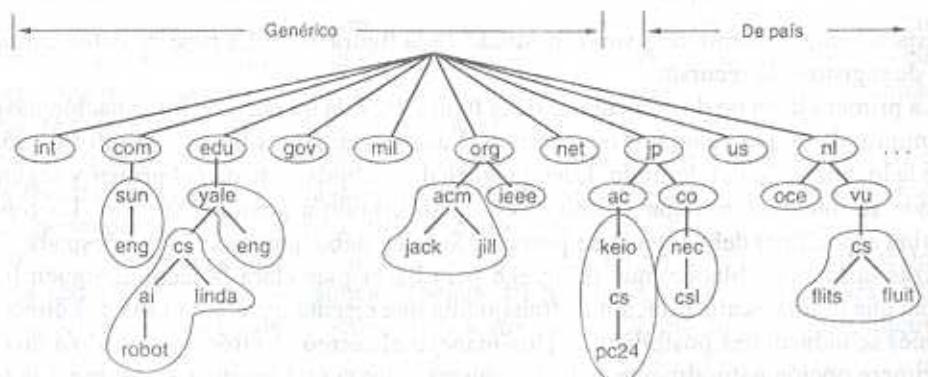


Figura 7-28. Parte del espacio de nombres DNS, donde se muestra la división en zonas.

servidores de nombres deseados y su ubicación. Por ejemplo, en la figura 7-28, Yale tiene un servidor para *yale.edu* que maneja *eng.yale.edu*, pero no *cs.yale.edu*, que es una zona aparte con sus propios servidores de nombres. Tal decisión podría tomarse cuando un departamento, como el de inglés, no desea operar su propio servidor de nombres, pero un departamento como el de informática sí. En consecuencia, *cs.yale.edu* es una zona aparte, pero *eng.yale.edu* no.

Cuando un resolvelor tiene una consulta referente a un nombre de dominio, la pasa a uno de los servidores de nombres locales. Si el dominio que se busca cae bajo la jurisdicción del servidor de nombres, como *ai.cs.yale.edu*, que cae bajo *cs.yale.edu*, devuelve los registros de recursos autorizados. Un registro autorizado es uno que viene de la autoridad que administra el registro, y por tanto siempre es correcto. Los registros autorizados contrastan con los registros en caché, que podrían no estar actualizados.

Por otro lado, si el dominio es remoto y no hay información disponible localmente sobre el dominio solicitado, el servidor de nombres envía un mensaje de consulta para el dominio solicitado al servidor de nombres de nivel superior. Para hacer más claro este proceso, considere el ejemplo de la figura 7-29. Aquí, un resolvelor de *flits.cs.vu.nl* quiere saber la dirección IP del host *linda.cs.yale.edu*. En el paso 1, envía una consulta al servidor de nombres local, *cs.vu.nl*. Esta consulta contiene el nombre de dominio buscado, el tipo (A) y la clase (IN).

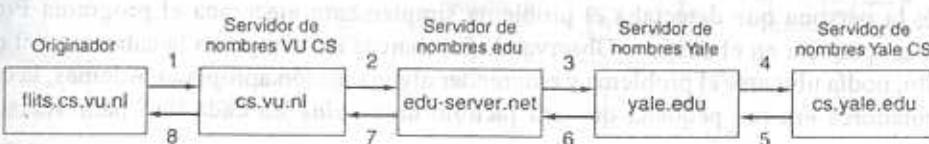


Figura 7-29. Manera en que un resolvelor busca un nombre remoto en ocho pasos.

Supongamos que el servidor de nombres local nunca ha tenido una consulta para este dominio antes, y no sabe nada sobre él; puede preguntar a algunos otros servidores de nombres cercanos, pero si ninguno de ellos sabe nada, enviará un paquete UDP al servidor de *edu* indicado en su base de datos (véase la figura 7-29), *edu-server.net*. Es improbable que este servidor sepa la dirección de *linda.cs.yale.edu*, y probablemente tampoco sabe la de *cs.yale.edu*, pero debe conocer a todos sus hijos, por lo que reenvía la solicitud al servidor de nombres de *yale.edu* (paso 3). A su vez, éste reenvía la solicitud a *cs.yale.edu* (paso 4), que debe tener los registros de recursos autorizados. Puesto que cada solicitud es de un cliente a un servidor, el registro de recursos solicitado regresa a través de los pasos 5 a 8.

Una vez que estos registros regresan al servidor de nombres *cs.vu.nl*, son ingresados en caché ahí, para el caso de que se necesiten después. Sin embargo, esta información no es autorizada, puesto que los cambios hechos en *cs.yale.edu* no se propagarán a todos los cachés del mundo que puedan saber sobre ella. Por esta razón, las entradas de caché no deben vivir demasiado tiempo. Ésta es la razón de que el campo *tiempo_de_vida* se incluya en cada registro de recurso; indica a los servidores de nombres remotos el tiempo durante el cual deben mantener en caché los registros. Si cierta máquina ha tenido la misma dirección IP durante años, puede ser

seguro poner en caché esa información durante un día. En el caso de información más volátil, podría ser más seguro purgar los registros tras unos cuantos segundos o un minuto.

Vale la pena mencionar que el método de consultas aquí descrito se conoce como **consulta recurrente**, puesto que cada servidor que no tiene toda la información solicitada sale a encontrarla en algún lado y luego la informa. Es posible un procedimiento alternativo. En él, cuando una consulta no puede satisfacerse localmente, falla la consulta, pero se devuelve el nombre del siguiente servidor a intentar a lo largo de la línea. Este procedimiento confiere al cliente mayor control sobre el proceso de búsqueda. Algunos servidores no implementan consultas recurrentes y siempre devuelven el nombre del siguiente servidor a intentar.

También vale la pena indicar que, cuando un cliente DNS no recibe una respuesta antes de terminar su temporizador, normalmente intentará otro servidor la siguiente vez. La suposición aquí es que el servidor probablemente está inactivo, y no que la solicitud o la respuesta se perdieron.

7.3. SNMP — PROTOCOLO SENCILLO DE ADMINISTRACIÓN DE REDES

En los primeros días de ARPANET, si el retardo a algún *host* se volvía inexplicablemente grande, la persona que detectaba el problema simplemente ejecutaba el programa Ping para rebotar un paquete en el destino. Observando las marcas de tiempo en la cabecera del paquete devuelto, podía ubicarse el problema y emprender alguna acción apropiada. Además, la cantidad de enrutadores era tan pequeña que era factible hacer ping en cada uno para ver si estaba enfermo.

Cuando ARPANET se convirtió en la Internet mundial, con múltiples columnas vertebrales (*backbones*) y operadores, esta solución dejó de ser adecuada, por lo que se requirieron mejores herramientas de administración de la red. En los RFC 1028 y 1067 se definieron dos intentos tempranos, que vivieron poco tiempo. En mayo de 1990, se publicó el RFC 1157, definiendo la versión 1 del SNMP (*Simple Network Management Protocol*, protocolo sencillo de administración de redes). Junto con un documento acompañante (el RFC 1155) sobre información de administración, el SNMP proporcionó una manera sistemática de supervisar y administrar una red de cómputo. Esta estructura y su protocolo se implementaron ampliamente en los productos comerciales y se volvieron los estándares de facto para la administración de redes.

A medida que se adquirió experiencia, se hicieron evidentes las limitaciones del SNMP, por lo que se definió (en los RFC 1441 a 1452) una versión mejorada del SNMP (SNMPv2), que se volvió un estándar Internet. En las secciones siguientes haremos un análisis breve del modelo y el protocolo SNMP (queriendo decir SNMPv2).

Aunque el SNMP se diseñó con la idea de que fuera sencillo, cuando menos un autor ha logrado producir un libro de 600 páginas sobre él (Stallings, 1993a). Para descripciones más compactas (450-550 páginas), véanse los libros de Rose (1994) y Rose y McCloghrie (1995), quienes fueron parte del grupo de diseñadores del SNMP. Otras referencias son (Feit, 1995, y Hein y Griffiths, 1995).

7.3.1. El modelo SNMP

El modelo SNMP de una red administrada consta de cuatro componentes:

1. Nodos administrados.
2. Estaciones administradas.
3. Información de administración.
4. Un protocolo de administración.

Estas partes se ilustran en la figura 7-30 y se analizan a continuación.

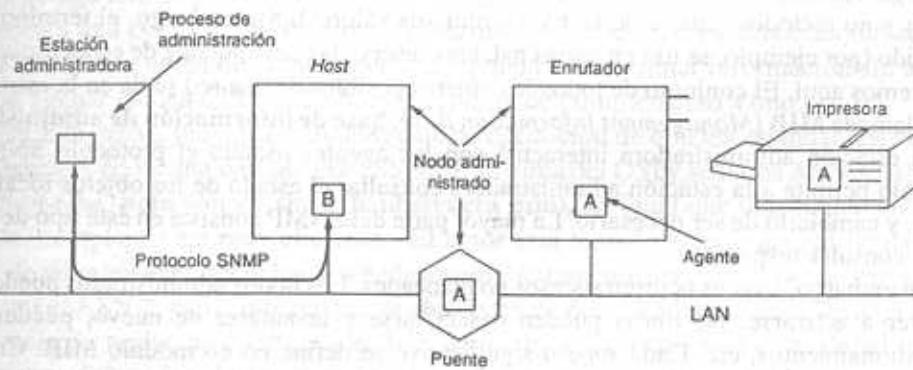


Figura 7-30. Componentes del modelo de administración SNMP.

Los nodos administrados pueden ser *hosts*, enrutadores, puentes, impresoras u otros dispositivos capaces de comunicar información de estado al mundo exterior. Para ser administrado directamente por el SNMP, un nodo debe ser capaz de ejecutar un proceso de administración SNMP, llamado agente SNMP. Todas las computadoras cumplen este requisito, al igual que una cantidad creciente de puentes, enrutadores y dispositivos periféricos diseñados para uso en redes. Cada agente mantiene una base de datos local de variables que describen su estado e historia y que afectan su operación.

La administración de la red se hace desde estaciones administradoras, que son, de hecho, computadoras de propósito general que ejecutan un *software* de administración especial. La estación administradora contiene uno o más procesos que se comunican con los agentes a través de la red, emitiendo comandos y recibiendo respuestas. En este diseño, toda la inteligencia está en las estaciones administradoras, a fin de mantener a los agentes tan sencillos como sea posible y minimizar su impacto sobre los dispositivos en los que se ejecutan. Muchas estaciones administradoras tienen una interfaz gráfica de usuario para que el administrador de la red pueda inspeccionar el estado de la red y emprenda acciones cuando se requieran.

Muchas redes reales son de varios proveedores, con *hosts* de uno o más fabricantes, puentes y enruteadores de otras compañías e impresoras de otros fabricantes. A fin de permitir que una estación administradora (potencialmente de otro proveedor más) hable con estos componentes diversos, la naturaleza de la información mantenida por todos los dispositivos debe especificarse rígidamente. Hacer que la estación administradora pregunte a un enruteador su tasa de pérdida de paquetes no es de utilidad si el enruteador no lleva el registro de su tasa de pérdidas. Por tanto, el SNMP describe (con riguroso detalle) la información exacta de cada tipo de agente que tiene que administrar y el formato con que éste tiene que proporcionarle los datos. La parte más grande del modelo SNMP es la definición de quién tiene que llevar el registro de qué y cómo se comunica esta información.

Muy brevemente, cada dispositivo mantiene una o más variables que describen su estado. En la documentación del SNMP, estas variables se llaman **objetos**, pero el término es engañoso porque no son objetos en el sentido de un sistema orientado a objetos, puesto que sólo tienen estados y no métodos (aparte de leer y escribir sus valores). Sin embargo, el término está tan difundido (por ejemplo, se usa en varias palabras reservadas del lenguaje de especificación) que lo usaremos aquí. El conjunto de todos los objetos posibles de una red se da en la estructura de datos llamada **MIB** (*Management Information Base*, base de información de administración).

La estación administradora interactúa con los agentes usando el protocolo SNMP. Este protocolo permite a la estación administradora consultar el estado de los objetos locales de un agente, y cambiarlo de ser necesario. La mayor parte del SNMP consiste en este tipo de comunicación consulta-respuesta.

Sin embargo, a veces ocurren sucesos no planeados. Los nodos administrados pueden caerse y volver a activarse, las líneas pueden desactivarse y levantarse de nuevo, pueden ocurrir congestionamientos, etc. Cada suceso significativo se define en un módulo MIB. Cuando un agente nota que ha ocurrido un suceso significativo, de inmediato lo informa a todas las estaciones administradoras de su lista de configuración. Este informe se llama **interrupción SNMP** (por razones históricas). El informe por lo general sencillamente indica que ha ocurrido un suceso; es responsabilidad de la estación administradora emitir consultas para averiguar los detalles. Dado que la comunicación entre los nodos administrados y la estación administradora no es confiable (es decir, no se valida), es deseable que la estación administradora de todas maneras sondee ocasionalmente cada nodo administrado, buscando sucesos inusuales, sólo por no dejar. El modelo de sondeo a intervalos grandes con aceleración al recibirse una interrupción se llama **sondeo dirigido a interrupción**.

Este modelo supone que cada nodo administrado es capaz de ejecutar un agente SNMP internamente. Los dispositivos más viejos y los dispositivos no contemplados para usarse en redes podrían no tener esta capacidad. Para manejarlos, el SNMP define un **agente apoderado**, es decir un agente que supervisa uno o más dispositivos no SNMP y se comunica con la estación administradora a nombre de ellos, y que posiblemente se comunique con los dispositivos mismos usando algún protocolo no estándar.

Por último, la seguridad y la validación de identificación desempeñan un papel preponderante en el SNMP. Una estación administradora también tiene la capacidad de aprender mucho sobre cada nodo que está bajo su control, y también puede apagarlos todos. Por tanto, es de gran

importancia que los agentes estén convencidos de que las consultas supuestamente originadas por la estación administradora realmente vienen de la estación administradora. En el SNMPv1, la estación administradora comprobaba su identidad poniendo una contraseña (de texto normal) en cada mensaje. En el SNMPv2, la seguridad se mejoró considerablemente usando técnicas criptográficas modernas del tipo que ya hemos estudiado. Sin embargo, esta adición hizo más voluminoso a un protocolo de por sí grande, y luego se desechó.

7.3.2. ASN.1 — Notación de sintaxis abstracta 1

El corazón del modelo SNMP es el grupo de objetos administrados por los agentes y leídos y escritos por la estación administradora. Para hacer posible la comunicación multiproveedor, es esencial que estos objetos se definan de una manera estándar y neutral desde el punto de vista de los proveedores. Es más, se requiere una forma estándar de codificarlos para su transferencia a través de una red. Si bien las definiciones en C satisfarían el primer requisito, tales definiciones no definen una codificación de bits en el alambre de modo tal que una estación administradora de 32 bits *little endian* de complemento a 2 pueda intercambiar información sin ambigüedades con un agente en una CPU de 16 bits *big endian* de complemento a uno.

Por esta razón, se requiere un lenguaje de definición de objetos estándar, así como reglas de codificación. El lenguaje usado por el SNMP se toma del OSI y se llama **ASN.1** (*Abstract Syntax Notation One, notación de sintaxis abstracta uno*). Al igual que una buena parte del OSI, es grande, complejo y no muy eficiente. (El autor está tentado a decir que, al llamarlo ASN.1 en lugar de simplemente ASN, los diseñadores implícitamente aceptaron que pronto sería reemplazado por el ASN.2, pero cortésmente se abstendrá de decirlo.) La supuesta ventaja del ASN.1 (la existencia de reglas de codificación de bits no ambiguas) ahora es en realidad una debilidad, puesto que las reglas de codificación se optimizan para minimizar la cantidad de bits en el alambre, al costo de desperdiciar tiempo de CPU en ambos extremos para su codificación y decodificación. Un esquema más sencillo, que usara enteros de 32 bits alineados según límites de 4 bytes probablemente habría sido mejor. Sin embargo, para bien o para mal, el SNMP está impregnado con ASN.1 (aunque se trata de un subgrupo simplificado), por lo que cualquiera que desee entender en verdad el SNMP debe conocer bien el ASN.1. De ahí, la siguiente explicación.

Comencemos por el lenguaje de descripción de datos, descrito en el Estándar internacional 8824. Después, estudiaremos las reglas de codificación, descritas en el Estándar internacional 8825. La sintaxis abstracta ASN.1 esencialmente es un lenguaje de declaración de datos primitivos. Permite al usuario definir objetos primitivos y luego combinarlos para formar otros más complejos. Una serie de declaraciones en ASN.1 es funcionalmente similar a las declaraciones encontradas en los archivos de cabecera asociados a muchos programas en C.

El SNMP tiene algunas convenciones lexicográficas que respetaremos. Sin embargo, éstas no son exactamente las mismas que usa ASN.1. Los tipos de datos interconstruidos se escriben en mayúsculas (por ejemplo, *INTEGER*). Los tipos definidos por el usuario comienzan con una letra mayúscula pero deben contener cuando menos un carácter distinto de una letra mayúscula. Los identificadores pueden contener letras mayúsculas y minúsculas, dígitos y guiones, pero deben comenzar con una letra minúscula (por ejemplo, *counter*). Los espacios blancos (tabuladores,

retornos de carro, etc.) no son relevantes. Por último, los comentarios comienzan con — y continúan hasta la siguiente aparición de —.

Los tipos de datos básicos del ASN.1 permitidos en el SNMP se muestran en la figura 7-31. (Generalmente ignoraremos las características del ASN.1, como los tipos *BOOLEAN* y *REAL*, que no están permitidos en el SNMP.) El uso de los códigos se describirá después.

Tipo primitivo	Significado	Código
INTEGER	Entero de longitud arbitraria	2
BIT STRING	Cadena de 0 o más bits	3
OCTET STRING	Cadena de 0 o más bytes sin signo	4
NULL	Marcador de lugar	5
OBJECT IDENTIFIER	Tipo de datos definido oficialmente	6

Figura 7-31. Tipos de datos primitivos ASN.1 permitidos en el SNMP.

Una variable de tipo *INTEGER* (entero) puede, en teoría, adoptar cualquier valor entero, pero otras reglas de SNMP limitan su alcance. Como ejemplo del uso de los tipos, considere la manera en que se declara y (opcionalmente) se establece inicialmente en 100 una variable, *count*, de tipo *INTEGER* en ASN.1:

```
count INTEGER ::= 100
```

Con frecuencia se requiere un subtipo cuyas variables se restringen a valores específicos o a un intervalo específico. Éstos se pueden declarar como sigue:

```
Status ::= INTEGER { up(1), down(2), unknown(3) }
```

```
PacketSize ::= INTEGER (0..1023)
```

Las variables del tipo *BIT STRING* (cadena de bits) y *OCTET STRING* (cadena de octetos) contienen cero o más bits y bytes, respectivamente. Un bit tiene 0 o 1. Un byte cae en el intervalo de 0 a 255, inclusive. En ambos tipos puede darse una longitud de cadena y un valor inicial.

Los *OBJECT IDENTIFIERS* (identificadores de objetos) ofrecen una manera de identificar objetos. En principio, cada objeto definido en cada estándar oficial puede identificarse de manera única. El mecanismo consiste en definir un árbol de estándares, y colocar cada objeto en cada estándar en una localidad única del árbol. La parte del árbol que incluye la MIB del SNMP se muestra en la figura 7-32.

El nivel superior del árbol lista todas las organizaciones de estándares importantes del mundo (desde el punto de vista de la ISO), es decir, el ISO y el CCITT (ahora ITU), más la combinación de los dos. Del nodo *iso* se definen cuatro arcos, uno de los cuales es para *identified-organization* (organización identificada), lo cual es una admisión por parte de la ISO de que tal vez algunas otras personas también intervienen vagamente en los estándares. El Departamento

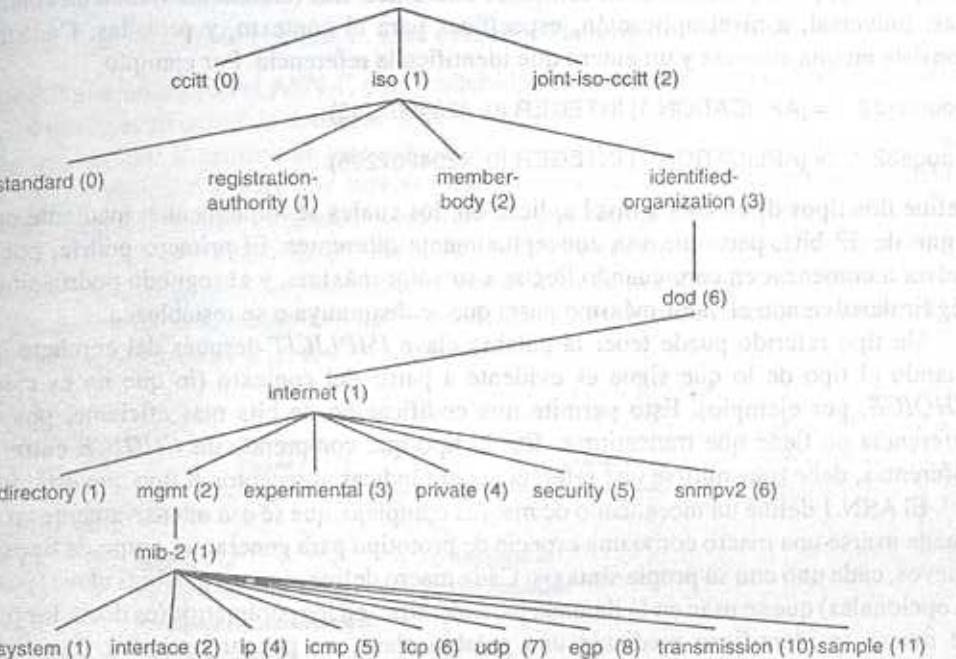


Figura 7-32. Parte del árbol de nombres de objetos del ASN.1.

de la Defensa de Estados Unidos tiene un lugar asignado en este subárbol, y ya asignó el número de Internet 1 en su jerarquía. En la jerarquía de Internet, la MIB del SNMP tiene el código 1.

Cada arco de la figura 7-32 tiene tanto una etiqueta como un número, de modo que los nodos pueden identificarse con una lista de arcos, usando etiqueta(número) o números. Por tanto, todos los objetos de la MIB de SNMP se identifican mediante una etiqueta de la forma

{iso identified-organization(3) dod(6) internet(1) mgmt(2) mib-2(1) ...}

o, de manera alternativa {1 3 6 1 2 1 ...}. También se permiten formas mixtas. Por ejemplo, la identificación anterior también puede escribirse como

{internet(1) 2 1 ...}

De esta manera, cada objeto de cada estándar puede representarse como un *OBJECT IDENTIFIER*.

El ASN.1 define cinco maneras de construir tipos nuevos a partir de los básicos. *SEQUENCE* es una lista ordenada de tipos, parecida a una estructura en C y a un registro en Pascal. *SEQUENCE OF* es un arreglo de una dimensión de un solo tipo. *SET* y *SET OF* son análogos, pero desordenados. *CHOICE* crea una unión a partir de una lista dada de tipos. Los dos constructores de conjuntos no se usan en ninguno de los documentos SNMP.

Otra manera de crear tipos nuevos es haciendo referencia a los viejos. La referencia de un tipo es algo parecido a la práctica de C de definir tipos nuevos, digamos *time_t* y *size_t*, ambos

de tipo *long*, pero que se usan en contextos diferentes. Las referencias vienen en cuatro categorías: universal, a nivel aplicación, específicas para el contexto, y privadas. Cada referencia consiste en una etiqueta y un entero que identifica la referencia. Por ejemplo,

`Counter32 ::= [APPLICATION 1] INTEGER (0..4294967295)`

`Gauge32 ::= [APPLICATION 2] INTEGER (0..4294967295)`

define dos tipos diferentes a nivel aplicación, los cuales se implementan mediante enteros sin signo de 32 bits, pero que son conceptualmente diferentes. El primero podría, por ejemplo, volver a comenzar en cero cuando llegue a su valor máximo, y el segundo podría simplemente seguir devolviendo el valor máximo hasta que se disminuya o se restablezca.

Un tipo referido puede tener la palabra clave *IMPLICIT* después del corchete que cierra cuando el tipo de lo que sigue es evidente a partir del contexto (lo que no es cierto en un *CHOICE*, por ejemplo). Esto permite una codificación de bits más eficiente, puesto que la referencia no tiene que transmitirse. En un tipo que comprende un *CHOICE* entre dos tipos diferentes, debe transmitirse una referencia para indicar al receptor el tipo que está presente.

El ASN.1 define un mecanismo de macros complejo, que se usa intensivamente en el SNMP. Puede usarse una macro como una especie de prototipo para generar un grupo de tipos y valores nuevos, cada uno con su propia sintaxis. Cada macro define algunas palabras clave (posiblemente opcionales) que se usan en la llamada para identificar a los parámetros (es decir, los parámetros de macro se identifican mediante una palabra clave, no por su posición). Los detalles del funcionamiento de las macros ASN.1 están más allá del alcance de este libro. Baste con decir que una macro se invoca dando su nombre seguido de (parte de) sus palabras clave y sus valores para esta llamada. Las macros se expanden en el momento de la compilación, no durante la ejecución. A continuación se citarán algunos ejemplos de macros.

Sintaxis de transferencia ASN.1

Una sintaxis de transferencia ASN.1 define la manera en que los valores de los tipos ASN.1 se convierten sin ambigüedad en una secuencia de bytes para su transmisión (y se decodifican sin ambigüedad en la otra terminal). La sintaxis de transferencia usada por el ASN.1 se llama *BER* (*Basic Encoding Rules*, reglas básicas de codificación). El ASN.1 tiene otras sintaxis de transferencia que el SNMP no usa. Las reglas son recurrentes, por lo que la codificación de un objeto estructurado es simplemente la concatenación de las codificaciones de los objetos componentes. De esta manera, todas las codificaciones de objetos pueden reducirse a una secuencia bien definida de objetos primitivos codificados. La codificación de estos objetos, a su vez, la definen las BER.

El principio que guía las reglas básicas de codificación es que cada valor transmitido, tanto primitivo como construido, consiste en cuatro campos:

1. Identificador (tipo o etiqueta).
2. La longitud del campo de datos, en bytes.

3. El campo de datos.

4. La bandera de fin de contenido, si se desconoce la longitud de los datos.

El último está permitido por el ASN.1, pero prohibido específicamente en el SNMP, por lo que supondremos que la longitud de datos siempre se conoce.

El primer campo identifica el elemento que le sigue, y tiene tres subcampos, como se muestra en la figura 7-33. Los dos bits de orden mayor identifican el tipo de etiqueta. El siguiente bit indica si el valor es primitivo (0) o no (1). Las banderas de etiqueta son 00, 01, 10 y 11, para *UNIVERSAL*, *APPLICATION*, específica para el contexto y *PRIVATE*, respectivamente. Los 5 bits restantes pueden usarse para codificar el valor de la etiqueta si ésta se encuentra en el intervalo de 0 a 30. Si la etiqueta es de 31 o más, los 5 bits de orden menor contienen 11111, con el valor verdadero en el siguiente byte o bytes.

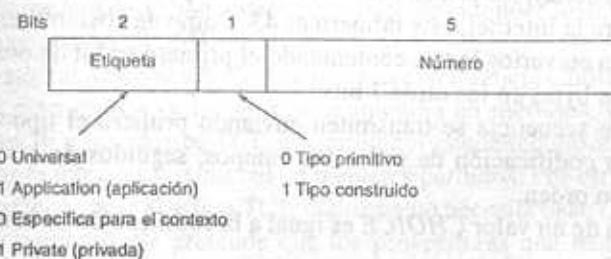


Figura 7-33. El primer byte de cada elemento de datos enviado en la sintaxis de transferencia ASN.1.

La regla empleada para codificar valores de las etiquetas mayores que 30 se ha diseñado para manejar números arbitrariamente grandes. Cada byte identificador después del primero contiene 7 bits de datos. El bit de orden mayor se establece en 0 en todos los bytes menos el último byte. Por tanto, pueden manejarse valores de etiqueta de hasta $2^7 - 1$ en 2 bytes y de hasta $2^{14} - 1$ en 3 bytes.

La codificación de los tipos *UNIVERSAL* es directa. Cada tipo primitivo tiene asignado un código, que se da en la tercera columna de la figura 7-31. *SEQUENCE* y *SEQUENCE OF* comparten el código 16. *CHOICE* no tiene un código, puesto que cualquier valor enviado siempre tiene un tipo específico. Los otros códigos son para tipos no usados en el SNMP.

A continuación del campo de identificador de 1 byte viene un campo que indica la cantidad de bytes que ocupan los datos. Las longitudes menores que 128 bytes se codifican directamente en 1 byte cuyo bit de la izquierda es 0; las mayores usan varios bytes, conteniendo el primer byte un 1 en el bit de orden mayor y el campo de longitud (de hasta 127 bytes) en los 7 bits de orden menor. Por ejemplo, si la longitud de los datos es de 1000 bytes, el primer byte contiene 130 para indicar que continúa un campo de longitud de 2 bytes. Entonces vienen 2 bytes cuyo valor es de 1000, con el byte de orden mayor primero.

La codificación del campo de datos depende del tipo de datos presente. Los enteros se codifican en complemento a dos. Un entero positivo menor que 128 requiere 1 byte, un entero positivo menor que 32,768 requiere 2 bytes, etc. El byte más significativo se transmite primero.

Las cadenas de bits se codifican como ellas mismas. El único problema es la manera de indicar la longitud. El campo de longitud indica la cantidad de *bytes* que tiene el valor, no la cantidad de *bits*. La solución escogida es transmitir un byte antes de la cadena de bits real que indique la cantidad de bits (0 a 7) del byte final que no se usa. Por tanto, la codificación de la cadena de 9 bits '01001111' sería 07, 4F, 80 (hexadecimal).

Las cadenas de octetos son fáciles. Los bytes de la cadena simplemente se transmiten en estilo *big endian* estándar, de izquierda a derecha.

El valor nulo se indica estableciendo la longitud del campo en 0. De hecho, no se transmite ningún valor numérico.

Un *OBJECT IDENTIFIER* se codifica como la secuencia de enteros que representa. Por ejemplo, la Internet es {1, 3, 6, 1}. Sin embargo, dado que el primer número siempre es 0, 1 o 2, y el segundo es menor que 40 (por definición: la ISO simplemente no reconocerá la 41^a categoría que aparezca en su puerta), los primeros dos números, *a* y *b*, se codifican como 1 byte que tiene el valor $40a + b$. Para la Internet, este número es 43. Como de costumbre, los números mayores que 127 se codifican en varios bytes, conteniendo el primero su bit de orden mayor establecido en 1 y una cuenta de bytes en los otros 7 bits.

Ambos tipos de secuencia se transmiten enviando primero el tipo o referencia, luego la longitud total de la codificación de todos los campos, seguidos de los campos mismos. Los campos se envían en orden.

La codificación de un valor *CHOICE* es igual a la codificación de la estructura de datos que se está transfiriendo.

En la figura 7-34 se muestra un ejemplo de la codificación de algunos valores. Los valores codificados son el *INTEGER* 49, la *OCTET STRING* '110', la cadena de octetos "xy", el único valor posible de *NULL*, el *OBJECTIDENTIFIER* de Internet {1, 3, 6, 1}, y un valor *Gauge32* de 14.

	Tipo de etiqueta	Número de etiqueta	Longitud	Valor
Entero 49		00000010	00000001	00110001
Cadena de bits '110'		00000011	00000010	00000101 11000000
Cadena de octetos "xy"		00000100	00000010	01111000 01111001
NULO		00000101	00000000	
Objeto Internet		00000110	00000011	00101011 00000110 00000001
Gauge 32 14		01000010	00000001	00001110

Figura 7-34. Codificación ASN.1 de algunos valores de ejemplo.

7.3.3. SMI — Estructura de la información de administración

En la sección precedente, estudiamos sólo aquellas partes del ASN.1 que se usan en el SNMP. En realidad, los documentos SNMP se organizan de manera diferente. El RFC 1442 primero dice que el ASN.1 se usará para describir las estructuras de datos SNMP, pero luego dedica 57 páginas a desechar partes del estándar ASN.1 que no quiere y agregar nuevas definiciones (en ASN.1) que se requieren. En particular, el RFC 1442 define cuatro macros clave y ocho nuevos tipos de datos que se usan intensivamente en todo el SNMP. Es este sub-super-conjunto del ASN.1, que recibe el poco elegante nombre de *SMI* (*Structure of Management Information*, estructura de información de administración), lo que se usa en realidad para definir las estructuras de datos del SNMP.

Aunque este enfoque es un tanto burocrático, son necesarias algunas reglas si los productos de cientos de proveedores han de hablar entre sí y realmente entender lo que dicen. Por tanto, es pertinente hablar un poco sobre el SMI.

En el nivel más bajo, las variables SNMP se definen como objetos individuales. Los objetos relacionados se reúnen en grupos, y los grupos se integran en módulos. Por ejemplo, existen grupos para los objetos IP y los objetos TCP. Un enrutador puede reconocer los grupos IP, dado que a su administrador le interesa la cantidad de paquetes perdidos. Por otra parte, un enrutador de bajo nivel podría no reconocer el grupo TCP, dado que no necesita usar TCP para desempeñar sus funciones de enrutamiento. Se pretende que los proveedores que reconocen un grupo reconozcan todos los objetos de ese grupo. Sin embargo, un proveedor que reconoce un módulo no tiene que reconocer todos sus grupos, pues podría ser que no todos se aplicaran al dispositivo.

Todos los módulos MIB comienzan con una invocación de la macro *MODULE-IDENTITY*. Sus parámetros proporcionan el nombre y la dirección del implementador, la historia de modificaciones y otra información administrativa. Típicamente, a esta llamada le sigue una invocación de la macro *OBJECT-IDENTITY*, que indica si el módulo cabe en el árbol de nombres de la figura 7-32.

Después vienen una o más invocaciones de la macro *OBJECT-TYPE*, que nombra las variables manejadas y especifica sus propiedades. El agrupamiento de variables en grupos se hace por convención; no hay enunciados *BEGIN-GROUP* ni *END-GROUP* en el ASN.1 o el SMI.

La macro *OBJECT-TYPE* tiene cuatro parámetros requeridos y cuatro (a veces) opcionales. El primer parámetro requerido es *SYNTAX* y define el tipo de datos de la variable como uno de los tipos listados en la figura 7-35. En su mayor parte, estos tipos deberían ser autoexplicativos, con los siguientes comentarios. El sufijo 32 se usa cuando el implementador en realidad quiere un número de 32 bits, incluso si las CPU de todas las máquinas a la vista son de 64 bits. Los *gauges* (medidores) difieren de los contadores en que no comienzan otra vez en cero cuando llegan a sus límites; se quedan ahí. Si un enrutador ha perdido exactamente 2^{32} paquetes, es mejor informar esto como $2^{32} - 1$ que como 0. El SMI también maneja arreglos, pero no los trataremos aquí. Para los detalles, véase (Rose, 1994).

Nombre	Tipo	Bytes	Significado
INTEGER	Numérica	4	Entero (32 bits en las implementaciones actuales)
Counter32	Numérica	4	Contador sin signo de 32 bits que da la vuelta
Gauge32	Numérica	4	Valor sin signo que no da la vuelta
Integer32	Numérica	4	Entero de 32 bits, incluso en una CPU de 64 bits
UInteger32	Numérica	4	Como Integer32, pero sin signo
Counter64	Numérica	8	Contador de 64 bits
TimeTicks	Numérica	4	Pulsos de tiempo en centésimas de segundo desde algún momento
BIT STRING	Cadena	4	Mapa de bits de 1 a 32 bits
OCTET STRING	Cadena	≥ 0	Cadena de bytes de longitud variable
Opaque	Cadena	≥ 0	Obsoleto; sólo para compatibilidad hacia atrás
OBJECT IDENTIFIER	Cadena	> 0	Lista de enteros de la figura 7-32
IpAddress	Cadena	4	Dirección Internet decimal con puntos
NsapAddress	Cadena	< 22	Dirección NSAP de OSI

Figura 7-35. Tipos de datos usados para las variables SNMP supervisadas.

Además de requerir una especificación del tipo de datos usados por la variable que se está declarando, la macro *OBJECT TYPE* requiere tres parámetros más. *MAX-ACCESS* contiene información sobre el acceso a la variable. Los valores más comunes son lectura-escritura y sólo lectura. Si una variable es lectura-escritura, la estación administradora puede establecerla. Si es sólo lectura, la estación administradora puede leerla pero no establecerla.

El *STATUS* tiene tres valores posibles. Una variable actual se ajusta a la especificación SNMP actual. Una variable obsoleta no se ajusta, pero se ajustaba a alguna versión más vieja. Una variable desaprobada está en medio; en realidad es obsoleta, pero el comité que escribió el estándar no se atrevió a decir esto en público por miedo a la reacción de los proveedores cuyos productos la usan. Sin embargo, sus dfas están contados.

El último parámetro requerido es *DESCRIPTION*: una cadena ASCII que indica lo que hace la variable. Si un administrador compra un bonito dispositivo nuevo, lo consulta desde la estación de administración y descubre que lleva el registro de *pktCnt*, la obtención del campo *DESCRIPTION* supuestamente dará una pista del tipo de paquetes que está contando. Este campo está dirigido exclusivamente al usuario humano (no le sirve a la computadora).

Un ejemplo sencillo de una declaración *OBJECT TYPE* se da en la figura 7-36. La variable se llama *lostPackets* y puede ser útil en un enrutador o en otro dispositivo que maneja paquetes. El valor después del signo ::= lo coloca en el árbol.

```

lostPackets OBJECT TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "La cantidad de paquetes perdidos desde la última activación"
 ::= {experimental 20}

```

Figura 7-36. Variable SNMP de ejemplo.

7.3.4. La MIB — Base de información de administración

El conjunto de objetos administrados por el SNMP se define en la MIB. Por conveniencia, estos objetos se agrupan (actualmente) en 10 categorías, que corresponden a los 10 nodos bajo *mib-2* en la figura 7-32. (Nótese que *mib-2* corresponde al SNMPv2 y que el objeto 9 ya no está presente.) La intención de las 10 categorías es proporcionar una base de lo que debe entender una estación administradora. Ciertamente se agregarán nuevas categorías y objetos en el futuro, y los proveedores están en libertad de definir objetos adicionales para sus productos. Las 10 categorías se definen en la figura 7-37.

Grupo	# objetos	Descripción
System	7	Nombre, ubicación y descripción del equipo
Interfaces	23	Interfaces de red y su tráfico medido
AT	3	Traducción de direcciones (desaprobada)
IP	42	Estadísticas de paquetes IP
ICMP	26	Estadísticas sobre los paquetes ICMP recibidos
TCP	19	Algoritmos, parámetros y estadísticas TCP
UDP	6	Estadísticas de tráfico UDP
EGP	20	Estadísticas de tráfico de protocolo de pasarela exterior
Transmission	0	Reservado para MIB de medio específico
SNMP	29	Estadísticas de tráfico SNMP

Figura 7-37. Grupos de objetos del MIB-II de Internet.

Aunque las limitaciones de espacio nos impiden entrar en los detalles de los 175 objetos definidos en la MIB-II, algunos comentarios pueden ser útiles. El grupo *system* permite al administrador encontrar el nombre del dispositivo, su constructor, el *hardware* y el *software* que contiene, su ubicación y lo que se supone que hace. También se proporcionan la hora del último

arranque y el nombre y dirección de la persona de contacto. Esta información implica que una compañía puede contratar administración de sistemas de otra compañía en una ciudad lejana y lograr que esta compañía sea capaz de determinar con facilidad la configuración actualmente administrada y la persona a contactar si hay problemas en diversos dispositivos.

El grupo *interfaces* tiene que ver con los adaptadores de red; lleva la pista de la cantidad de paquetes y bytes enviados y recibidos de la red, la cantidad de descartados, la cantidad de difusiones y el tamaño de la cola de salida.

El grupo AT estaba presente en la MIB-I y proporcionaba información sobre la correspondencia de direcciones (por ejemplo, direcciones Ethernet a IP). Esta información se movió a unas MIB específicas para protocolo en el SNMPv2.

El grupo IP se encarga del tráfico que entra y sale del nodo. Es especialmente pródigo en contadores que llevan el registro de la cantidad de paquetes descartados por diversas razones (por ejemplo, la inexistencia de una ruta conocida al destino o la falta de recursos). También están disponibles estadísticas sobre la fragmentación de datagramas y su reensamblaje. Todos estos elementos son particularmente importantes para la administración de las redes.

El grupo ICMP trata los mensajes de error de IP. Básicamente, este grupo tiene un contador por cada mensaje ICMP para registrar la cantidad que ha sido enviada de ese tipo.

El grupo TCP supervisa la cantidad actual y acumulada de conexiones abiertas, segmentos enviados y recibidos, así como varias estadísticas de error.

El grupo UDP lleva una bitácora de la cantidad de datagramas UDP enviados y recibidos, y la cantidad de estos últimos que no se pudieron entregar debido a un puerto desconocido o alguna otra razón.

El grupo EGP se usa para enrutadores que manejan el protocolo de pasarela exterior. El EGP lleva el registro de la cantidad de paquetes de cada tipo que salieron, entraron y se reenviaron correctamente, y que entraron y se descartaron.

El grupo de transmisión aparta un lugar para las MIB específicas para un medio. Por ejemplo, las estadísticas específicas de Internet pueden almacenarse aquí. El propósito de incluir un grupo vacío en la MIB-II es reservar el identificador {internet 2 1 9} para tales propósitos.

El último grupo es para recolectar estadísticas sobre la operación del SNMP mismo: la cantidad de mensajes enviados, el tipo de los mensajes, y demás.

La MIB-II se definió formalmente en el RFC 1213. El grueso del RFC 1213 consiste en 175 llamadas a macros semejantes a las de la figura 7-36, con comentarios que delinean los 10 grupos. Por cada uno de los 175 objetos definidos, se da el tipo de datos junto con una descripción en inglés del uso de la variable. Para mayor información sobre la MIB-II, el lector debe referirse a este RFC.

7.3.5. El protocolo SNMP

Ya hemos visto que el modelo subyacente del SNMP es una estación administradora que envía solicitudes a los agentes de nodos administrados, haciendo consultas sobre las 175 variables antes mencionadas, y sobre muchas otras variables específicas del proveedor. Nuestro último

tema es el protocolo mismo al que le hablan la estación administradora y los agentes. Este protocolo se define en el RFC 1448.

La forma normal de uso del SNMP es que la estación administradora envíe una solicitud a un agente pidiéndole información o mandándole actualizar su estado de cierta manera. Idealmente, la respuesta del agente simplemente es la información solicitada o la confirmación de que ha actualizado su estado según se solicitó. Los datos se envían empleando la sintaxis de transferencia del ASN.1. Sin embargo, también pueden informarse varios errores, como "no existe tal variable".

El SNMP define siete mensajes que pueden enviarse. Los seis mensajes de un iniciador se listan en la figura 7-38 (el séptimo mensaje es el de respuesta). Los primeros tres mensajes solicitan la devolución de valores de variables. El primer formato nombra explícitamente las variables que quiere. El segundo solicita la siguiente variable, permitiendo al administrador ir paso por paso a través de la MIB en orden alfabético (la predeterminada es la primera variable). El tercero es para transferencias grandes, como tablas.

Mensaje	Descripción
Get-request	Solicita el valor de una o más variables
Get-next-request	Solicita la variable que sigue a ésta
Get-bulk-request	Obtiene una tabla grande
Set-request	Actualiza una o más variables
Inform-request	Mensaje de administrador a administrador describiendo la MIB local
SnmpV2-trap	Informe de interrupciones de agente a administrador

Figura 7-38. Tipos de mensajes SNMP.

Después viene un mensaje que permite al administrador actualizar las variables de un agente, hasta el límite permitido por la especificación del objeto, claro está. Sigue una solicitud informal que permite a un administrador indicarle a otro las variables que está manejando. Por último, viene el mensaje enviado de una agente a un administrador cuando ha habido una interrupción.

7.4. CORREO ELECTRÓNICO

Habiendo terminado de ver algunos de los protocolos de apoyo usados en la capa de aplicación, por fin llegamos a las aplicaciones concretas. A la pregunta: "¿qué vas a hacer ahora?", poca gente contestará: "voy a buscar algunos nombres con el DNS". La gente dice que va a leer su correo electrónico o las noticias, a navegar por el Web, o a ver una película en la red. En lo que resta de este capítulo explicaremos con cierto detalle el funcionamiento de estas cuatro aplicaciones.

El correo electrónico, o *email*, como se le conoce en inglés, ha existido por más de dos décadas. Los primeros sistemas de correo electrónico simplemente consistían en protocolos de transferencia de archivos, con la convención de que la primera línea de cada mensaje (es decir, archivo) contenía la dirección del destinatario. A medida que pasó el tiempo, las limitaciones de este enfoque se hicieron obvias. Algunas de las quejas eran:

1. El envío de un mensaje a un grupo de gente era laborioso. Los administradores con frecuencia necesitaban esta facilidad para enviar memorandos a todos sus subordinados.
2. Los mensajes no tenían estructura interna, dificultando el proceso por computadora. Por ejemplo, si un mensaje reenviado se incluía en el cuerpo de otro mensaje, la extracción de la parte reenviada del mensaje recibido era difícil.
3. El originador (remitente) nunca sabía si el mensaje había llegado o no.
4. Si alguien planeaba ausentarse por un viaje de negocios durante varias semanas y quería que todo el correo entrante fuera manejado por su secretaria, esto no era fácil de arreglar.
5. La interfaz de usuario estaba mal integrada al sistema de transmisión, requiriendo que el usuario primero editara un archivo, luego saliera del editor e invocara el programa de transferencia.
6. No era posible crear y enviar mensajes que contuvieran una mezcla de texto, dibujos, facsímiles y voz.

A medida que se acumuló experiencia, se propusieron sistemas de correo electrónico más elaborados. En 1982, se publicaron las propuestas de correo electrónico del ARPANET como RFC 821 (protocolo de transmisión) y RFC 822 (formato de mensaje). Desde entonces se han convertido en los estándares de facto de Internet. Dos años después, el CCITT bosquejó su recomendación X.400, que luego se tomó como base para el MOTIS de OSI. En 1988, el CCITT modificó el X.400 para alinearlo con MOTIS. MOTIS debía ser la aplicación insignia del OSI, un sistema que era todo para todos.

Tras una década de competencia, los sistemas de correo electrónico basados en el RFC 822 se usan ampliamente, y los basados en el X.400 han desaparecido bajo el horizonte. La manera en que un sistema armado por un puñado de graduados en informática venció a un estándar internacional oficial con fuerte respaldo de todos los PTT del mundo, muchos gobiernos y una parte importante de la industria de la computación trae a la mente la historia bíblica de David y Goliat. La razón del éxito del RFC 822 no es que sea tan bueno, sino que el X.400 está tan mal diseñado y es tan complejo que nadie pudo implementarlo bien. Dada la opción entre un sistema de correo electrónico sencillo pero funcional basado en el RFC 822 y un sistema de correo electrónico X.400 supuestamente maravilloso pero disfuncional, muchas organizaciones escogieron lo primero. Si desea una diatriba extensa sobre los defectos del X.400, véase el apéndice C de (Rose, 1993). En consecuencia, nuestro análisis del correo electrónico se enfocará en el RFC 821 y el RFC 822 como se usa en Internet.

7.4.1. Arquitectura y servicios

En esta sección proporcionaremos una panorámica de lo que pueden hacer los sistemas de correo electrónico y la manera en que están organizados. Normalmente consisten en dos subsistemas: los agentes de usuario, que permiten a la gente leer y enviar correo electrónico, y los agentes de transferencia de mensajes, que mueven los mensajes del origen al destino. Los agentes de usuario son programas locales que proporcionan un método basado en comandos, basado en menús o de interacción gráfica con el sistema de correo electrónico. Los agentes de transferencia de mensajes son por lo común daemons del sistema que operan en segundo plano y mueven correo electrónico a través del sistema.

Por lo común, los sistemas de correo electrónico desempeñan cinco funciones básicas, como se describe a continuación. La composición se refiere al proceso de crear mensajes y respuestas. Aunque puede usarse cualquier editor de texto para el cuerpo del mensaje, el sistema mismo puede proporcionar asistencia con el direccionamiento y los numerosos campos de cabecera que forman parte de cada mensaje. Por ejemplo, al contestar un mensaje, el sistema de correo electrónico puede extraer la dirección del originador e insertarla automáticamente en el lugar adecuado de la respuesta.

La transferencia se refiere a mover mensajes del originador al destinatario. En gran medida, esto requiere establecer una conexión con el destino o alguna máquina intermedia, enviar el mensaje y liberar la conexión. El sistema de correo electrónico debe hacer esto automáticamente, sin molestar al usuario.

La generación del informe tiene que ver con indicar al remitente lo que ocurrió con el mensaje: ¿se entregó, se rechazó o se perdió? Existen muchas aplicaciones en las que la confirmación de la entrega es importante y puede incluso tener una implicación legal ("Bien, Su Señoría, mi sistema de correo electrónico no es muy confiable, por lo que creo que la citación electrónica se perdió en algún lado").

La visualización de los mensajes de entrada es necesaria para que la gente pueda leer su correo electrónico. A veces se requiere cierta conversión o debe invocarse un visor especial, por ejemplo si el mensaje es un archivo PostScript o voz digitalizada. También se intentan a veces conversiones y formateo sencillos.

La disposición es el paso final y se refiere a lo que el destinatario hace con el mensaje tras recibirlo. Las posibilidades incluyen tirarlo antes de leerlo, desecharlo tras leerlo, guardarlo, etc. También debe ser posible recuperar y releer mensajes guardados, reenviarlos o procesarlos de otras maneras.

Además de estos servicios básicos, la mayoría de los sistemas de correo electrónico proporcionan una gran variedad de características avanzadas. Brevemente mencionaremos algunas de éstas. Cuando la gente se muda, o cuando está lejos durante cierto tiempo, podría querer el reenvío de su correo electrónico, por lo que el sistema debe ser capaz de hacer esto de manera automática.

La mayoría de los sistemas permite a los usuarios crear buzones de correo (*mailboxes*) para almacenar el correo entrante. Se requieren comandos para crear y destruir buzones de correo, inspeccionar el contenido de los buzones, insertar y borrar mensajes de los buzones, etcétera.

Los gerentes corporativos con frecuencia necesitan enviar un mensaje a cada uno de sus subordinados, clientes o proveedores. Esto da pie al concepto de lista de correo (*mailing list*), que es una lista de direcciones de correo electrónico. Al enviarse un mensaje a la lista de correo, se entregan copias idénticas a todos los de la lista.

El correo registrado es otra idea importante, para permitir al originador saber que su mensaje ha llegado. Como alternativa, podría desearse la notificación automática de correo que no se puede entregar. De cualquier modo, el originador debe tener algún control sobre el informe de lo ocurrido.

Otras características avanzadas son copias al carbón, correo electrónico de alta prioridad, correo electrónico secreto (cifrado), destinatarios alternos si el primario no está disponible y la capacidad de que las secretarías manejen el correo de su jefe.

El correo electrónico se usa ampliamente hoy día en la industria para la comunicación intra-compañía. Permite que los empleados remotos cooperen en proyectos complejos, incluso estando en otros husos horarios. Al eliminar la mayoría de las señales asociadas al rango, edad y género, los debates realizados por correo electrónico tienden a enfocarse principalmente en las ideas, no en el *status* corporativo. Con el correo electrónico, una idea brillante de un estudiante becario puede tener más impacto que una idea tonta de un vicepresidente ejecutivo. Algunas compañías han estimado que el correo electrónico ha mejorado su productividad hasta en 30% (Perry y Adam, 1992).

Una idea clave de todos los sistemas modernos de correo electrónico es la distinción entre la **envoltura** y su contenido. La envoltura encapsula el mensaje; contiene toda la información necesaria para transportar el mensaje, como dirección de destino, prioridad y nivel de seguridad, la cual es diferente del mensaje mismo. Los agentes de transporte del mensaje usan la envoltura para enrutar, al igual que la oficina postal.

El mensaje dentro de la envoltura contiene dos partes: la **cabecera** y el **cuerpo**. La cabecera contiene información de control para los agentes de usuario. El cuerpo es por completo para el destinatario humano. Las envolturas y los mensajes se ilustran en la figura 7-39.

7.4.2. El agente de usuario

Los sistemas de correo electrónico tienen dos partes básicas, como hemos visto: los agentes de usuario y los agentes de transferencia de mensajes. En esta sección veremos a los agentes de usuario. Un agente de usuario normalmente es un programa (a veces llamado lector de correo) que acepta una variedad de comandos para componer, recibir y contestar los mensajes, así como para manipular los buzones de correo. Algunos agentes de usuario tienen una interfaz elegante operada por menús o por iconos que requiere un ratón, mientras que otros esperan comandos de un carácter desde el teclado. Funcionalmente, ambos son iguales.

Envío de correo electrónico

Para enviar un mensaje de correo electrónico, el usuario debe proporcionar el mensaje, la dirección de destino y, posiblemente, algunos otros parámetros (por ejemplo, prioridad o nivel de seguridad). El mensaje puede producirse con un editor de textos independiente, un programa

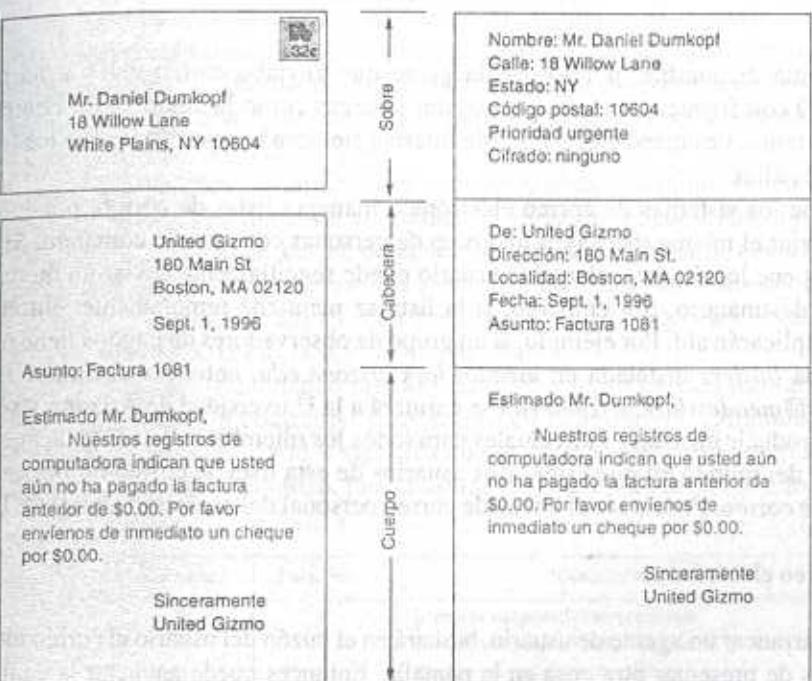


Figura 7-39. Sobres y mensajes. (a) Correo postal. (b) Correo electrónico.

de procesamiento de textos o, posiblemente, un editor de textos incorporado en el agente de usuario. La dirección de destino debe estar en un formato que el agente de usuario pueda manejar. Muchos agentes de usuario esperan direcciones DNS de la forma *buzón_de_correo@ubicación*. Dado que las hemos estudiado antes, no repetiremos ese material aquí.

Sin embargo, vale la pena indicar que existen otras formas de direccionamiento. En particular, las direcciones X.400 tienen un aspecto radicalmente diferente del de las direcciones DNS; se componen de pares *atributo = valor*, por ejemplo,

/C=US/SP=MASSACHUSETTS/L=CAMBRIDGE/PA=360 MEMORIAL DR./CN=KEN SMITH/

Esta dirección especifica un país, estado, localidad, dirección personal y nombre común (Ken Smith). Son posibles muchos otros atributos, de modo que puede enviarse correo electrónico a alguien cuyo nombre no se conoce, siempre y cuando se conozca un número suficiente de los otros atributos (por ejemplo, compañía y puesto). Mucha gente piensa que esta forma de nombrar es considerablemente menos funcional que los nombres DNS.

Sin embargo, con toda justicia, los diseñadores del X.400 supusieron que la gente usaría alias (cadenas cortas asignadas por el usuario) para identificar a los destinatarios, por lo que nunca verían siquiera las direcciones completas. Sin embargo, el software necesario nunca

estuvo ampliamente disponible, por lo que la gente que enviaba correo a los usuarios con direcciones X.400 con frecuencia tenía que escribir cadenas como la anterior. En contraste, la mayoría de los sistemas de correo electrónico de Internet siempre han permitido que los usuarios tengan archivos de alias.

La mayoría de los sistemas de correo electrónico manejan listas de correo, por lo que un usuario puede enviar el mismo mensaje a un grupo de personas con un solo comando. Si la lista de correo se mantiene localmente, el agente usuario puede sencillamente enviar un mensaje por separado a cada destinatario. Sin embargo, si la lista se mantiene remotamente, entonces los mensajes se multiplicarán ahí. Por ejemplo, si un grupo de observadores de pájaros tiene una lista de correo llamada *birders* instalada en *meadowlark.arizona.edu*, entonces cualquier mensaje enviado a *birders@meadowlark.arizona.edu* se enrutaría a la Universidad de Arizona y se multiplicaría ahí para producir mensajes individuales para todos los miembros de la lista de correo, sin importar el lugar del mundo en que estén. Los usuarios de esta lista de correo no pueden saber que es una lista de correo. Podría ser el buzón de correo personal del profesor Gabriel O. Birders.

Lectura del correo electrónico

Por lo común, al arrancar un agente de usuario, buscará en el buzón del usuario el correo electrónico recibido, antes de presentar otra cosa en la pantalla. Entonces puede anunciar la cantidad de mensajes en el buzón o presentar un resumen de una línea de cada uno y esperar un comando.

Como ejemplo del funcionamiento de un agente de usuario, veamos una situación típica de correo. Tras iniciar el agente de usuario, el usuario solicita un resumen de su correo electrónico. Aparece entonces una pantalla como la de la figura 7-40. Cada línea se refiere a un mensaje. En este ejemplo, el buzón contiene ocho mensajes.

Cada línea de la pantalla contiene varios campos extraídos del sobre o de la cabecera del mensaje correspondiente. En un sistema sencillo de correo electrónico, la selección de campos a presentar está incorporada en el programa. En uno más refinado, el usuario puede especificar los campos a presentar proporcionando un *perfil de usuario*, un archivo que describe el formato de

#	Indicadores	Bytes	Remitente	Asunto
1	K	1030	asw	Cambios a MINIX
2	KA	6348	radia	Comentarios sobre el material que me enviste
3	K F	4519	Amy N. Wong	Solicitud de información
4		1236	bal	Fecha límite para propuesta de subvención
5		103620	kaashook	Texto del documento DCS
6		1223	emily E.	Apuntador a página de WWW
7		3110	sanya	Informes de arbitraje sobre el documento
8		1204	dmr	Re: Visita de mis estudiantes

Figura 7-40. Ejemplo de exhibición del contenido de un buzón.

presentación. En este ejemplo, el primer campo es el número de mensaje. El segundo campo, *indicadores*, puede contener una *K*, lo que significa que el mensaje no es nuevo, sino que ha sido leído previamente y guardado en el buzón; una *A*, lo que quiere decir que el mensaje ya ha sido contestado; y/o una *F*, lo que indica que ese mensaje ha sido reenviado a alguien más. Son posibles también otros indicadores.

El tercer campo indica la longitud del mensaje y el cuarto indica quién envió el mensaje. Dado que este campo simplemente se extrae del mensaje, puede contener nombres de pila, nombres completos, iniciales, claves de acceso o cualquier otra cosa que el remitente decida poner ahí. Por último, el campo de *asunto* da un resumen breve del tema del mensaje. La gente que no incluye un campo de *asunto* con frecuencia descubre que las respuestas a su correo electrónico tienden a no recibir la prioridad más alta.

Una vez que se han visualizado las cabeceras, el usuario puede ejecutar cualquiera de los comandos disponibles. Un conjunto típico se lista en la figura 7-41. Algunos de los comandos requieren un parámetro. El signo # significa que se espera el número de un mensaje (o tal vez varios mensajes). Como alternativa, puede usarse la letra *a* para referirse a todos los mensajes.

Comando	Parámetro	Descripción
<i>h</i>	#	Presenta cabecera(s) en la pantalla
<i>c</i>	#	Presenta cabecera actual únicamente
<i>t</i>		Tipo de mensaje(s) en la pantalla
<i>s</i>	dirección	Envía mensaje
<i>f</i>	#	Reenvía mensaje(s)
<i>a</i>	#	Contesta mensaje(s)
<i>d</i>	#	Borra mensaje(s)
<i>u</i>	#	Recupera mensaje(s) previamente borrados
<i>m</i>	#	Mueve mensaje(s) a otro buzón
<i>k</i>	#	Guarda mensaje(s) tras salir
<i>r</i>	buzón	Lee un buzón nuevo
<i>n</i>		Va al siguiente mensaje y lo presenta
<i>b</i>		Regresa al mensaje previo y lo presenta
<i>g</i>	#	Va a un mensaje específico, pero no lo presenta
<i>e</i>		Sale del sistema de correo y actualiza el buzón

Figura 7-41. Comandos típicos de manejo de correo.

Existen incontables programas de correo electrónico. Nuestro programa de ejemplo está modelado según el usado por el sistema Mmif de UNIX, y es bastante sencillo. El comando *h* presenta una o más cabeceras con el formato de la figura 7-40. El comando *c* imprime la cabecera del mensaje actual. El comando *t* presenta (visualiza) en la pantalla el mensaje o mensajes requeridos. Los comandos posibles son *t 3* para presentar el mensaje 3, *t 4-6* para presentar los mensajes 4 a 6, y *t a* para presentarlos todos.



El siguiente grupo de tres comandos tiene que ver con el envío de mensajes en lugar de su recepción. El comando *s* envía un mensaje llamando a un editor adecuado (por ejemplo, el indicado en el perfil de usuario) para que el usuario pueda elaborar el mensaje. Mediante revisores ortográficos, de gramática y de dicción se puede comprobar que el mensaje sea sintácticamente correcto. Desafortunadamente, la generación actual de programas de correo electrónico no tiene revisores para ver si el remitente sabe de lo que está hablando. Cuando se ha terminado el mensaje, está listo para su transmisión al agente de transferencia de mensajes.

El comando *freenvf* envía un mensaje desde el buzón, solicitando una dirección a la cual enviarlo. El comando *a* extrae la dirección de origen del mensaje a ser contestado y llama al editor para que el usuario pueda elaborar la contestación.

El siguiente grupo de comandos es para manipular los buzones de correo. Los usuarios por lo regular tienen un buzón por cada usuario con el que mantienen correspondencia, además del buzón para el correo electrónico de entrada que ya hemos visto. El comando *d* borra un mensaje del buzón, pero el comando *u* deshace el borrado. (El mensaje en realidad no se borra sino hasta que el usuario sale del programa de correo electrónico.) El comando *m* transfiere un mensaje a otro buzón. Ésta es la manera normal de guardar correo electrónico importante una vez leído. El comando *k* guarda el mensaje indicado en el buzón después de leerse. Si se lee un mensaje pero no se guarda explícitamente, se emprende alguna acción predeterminada al salir del programa de correo electrónico, como pasarlo a un buzón especial predeterminado. Por último, el comando *r* sirve para terminar con el buzón actual y pasar a otro.

Los comandos *n*, *b* y *g* son para moverse dentro del buzón actual. Es común que un usuario lea el mensaje *l*, lo conteste, mueva o borre, y luego indique *n* para obtener el siguiente. La utilidad de este comando es que el usuario no tiene que llevar el registro de dónde está. Es posible ir hacia atrás usando *b* o a un mensaje específico mediante *g*.

Por último, el comando *e* termina el programa de correo electrónico y hace los cambios necesarios, como borrar algunos mensajes y marcar otros como guardados. Este comando sobreescribe el buzón, reemplazando su contenido.

En los sistemas de correo electrónico diseñados para principiantes, cada uno de estos comandos típicamente está asociado a un ícono de la pantalla, por lo que el usuario no tiene que recordar que *a* significa *respuesta*. En cambio, tiene que recordar que la pequeña imagen de una persona con su boca abierta significa respuesta y no visualizar mensaje.

Debe quedar claro por este ejemplo que el correo electrónico ha recorrido un largo camino desde los días en que era simplemente una transferencia de archivos. Los agentes de usuario refinados hacen posible la administración de grandes volúmenes de correo electrónico. Para la gente como el autor, que (a su pesar) recibe y envía miles de mensajes al año, tales herramientas son invaluables.

7.4.3. Formatos de mensaje

Pasemos ahora de la interfaz de usuario al formato de los mensajes de correo electrónico mismos. Primero veremos el correo electrónico ASCII básico usando el RFC 822. Tras eso, veremos las extensiones multimedia del RFC 822.

RFC 822

Los mensajes consisten en una envoltura primitiva (descrita en el RFC 821), algunos campos de cabecera, una línea en blanco, y el cuerpo del mensaje. Cada campo de cabecera consiste (lógicamente) en una sola línea de texto ASCII que contiene el nombre del campo, dos puntos (:) y, para la mayoría de los campos, un valor. El RFC 822 es un estándar viejo y no distingue claramente la envoltura de los campos de cabecera, como lo haría un estándar nuevo. En el uso normal, el agente de usuario construye un mensaje y lo pasa al agente de transferencia de mensajes, quien entonces usa algunos campos de la cabecera para construir la envoltura, una mezcla un tanto anticuada de mensaje y envoltura.

Los principales campos de cabecera relacionados con el transporte del mensaje se listan en la figura 7-42. El campo *To:* indica la dirección DNS del destinatario primario. Está permitido tener varios destinatarios. El campo *Cc:* indica la dirección de los destinatarios secundarios. En términos de entrega, no hay diferencia entre los destinatarios primarios y los secundarios. Es una diferencia por entero sicológica que puede ser importante para los participantes, pero que no lo es para el sistema de correo. El término *Cc:* (copia al carbón) es un poco anticuado, puesto que las computadoras no usan papel carbón, pero está muy establecido. El campo *Bcc:* (copia al carbón ciega) es como el campo *Cc:*, excepto que esta línea se borra de todas las copias enviadas a los destinatarios primarios y secundarios. Esta característica permite a la gente mandar copias a terceros sin que los destinatarios primarios y secundarios lo sepan.

Cabecera	Significado
<i>To:</i>	Direcciones de email de los destinatarios primarios
<i>Cc:</i>	Direcciones de email de los destinatarios secundarios
<i>Bcc:</i>	Direcciones de email para las copias al carbón ciegas
<i>From:</i>	Persona o personas que crearon el mensaje
<i>Sender:</i>	Dirección de correo electrónico del remitente
<i>Received:</i>	Línea agregada por cada agente de transferencia en la ruta
<i>Return-Path:</i>	Puede usarse para identificar una trayectoria de regreso al remitente

Figura 7-42. Campos de cabecera RFC 822 relacionados con el transporte de mensajes.

Los siguientes dos campos, *From:* y *Sender:* indican quién escribió y envió el mensaje, respectivamente; pueden ser diferentes. Por ejemplo, un ejecutivo de negocios puede escribir un mensaje, pero su secretaria podría ser la que lo transmite. En este caso, el ejecutivo estaría listado en el campo *From:* y la secretaria en el campo *Sender:*. Se requiere el campo *From:*, pero el campo *Sender:* puede omitirse si es igual al campo *From:*. Estos campos se requieren en caso de que el mensaje no pueda entregarse y deba devolverse al remitente.

Se agrega una línea que contiene *Received*: por cada agente de transferencia de mensajes en el camino. La línea contiene la identidad del agente, la fecha y hora de recepción del mensaje, y otra información que puede servir para encontrar fallas en el sistema de enrutamiento.

El campo *Return-Path*: lo agrega el agente de transferencia de mensajes final y su intención era indicar la manera de regresar al remitente. En teoría, esta información puede tomarse de todas las cabeceras *Received*: (exceptuando el nombre del buzón del remitente), pero pocas veces se llena de esta manera y comúnmente contiene sólo la dirección del remitente.

Además de los campos de la figura 7-42, los mensajes RFC 822 también pueden contener una variedad de campos de cabecera usados por los agentes de usuario o los destinatarios. Los más comunes se listan en la figura 7-43. La mayoría de ellos son autoexplicativos, por lo que no los veremos en detalle.

Cabecera	Significado
Date:	Fecha y hora de envío del mensaje
Reply-To:	Dirección de email a la que deben enviarse las contestaciones
Message-Id:	Número único para referencia posterior a este mensaje
In-Reply-To:	Identificador del mensaje al que éste responde
References:	Otros identificadores de mensaje pertinentes
Keywords:	Claves seleccionadas por el usuario
Subject:	Resumen corto del mensaje para exhibir en una línea

Figura 7-43. Algunos campos usados en la cabecera de mensaje RFC 822.

El campo *Reply-To*: a veces se usa cuando ni la persona que redactó el mensaje ni la que lo envió quieren ver la respuesta. Por ejemplo, un administrador de mercadeo escribe un mensaje de correo electrónico para informar a los clientes sobre un producto nuevo. El mensaje lo envía una secretaria, pero el campo *Reply-To*: indica el jefe del departamento de ventas, quien puede contestar preguntas y surtir pedidos.

El documento RFC 822 explícitamente indica que los usuarios pueden inventar cabeceras nuevas para su uso privado, siempre y cuando comiencen con la cadena *X-*. Se garantiza que no habrá cabeceras futuras que usen nombres que comiencen con *X-*, a fin de evitar conflictos entre las cabeceras oficiales y las privadas. A veces, algunos estudiantes universitarios, pasándose de listos, incluyen campos como *X-Fruta-del-Día*: o *X-Enfermedad-de-la-Semana*:; que son legales, aunque no muy ilustrativos.

Tras las cabeceras viene el cuerpo del mensaje. Los usuarios pueden poner aquí lo que les venga en gana. Algunos terminan sus mensajes con firmas complicadas, incluidas caricaturas sencillas en ASCII, citas de autoridades mayores y menores, pronunciamientos políticos y renuncias de responsabilidades de todo tipo (por ejemplo, la corporación ABC no es responsable de mis opiniones; ni siquiera las puede entender).

MIME — Extensiones multipropósito de correo Internet

En los primeros días de ARPANET, el correo electrónico consistía exclusivamente en mensajes de texto escritos en inglés y expresados en ASCII. En tal entorno, el RFC 822 hacía todo el trabajo: especificaba las cabeceras, pero dejaba el contenido en manos del usuario. Hoy día, en la red mundial de Internet, este enfoque ya no es adecuado. Los problemas incluyen envío y recepción de:

1. Mensajes en idiomas con acentos (por ejemplo, español, francés y alemán).
2. Mensajes en alfabetos no latinos (por ejemplo, hebreo y ruso).
3. Mensajes en idiomas sin alfabetos (por ejemplo, chino y japonés).
4. Mensajes que no contienen texto (por ejemplo, audio y vídeo).

Se propuso una solución en el RFC 1341 y se actualizó en el RFC 1521. Esta solución, llamada **MIME** (*Multipurpose Internet Mail Extensions*; extensiones multipropósito de correo Internet), se usa ampliamente. Ahora la describiremos. Para información adicional sobre MIME, véase el RFC 1521 o (Rose, 1993).

La idea básica de MIME es continuar usando el formato RFC 822, pero agregar una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII. Al no desviarse del 822, los mensajes MIME pueden enviarse usando los programas y protocolos de correo electrónico existentes. Todo lo que tiene que cambiarse son los programas transmisores y receptores, lo que pueden hacer los usuarios mismos.

El MIME define cinco nuevas cabeceras de mensaje, como se muestra en la figura 7-44. La primera de éstas simplemente indica al agente de usuario receptor del mensaje que está tratando con un mensaje MIME, así como la versión del MIME que usa. Se considera que cualquier mensaje que no contenga una cabecera *MIME-Version*: es un mensaje de texto normal en inglés, y se procesa como tal.

La cabecera *Content-Description*: es una cadena ASCII que dice lo que está en el mensaje. Esta cabecera es necesaria para que el destinatario sepa si vale la pena descodificar y leer el mensaje. Si la cadena dice: "Foto del jerbo de Bárbara" y la persona que recibe el mensaje no es un gran fanático de los jerbos, el mensaje probablemente será descartado en lugar de descodificado para dar una foto a color de alta definición.

Cabecera	Significado
MIME-Version:	Identifica la versión de MIME
Content-Description:	Cadena de texto que describe el contenido
Content-Id:	Identificador único
Content-Transfer-Encoding:	Cómo se envuelve el mensaje para su transmisión
Content-Type:	Naturaleza del mensaje

Figura 7-44. Cabeceras RFC 822 agregadas por MIME.

La cabecera *Content-Id*: identifica el contenido; usa el mismo formato que la cabecera estándar *Message-Id*.

Content-Transfer-Encoding: indica la manera en que está envuelto el cuerpo para su transmisión a través de una red donde se podría tener problemas con la mayoría de los caracteres distintos de las letras, números y signos de puntuación. Se proporcionan cinco esquemas (más un escape hacia nuevos esquemas). El esquema más sencillo es simplemente texto ASCII. Los caracteres ASCII usan 7 bits y pueden transportarse directamente mediante el protocolo de correo electrónico siempre y cuando ninguna línea exceda 1000 caracteres.

El siguiente esquema más sencillo es lo mismo, pero usando caracteres de 8 bits, es decir, todos los valores de 0 a 255. Este esquema de codificación viola el protocolo (original) del correo electrónico de Internet, pero es usado por algunas partes de Internet que implementan ciertas extensiones al protocolo original. Mientras que la declaración de la codificación no la hace legal, hacerla explícita puede cuando menos explicar las cosas cuando algo sucede mal. Los mensajes que usan codificación de 8 bits deben aún adherirse a la longitud máxima de línea estándar.

Peor aún son los mensajes que usan codificación binaria. Éstos son archivos binarios arbitrarios que no sólo utilizan los 8 bits, sino que ni siquiera respetan el límite de 1000 caracteres por línea. Los programas ejecutables caen en esta categoría. No se da ninguna garantía de que los mensajes en binario llegarán correctamente, pero mucha gente los envía de todos modos.

La manera correcta de codificar mensajes binarios es usar codificación base64, a veces llamada **armadura ASCII**. En este esquema, se dividen grupos de 24 bits en unidades de 6 bits, enviándose cada unidad como carácter ASCII legal. La codificación es "A" para 0, "B" para 1, etc., seguidas por las 26 letras minúsculas, los 10 dígitos y, por último, + y / para el 62 y 63, respectivamente. Las secuencias == y = se usan para indicar que el último grupo contenía sólo 8 o 16 bits, respectivamente. Los retornos de carro y avances de línea se ignoran, por lo que pueden introducirse a voluntad para mantener la línea lo suficientemente corta. Puede enviarse un texto binario arbitrario usando este esquema.

En el caso de mensajes que son casi completamente ASCII, pero con algunos caracteres no ASCII, la codificación base 64 es algo ineficiente. En cambio, se usa una codificación conocida como codificación entrecomillada-imprimible. Simplemente es ASCII de 7 bits, con todos los caracteres por encima de 127 codificados como un signo de igual seguido del valor del carácter en dos dígitos hexadecimales.

En resumen, los datos binarios deben enviarse codificados en base 64 o en forma entrecomillada-imprimible. Cuando hay razones válidas para no usar uno de estos esquemas, es posible especificar una codificación definida por el usuario en la cabecera *Content-Transfer-Encoding*:

La última cabecera mostrada en la figura 7-44 en realidad es la más interesante. Especifica la naturaleza del cuerpo del mensaje. Hay siete tipos definidos en el RFC 1521, cada uno de los cuales tiene uno o más subtipos. El tipo y el subtipo se separan mediante una diagonal, como en

Content-Type: video/mpeg

El subtipo debe indicarse explícitamente en la cabecera; no se proporcionan valores predeterminados. La lista inicial de tipos y subtipos especificada en el RFC 1521 se da en la figura 7-45. Se

han agregado muchos otros desde entonces, y se agregan entradas nuevas todo el tiempo, a medida que surge la necesidad.

Tipo	Subtipo	Descripción
Text	Plain	Texto sin formato
	Richtext	Texto con comandos de formato sencillos
Image	Gif	Imagen fija en formato GIF
	Jpeg	Imagen fija en formato JPEG
Audio	Basic	Sonido
Video	Mpeg	Película en formato MPEG
Application	Octet-stream	Secuencia de bytes no interpretada
	Postscript	Documento imprimible en PostScript
Message	Rfc822	Mensaje MIME RFC 822
	Partial	Mensaje dividido para su transmisión
	External-body	El mensaje mismo debe obtenerse de la red
Multipart	Mixed	Partes independientes en el orden especificado
	Alternative	Mismo mensaje en diferentes formatos
	Parallel	Las partes deben verse simultáneamente
	Digest	Cada parte es un mensaje RFC 822 completo

Figura 7-45. Tipos y subtipos MIME definidos en el RFC 1521.

Repasemos ahora la lista de tipos. El tipo *text* es para texto normal. La combinación *text/plain* es para mensajes ordinarios que pueden visualizarse como se reciben, sin codificación ni ningún procesamiento posterior. Esta opción permite el transporte de mensajes ordinarios en MIME con sólo unas pocas cabeceras extra.

El subtipo *text/richtext* permite la inclusión de un lenguaje de marcación sencillo en el texto. Este lenguaje proporciona una manera independiente del sistema para indicar negritas, cursivas, tamaños en puntos menores y mayores, sangrías, justificaciones, sub y superíndices, y formatos sencillos de página. El lenguaje de marcación se basa en el SGML, el *Standard Generalized Markup Language*, lenguaje generalizado estándar de marcación, también usado como la base del HTML del *World Wide Web*. Por ejemplo, el mensaje

Ha llegado el momento, dijo la <i>morsa</i>...
se presentaría como

Ha llegado el momento, dijo la morsa...

Es responsabilidad del sistema receptor seleccionar la presentación adecuada. Si hay negritas y cursivas disponibles, pueden usarse; de otra manera, pueden usarse colores, parpadeos, video inverso, etc., como énfasis. Los diferentes sistemas pueden, y de hecho lo hacen, hacer selecciones distintas.

El siguiente tipo MIME es *image*, que se usa para transmitir imágenes fijas. Hoy día se usan muchos formatos para almacenar y transmitir imágenes, tanto con compresión como sin ella. Dos de estos, GIF y JPEG son subtipos oficiales, pero sin duda se agregarán otros después.

Los tipos *audio* y *video* son para sonido e imágenes en movimiento, respectivamente. Nótese que *video* incluye sólo la información visual, no la pista de sonido. Si se debe transmitir una película con sonido, tal vez sea necesario transmitir las partes de video y de audio por separado, dependiendo del sistema de codificación usado. El único formato de video definido hasta ahora es el diseñado por el grupo de modesto nombre *Moving Picture Experts Group* (grupo de expertos en imágenes en movimiento), el MPEG.

El tipo *application* es un tipo general para los formatos que requieren procesamiento externo no cubierto por ninguno de los otros tipos. Un *octet-stream* simplemente es una secuencia de bytes no interpretados. A la recepción de una de tales cadenas, un agente de usuario probablemente debería presentarla en pantalla sugiriendo al usuario que se copie en un archivo y solicitando un nombre de archivo. El procesamiento posterior es responsabilidad del usuario.

El otro subtipo definido es *postscript*, que se refiere al lenguaje PostScript producido por Adobe Systems y ampliamente usado para describir páginas impresas. Muchas impresoras tienen intérpretes PostScript integrados. Aunque un agente de usuario puede simplemente llamar a un intérprete PostScript externo para visualizar los archivos PostScript entrantes, hacerlo no está exento de riesgos. PostScript es un lenguaje de programación completo. Con el tiempo necesario, una persona lo bastante masoquista podría escribir un compilador de C o un sistema de manejo de base de datos en PostScript. La visualización de un mensaje PostScript entrante puede hacerse ejecutando el programa PostScript contenido en él. Además de exhibir algo de texto, este programa puede leer, modificar o borrar los archivos del usuario, y tener otros efectos secundarios desagradables.

El tipo *message* permite que un mensaje esté encapsulado por completo dentro de otro. Este esquema es útil para reenviar, por ejemplo, correo electrónico. Cuando se encapsula un mensaje RFC 822 completo en un mensaje exterior, debe usarse el subtipo *rfc822*.

El subtipo *partial* hace posible dividir un mensaje encapsulado en pedazos y enviarlos por separado (por ejemplo, si el mensaje encapsulado es demasiado grande). Los parámetros hacen posible reensamblar correctamente todas las partes en el destino en el orden correcto.

Por último, el subtipo *external-body* puede usarse para mensajes muy grandes (por ejemplo, películas en vídeo). En lugar de incluir el archivo MPEG en el mensaje, se da una dirección FTP y el agente de usuario del receptor puede obtenerlo a través de la red al momento que se requiera. Esta facilidad es especialmente útil al enviar una película a una lista de correo, cuando sólo se espera que unos cuantos la vean (imagine correo electrónico chatarra que contenga videos de propaganda).

El último tipo es *multipart*, que permite que un mensaje contenga más de una parte, con el comienzo y el fin de cada parte claramente delimitados. El subtipo *mixed* permite que cada parte sea diferente, sin ninguna estructura adicional impuesta. En contraste, con el subtipo *alternative*, cada parte debe contener el mismo mensaje, pero expresado en un medio o codificación diferente. Por ejemplo, un mensaje puede enviarse como ASCII común, como texto enriquecido (*richtext*) y como PostScript. Un agente de usuario adecuadamente diseñado que reciba uno de tales mensajes lo presentará en PostScript de ser posible. La segunda posibilidad sería como texto enriquecido. Si ninguna de las dos fuera posible, se presentaría el texto ASCII normal. Las partes deben ordenarse de la más sencilla a la más compleja para ayudar a que los receptores con agentes de usuario pre-MIME puedan encontrarle algún sentido al mensaje (por ejemplo, aun un usuario pre-MIME puede leer texto ASCII común).

El subtipo *alternative* puede servir también para varios lenguajes. En este contexto, puede pensarse en la piedra Rosetta como uno de los primeros mensajes *multipart/alternative*.

Un ejemplo multimedia se muestra en la figura 7-46. Aquí se transmiten una felicitación de cumpleaños como texto y como canción. Si el receptor tiene capacidad de audio, el agente de usuario traerá el archivo de sonido, *birthday.snd*, y lo ejecutará. Si no, se presenta la letra en la pantalla en absoluto silencio. Las partes están delimitadas por dos guiones seguidos de la cadena (definida por el usuario) especificada en el parámetro *boundary*.

Nótese que la cabecera *Content-Type* aparece en tres lugares en este ejemplo. En el nivel superior, la cabecera indica que el mensaje tiene varias partes. En cada parte, la cabecera indica el tipo y subtipo de la parte. Por último, en el texto de la segunda parte, la cabecera es necesaria para indicarle al agente de usuario la clase de archivo externo que debe conseguir. Para indicar esta pequeña diferencia de uso, hemos usado letras en minúscula, aunque las cabeceras no hacen distinciones entre mayúsculas y minúsculas. De la misma manera, se requiere la cabecera *content-transfer-encoding* para cualquier cuerpo externo que no esté codificado como ASCII de 7 bits.

Regresando a los subtipos para mensajes multipart, existen dos posibilidades más. El subtipo *parallel* se usa cuando todas las partes deben "verse" simultáneamente. Por ejemplo, las películas con frecuencia tienen un canal de audio y un canal de video. Las películas son más efectivas si estos dos canales se reproducen en paralelo, en lugar de consecutivamente.

Por último, el subtipo *digest* se usa cuando se empacan muchos mensajes juntos en un mensaje compuesto. Por ejemplo, algunos grupos de discusión de Internet recolectan mensajes de los subscriptores y luego los envían como un solo mensaje *multipart/digest*.

7.4.4. Transferencia de mensajes

El sistema de transferencia de mensajes se ocupa de transmitir los mensajes del remitente al destinatario. La manera más sencilla de hacer esto es establecer una conexión de transporte de la máquina de origen a la de destino y sencillamente transferir el mensaje. Tras examinar la manera en que se hace normalmente esto, estudiaremos algunas situaciones en las que no funciona esto, y lo que puede hacerse al respecto.

```

From: elinor@abc.com
To: carolyn@xyz.com
MIME-Version: 1.0
Message-ID: <0704760941.AA00747@abc.com>
Content-Type: multipart/alternative; boundary=qwertyuiopasdflghjklzxcvbnm
Subject: La Tierra orbita al Sol un número entero de veces

```

Este es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.

```

--qwertyuiopasdflghjklzxcvbnm
Content-Type: text/richtext

```

```

Feliz cumpleaños a ti
Feliz cumpleaños a ti
Feliz cumpleaños, querida <b>Carolyn</b> <b>Carolyn</b>
Feliz cumpleaños a ti

```

```

--qwertyuiopasdflghjklzxcvbnm
Content-Type: message/external-body;
access-type="anon-ftp",
site="bicycle.abc.com";
directory="pub";
name="birthday.snd"

```

```

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdflghjklzxcvbnm--

```

Figura 7-46. Mensaje multiparte que contiene alternativas de texto enriquecido y audio.

SMTP — Protocolo sencillo de transferencia de correo

En el Internet, el correo electrónico se entrega al hacer que la máquina de origen establezca una conexión TCP con el puerto 25 de la máquina de destino. Escuchando en este puerto está un *daemon* de correo electrónico que habla con el SMTP (*Simple Mail Transfer Protocol*, protocolo sencillo de transferencia de correo). Este *daemon* acepta conexiones de entrada y copia mensajes de ellas a los buzones adecuados. Si no puede entregarse un mensaje, se devuelve al transmisor un informe de error que contiene la primera parte del mensaje que no pudo entregarse.

El SMTP es un protocolo ASCII sencillo. Tras establecer la conexión TCP con el puerto 25, la máquina transmisora, operando como cliente, espera que la máquina receptora, operando como servidor, hable primero. El servidor comienza por enviar una línea de texto que proporciona su identidad e indica si está preparado o no para recibir correo. Si no lo está, el cliente libera la conexión y lo intenta después.

Si el servidor está dispuesto a aceptar correo electrónico, el cliente anuncia de quién viene el mensaje, y a quién está dirigido. Si existe tal destinatario en el destino, el servidor da al cliente permiso para enviar el mensaje. Entonces el cliente envía el mensaje y el servidor acusa su recibo. Por lo general no se requieren sumas de comprobación porque el TCP proporciona una corriente de bits confiable. Si hay más correo electrónico, se envía ahora. Una vez que todo el correo electrónico ha sido intercambiado en ambas direcciones, se libera la conexión. Un ejemplo de diálogo para el envío del mensaje de la figura 7-46, incluidos los códigos numéricos usados por el SMTP, se muestra en la figura 7-47. Las líneas enviadas por el cliente se marcan con C:, aquellas enviadas por el servidor se marcan con S:

Pueden ser útiles algunos comentarios sobre la figura 7-47. El primer comando del cliente ciertamente es *HELO*. De las dos abreviaturas de cuatro caracteres de *HELLO*, ésta tiene numerosas ventajas sobre su competidora.* La razón por la que los comandos tienen que ser de cuatro caracteres se ha perdido en las brumas del tiempo.

En la figura 7-47, el mensaje se envía a un solo receptor, por lo que se usa un solo comando *RCPT*. Se permiten muchos de tales comandos para enviar un solo mensaje a receptores múltiples; cada uno se acusa o rechaza individualmente. Incluso si se rechazan algunos destinatarios (porque no existen en el destino), el mensaje puede enviarse a los demás.

Por último, aunque la sintaxis de los comandos de cuatro caracteres del cliente se especifica con rigidez, la sintaxis de las respuestas es menos rígida. Sólo cuenta el código numérico. Cada implementación puede poner la cadena que deseé después del código.

Aunque el protocolo SMTP está bien definido (por el RFC 821), pueden surgir algunos problemas. Uno se relaciona con la longitud del mensaje. Algunas implementaciones más viejas no pueden manejar mensajes mayores que 64 kB. Otro problema se relaciona con las terminaciones de temporización. Si el cliente y el servidor tienen temporizaciones diferentes, uno de ellos puede terminar mientras que el otro continúa trabajando, terminando inesperadamente la conexión. Por último, en contadas ocasiones, pueden dispararse tormentas de correo infinitas. Por ejemplo, si el *host* 1 contiene la lista de correo A y el *host* dos contiene la lista de correo B y cada lista contiene una entrada para la otra lista, entonces un mensaje enviado a cualquiera de las listas generará una cantidad sin fin de tráfico de correo electrónico.

Para superar algunos de estos problemas, se ha definido el SMTP extendido (ESMTP) en el RFC 1425. Los clientes que deseen usarlo deben enviar inicialmente un mensaje *EHLO*, en lugar de *HELO*. Si el saludo se rechaza, esto indica que el servidor es un servidor SMTP normal, y el cliente debe proceder de la manera normal. Si se acepta el *EHLO*, entonces se permiten los comandos y parámetros nuevos. La estandarización de estos comandos y parámetros es un proceso en curso.

Pasarelas de correo electrónico

El correo electrónico usando SMTP funciona mejor cuando tanto el transmisor como el receptor están en Internet y pueden manejar conexiones TCP entre el transmisor y el receptor. Sin

* En inglés, *hell* significa "infierno". (N. del supervisor.)

```

S: 220 servicio SMTP xyz.com listo
C: HELO abc.com
S: 250 xyz.com dice hola a abc.com
C: MAIL FROM: <elinor@abc.com>
S: 250 transmisor ok
C: RCPT TO: <carolyn@xyz.com>
S: 250 receptor ok
C: DATA
S: 354 envía correo; termina con una linea únicamente con "."
C: From: elinor@abc.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abc.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: La Tierra orbita al Sol un número entero de veces
C:
C: Éste es el preámbulo. El agente de usuario lo ignora. Tenga un bonito día.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/richtext
C:
C: Feliz cumpleaños a tí
C: Feliz cumpleaños a tí
C: Feliz cumpleaños, querida <b>Carolyn</b>
C: Feliz cumpleaños a tí
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C: access-type="anon-ftp";
C: site="bicycle.abc.com";
C: directory="pub";
C: name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C:
S: 250 mensaje aceptado
C: QUIT
S: 221 xyz.com cerrando conexión

```

Figura 7-47. Transferencia de un mensaje de *elinor@abc.com* a *carolyn@xyz.com*.

embargo, muchas máquinas que no están en Internet también quieren transmitir y recibir correo electrónico de instalaciones de Internet. Por ejemplo, muchas compañías intencionalmente no quieren estar en Internet por razones de seguridad. Algunas de ellas incluso se aíslan de Internet levantando muros de seguridad entre ellas mismas y la Internet.

Ocurre otro problema cuando el transmisor habla sólo RFC 822 y el otro habla sólo un protocolo de correo X.400 o alguno patentado, específico del proveedor. Dado que estos dos mundos difieren en cuanto al formato de mensaje y los protocolos, es imposible la comunicación directa.

Ambos problemas se resuelven usando pasarelas de correo electrónico. En la figura 7-48, el *host 1* sólo habla TCP/IP y RFC 822, mientras que el *host 2* habla sólo TP4 y X.400 de OSI. No obstante, pueden intercambiar correo usando una pasarela de correo electrónico. El procedimiento es que el *host 1* establece una conexión TCP con la pasarela y luego usa SMTP para transferir un mensaje (1) ahí. El *daemon* de la pasarela pone el mensaje en el *buffer de mensajes* destinados al *host 2*. Después, se establece una conexión TP4 (el equivalente OSI del TCP) con el *host 2* y el mensaje (2) se transfiere usando el equivalente OSI del SMTP. Todo lo que tiene que hacer el proceso de la pasarela es extraer los mensajes entrantes de una cola y depositarlos en otra.

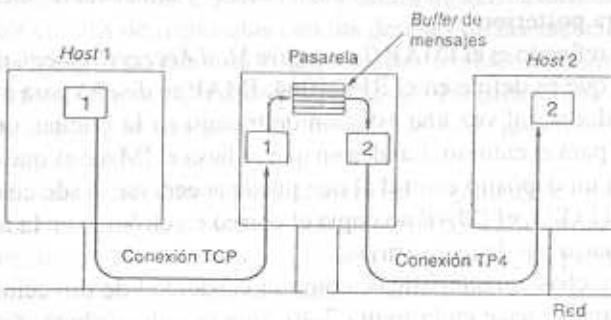


Figura 7-48. Transferencia de correo electrónico usando una pasarela de correo electrónico de capa de aplicación.

Parece fácil, pero no lo es. El primer problema es que las direcciones Internet y las direcciones X.400 son diferentes por completo. Se requiere un complejo mecanismo de transformación entre ellas. El segundo problema es que los campos de envoltura y de cabecera presentes en un sistema podrían no estar presentes en el otro. Por ejemplo, si un sistema requiere clases de prioridad y el otro no tiene este concepto, en una dirección debe desecharse información valiosa y en la otra debe generarse de la nada.

Un problema más grave aún es qué debe hacerse si son incompatibles partes del cuerpo. ¿Qué debe hacer una pasarela con un mensaje de Internet cuyo cuerpo contiene una referencia a un archivo de audio que debe ser obtenido mediante FTP si el sistema de destino no maneja este concepto? ¿Qué debe hacer la pasarela cuando un sistema X.400 le dice que entregue un mensaje a cierta dirección pero, si falla, envíe el contenido por fax? El uso del fax no es parte del modelo RFC 822. Queda claro que aquí no hay soluciones sencillas. Para los mensajes de texto sencillos sin estructura en ASCII, las pasarelas son una solución razonable, pero para cualquier cosa más complicada la idea tiende a desmoronarse.

Entrega final

Hasta ahora, hemos supuesto que todos los usuarios trabajan en máquinas capaces de enviar y recibir correo electrónico. Con frecuencia esta situación es falsa. Por ejemplo, en muchas compañías los usuarios trabajan en PC de escritorio que no están en Internet y que son incapaces de enviar o recibir correo electrónico fuera de la compañía. En cambio, la compañía tiene uno o más servidores de correo electrónico que pueden enviar y recibir correo electrónico. Para enviar o recibir mensajes, una PC debe hablar con un servidor de correo electrónico usando alguna clase de protocolo de entrega.

Un protocolo sencillo usado para traer correo electrónico de un buzón remoto es el POP3 (*Post Office Protocol, protocolo de oficina postal*), que se define en el RFC 1225. El POP3 tiene comandos para que un usuario establezca una sesión, la termine, obtenga mensajes y los borre. El protocolo mismo consiste en texto ASCII y se asemeja un poco al SMTP. El objetivo del POP3 es obtener correo electrónico del buzón remoto y almacenarlo en la máquina local del usuario para su lectura posterior.

Un protocolo más refinado es el IMAP (*Interactive Mail Access Protocol, protocolo interactivo de acceso a correo*), que se define en el RFC 1064. IMAP se diseñó para ayudar al usuario que tiene varias computadoras, tal vez una estación de trabajo en la oficina, una PC en casa y una computadora portátil para el camino. La idea en que se basa el IMAP es que el servidor de correo electrónico mantenga un depósito central al que pueda accederse desde cualquier máquina. Por tanto, a diferencia del POP3, el IMAP no copia el correo electrónico en la máquina personal del usuario porque el usuario puede tener varias.

El IMAP tiene muchas características, como la capacidad de direccionar el correo no por número de arriba, como se hace en la figura 7-40, sino usando atributos (por ejemplo, dame el primer mensaje de Sam). En este enfoque, un buzón se parece más a un sistema de base de datos relacional que a una secuencia lineal de mensajes.

Un tercer protocolo de entrega es el DMSP (*Distributed Mail System Protocol, protocolo de sistema de correo distribuido*), que es parte del sistema PCMAIL y se describe en el RFC 1056. Éste no supone que todo el correo electrónico está en un servidor, como el POP3 y el IMAP. En cambio, permite a los usuarios descargar correo del servidor a una estación de trabajo, PC o portátil y luego desconectarse. El correo electrónico puede leerse y contestarse estando desconectado. Al ocurrir una reconexión después, el correo electrónico se transferirá y el sistema se resincronizará.

Independientemente de si el correo electrónico se entrega directamente a la estación de trabajo del usuario o a un servidor remoto, muchos sistemas proporcionan ganchos para el procesamiento adicional del correo electrónico entrante. Una herramienta especialmente valiosa para muchos usuarios de correo electrónico es la capacidad de establecer filtros. Éstos son reglas que se consultan cuando llega el correo electrónico o cuando se inicia el agente de usuario. Cada regla especifica una condición y una acción. Por ejemplo, una regla podría decir que cualquier mensaje de Andrew S. Tanenbaum debe presentarse en una fuente negrita roja parpadeante de 24 puntos (o, de manera alternativa, descartarse automáticamente sin comentarios).

Otra característica de entrega con que a menudo se cuenta es la capacidad para reenviar (temporalmente) el correo electrónico entrante a una dirección diferente. Esta dirección incluso puede ser una computadora operada por un sistema localizador comercial, que entonces llama al usuario por radio o satélite, presentando la línea *Subject: en su beeper*.

Otra característica común de la entrega final es la capacidad de instalar un *daemon* de vacaciones. Éste es un programa que examina cada mensaje de entrada y envía al transmisor una respuesta insípida como

Hola. Estoy de vacaciones. Regresaré el 24 de agosto. Que tenga un bonito día.

Tales respuestas también pueden especificar la manera de manejar asuntos urgentes en ese interin, indican que otras personas podrían resolver problemas específicos, etc. La mayoría de los *daemons* de vacaciones mantienen un registro de a quiénes se enviaron respuestas prefabricadas y se abstienen de enviar a la misma persona una segunda respuesta. Los mejores también verifican si el mensaje de entrada se envió a una lista de correo, y de ser así, no envían ninguna respuesta prefabricada. (La gente que envía mensajes a listas grandes de correo durante el verano probablemente no quiere recibir cientos de respuestas con los detalles de las vacaciones de todos.)

El autor recientemente se topó con una forma extrema de procesamiento de entregas cuando envió un mensaje de correo electrónico a una persona que asegura recibir 600 mensajes al día. Su identidad no se divulgará, no sea que los lectores de este libro también le envíen correo electrónico. Llamémoslo Juan.

Juan instaló un robot de correo electrónico que revisa cada mensaje entrante para ver si es de un corresponsal nuevo. De serlo, devuelve un mensaje prefabricado explicando que Juan ya no puede leer personalmente todo su correo electrónico. En cambio, ha producido un documento FAQ (de preguntas comunes) que contesta muchas preguntas que se le hacen con frecuencia. Por lo regular, los grupos de noticias tienen documentos FAQ, no gente.

El FAQ de Juan da su dirección, fax y números telefónicos e indica la manera de comunicarse con su compañía; explica la manera de contratarlo como expositor y describe dónde pueden conseguirse sus artículos y otros documentos. El FAQ también proporciona referencias al software que ha escrito, una conferencia que imparte, un estándar que editó, etc. Tal vez este enfoque sea necesario, pero probablemente un FAQ personal es el último símbolo de *estatus*.

7.4.5. Confidencialidad en el correo electrónico

Cuando un mensaje de correo electrónico se envía entre dos sitios distantes, generalmente transitárá por docenas de máquinas en el camino. Cualquiera de éstas puede leer y registrar el mensaje para su uso posterior. La confidencialidad es inexistente, a pesar de lo que piensa mucha gente (Wisband y Reining, 1995). No obstante, mucha gente quisiera poder enviar correo electrónico que el destinatario pueda leer, pero nadie más: ni su jefe, ni los *hackers*, ni siquiera el *gobierno*. Este deseo ha estimulado a varias personas y grupos a aplicar los principios criptográficos que estudiamos antes al correo electrónico para producir correo seguro. En las siguientes secciones estudiaremos dos sistemas de correo electrónico seguros de amplio uso, PGP y PEM. Para información adicional, véase (Kaufman *et al.*, 1995; Schneier, 1995; Stallings, 1995b, y Stallings, 1995c).

PGP — Bastante buena confidencialidad

Nuestro primer ejemplo, PGP (*Pretty Good Privacy*, bastante buena confidencialidad) es en esencia el vástago de una sola persona, Phil Zimmermann (Zimmermann, 1995a, 1995b). PGP es un paquete completo de seguridad de correo electrónico que proporciona confidencialidad, validación de identificación, firmas digitales y compresión, todo de una forma fácil de usar. Es más, el paquete completo, incluido todo el código fuente, se distribuye sin cargo por Internet, boletines electrónicos y redes comerciales. Debido a su calidad, precio (cero) y fácil disponibilidad en las plataformas MS-DOS/Windows, UNIX y Macintosh, es de amplio uso hoy día. También hay una versión comercial disponible para aquellas compañías que requieren apoyo.

PGP también se ha visto envuelto en varias controversias (Levy, 1993). Puesto que está disponible gratuitamente en Internet, el gobierno de Estados Unidos ha alegado que la posibilidad de que lo obtengan extranjeros constituye una violación de las leyes concernientes a la exportación de armamentos. Se produjeron versiones posteriores fuera de Estados Unidos para evitar esta restricción. Otro problema ha tenido que ver con una supuesta violación de la patente del RSA, pero ese problema ya fue resuelto en las versiones 2.6 en adelante. Sin embargo, no a todo el mundo le gusta la idea de que la gente guarde secretos, por lo que los enemigos del PGP siempre están al acecho en las sombras, esperando el momento de atacar. Acorde con la situación, el lema de Zimmermann es: "si la confidencialidad se vuelve ilegal, sólo los ilegales tendrán confidencialidad".

El PGP intencionalmente usa algoritmos criptográficos existentes en lugar de inventar nuevos; se basa en gran medida en el RSA, el IDEA y el MD5, todos algoritmos que han resistido análisis extensos de colegas y no fueron diseñados ni influidos por ninguna agencia gubernamental que estuviera tratando de debilitarlos. Para la gente que tiende a desconfiar del gobierno, esta propiedad es un gran punto a favor.

El PGP maneja compresión de textos, secreto y firmas digitales, y también proporciona amplios recursos de manejo de claves. Para ver cómo funciona el PGP, consideremos el ejemplo de la figura 7-49. Aquí, Alicia quiere enviar un mensaje de texto normal firmado, P , a Benjamín de una manera segura. Tanto Alicia como Benjamín tienen claves RSA privadas (D_A) y públicas (E_B). Supongamos que cada uno conoce la clave pública del otro; cubriremos después la administración de claves.

Alicia comienza por invocar el programa PGP en su computadora. El PGP primero dispersa su mensaje, P , usando MD5, y luego cifra la dispersión resultante usando su clave RSA privada, D_A . Cuando Benjamín recibe el mensaje, puede descifrar la dispersión con la clave pública de Alicia y comprobar que es correcta. Incluso si alguien más (por ejemplo, Trudy) pudiera adquirir la dispersión en esta etapa y descifrarla con la clave pública de Alicia, la robustez del MD5 asegura que sería infactible computacionalmente producir otro mensaje con la misma dispersión MD5.

La dispersión cifrada y el mensaje original ahora están concatenados en un solo mensaje, $P1$, y comprimidos mediante el programa ZIP, que usa el algoritmo Ziv-Lempel (Ziv y Lempel, 1977). Llamemos a la salida de este paso $P1.Z$.

K_M : Clave de mensaje de una sola vez para IDEA

\otimes : Concatenación

Clave RSA privada de Alicia, D_A

Clave RSA pública de Benjamín, E_B

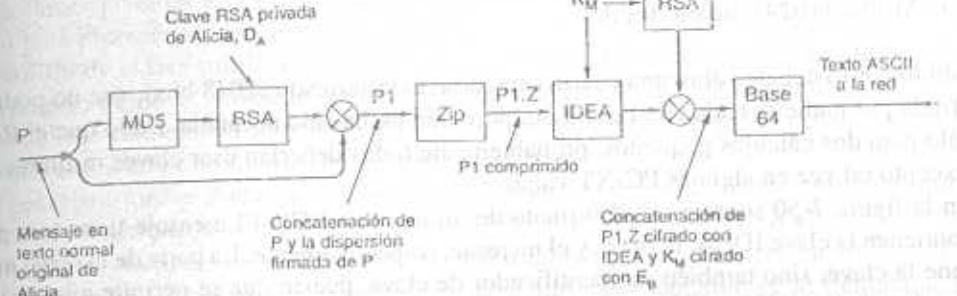


Figura 7-49. PGP en operación para el envío de un mensaje.

A continuación, el PGP solicita a Alicia una entrada al azar. Tanto el contenido como la velocidad de tecleo se usan para generar una clave de mensaje IDEA de 128 bits, K_M (llamada clave de sesión en la literatura PGP, pero que en realidad es un nombre inapropiado, pues no hay una sesión). K_M se usa ahora para cifrar $P1.Z$ con IDEA en modo de realimentación de cifrado. Además, K_M se cifra con la clave pública de Benjamín, E_B . Estos dos componentes se concatenan y convierten a base64, como estudiamos en la sección sobre MIME. El mensaje resultante contiene entonces sólo letras, dígitos y los símbolos +, / e =, lo que significa que puede ponerse en un cuerpo RFC 822 y esperar que llegue sin modificación.

Cuando Benjamín recibe el mensaje, invierte la codificación base64 y descifra la clave IDEA usando su clave RSA privada. Con la clave así obtenida, Benjamín descifra el mensaje para obtener $P1.Z$. Tras descomprimir esto, Benjamín separa el texto normal de la dispersión cifrada y descifra la dispersión usando la clave pública de Alicia. Si la dispersión del texto normal concuerda con su propio cálculo MD5, sabe que P es el mensaje correcto y que vino de Alicia.

Vale la pena señalar que el RSA sólo se usa aquí en dos lugares: para cifrar la parcialización MD5 de 128 bits y para cifrar la clave IDEA de 128 bits. Aunque el RSA es lento, sólo tiene que cifrar 256 bits, lo cual no es un gran volumen de datos. Es más, los 256 bits de texto normal son excesivamente aleatorios, por lo que se requerirá una cantidad considerable de trabajo por parte de Trudy para determinar si una clave adivinada es la correcta. El cifrado de trabajo pesado lo hace IDEA, que es más rápido que el RSA en varios órdenes de magnitud. Por tanto, el PGP proporciona seguridad, compresión y firma digital, y lo hace de una manera mucho más eficiente que el esquema ilustrado en la figura 7-23.

El PGP maneja tres longitudes de clave RSA. Es responsabilidad del usuario seleccionar la más adecuada. Las longitudes son:

1. Casual (384 bits); puede ser descifrada hoy día por gente con grandes presupuestos.
2. Comercial (512 bits); podría ser descifrable por organizaciones de tres iniciales.
3. Militar (1024); indescifrable.

Ha habido cierto debate sobre una cuarta categoría: extraterrestre (2048 bits), que no podría ser descifrada por nadie ni nada en el universo, pero aún no ha sido aceptada. Dado que el RSA se usa sólo para dos cálculos pequeños, probablemente todos deberían usar claves militares siempre, excepto tal vez en algunas PC-XT viejas.

En la figura 7-50 se muestra el formato de un mensaje PGP. El mensaje tiene tres partes, que contienen la clave IDEA, la firma y el mensaje, respectivamente. La parte de la clave no sólo contiene la clave, sino también un identificador de clave, puesto que se permite a los usuarios tener varias claves públicas.



Figura 7-50. Mensaje PGP.

La parte de firma contiene una cabecera, de la que no nos ocuparemos aquí. A la cabecera le sigue una marca de tiempo, el identificador de la clave pública del transmisor, que puede servir para descifrar la dispersión de la firma, cierta información de tipo que identifica los algoritmos usados (para permitir el uso del MD5 y el RSA2 cuando se inventen) y la dispersión cifrada misma.

La parte del mensaje también contiene una clave, el nombre predeterminado del archivo a usar si el receptor escribe el archivo en disco, una marca de tiempo de creación del mensaje y, por último, el mensaje mismo.

La administración de claves ha recibido mucha atención en el PGP, puesto que es el talón de Aquiles de todos los sistemas de seguridad. Cada usuario mantiene localmente dos estructuras de datos: un anillo de claves privadas y un anillo de claves públicas. El anillo de claves privadas contiene uno o más pares de clave privada-clave pública personales. La razón para reconocer varios pares por usuario es permitir a los usuarios cambiar sus claves públicas periódicamente o cuando se piensa que una está comprometida, sin invalidar los mensajes que actualmente están en preparación o en tránsito. Cada par asociado tiene un identificador, para

que el remitente de un mensaje pueda indicar al destinatario la clave pública usada para cifrarlo. Los identificadores de mensaje consisten en los 64 bits de orden menor de la clave pública. Los usuarios son responsables de evitar conflictos con sus identificadores de clave pública. Las claves privadas en el disco se cifran usando una contraseña especial (arbitriariamente grande) para protegerlas contra ataques furtivos.

El anillo de claves públicas contiene claves públicas de los correspondientes del usuario; se requieren para cifrar las claves de mensaje asociadas a cada mensaje. Cada entrada del anillo de claves públicas contiene no sólo la clave pública, sino también su identificador de 64 bits y una indicación del nivel de confianza del usuario en la clave.

El problema que se ataca aquí es el siguiente. Supóngase que se mantienen claves públicas en boletines electrónicos. Una manera como Trudy pueda leer el correo electrónico secreto de Benjamín es atacar el tablero de avisos y reemplazar la clave pública de Benjamín por una de su propia selección. Cuando Alicia obtiene la clave que supuestamente es de Benjamín, Trudy puede montar un ataque de brigada de cubetas contra Benjamín.

Para evitar tales ataques, o cuando menos minimizar sus consecuencias, Alicia necesita saber qué tanto puede confiar en el elemento llamado "clave de Benjamín" de su anillo de claves públicas. Si sabe que Benjamín personalmente le entregó un disquete con la clave, puede establecer el valor de confianza en el nivel más alto.

Sin embargo, en la práctica, la gente con frecuencia recibe claves públicas haciendo una consulta a un servidor de claves confiable, de los cuales hay bastantes en operación en Internet. Al recibir un servidor de claves la solicitud de la clave pública de alguien, genera una respuesta que contiene la clave pública, una marca de tiempo, y la fecha de expiración de la clave; luego parcializa esta respuesta con MD5 y firma la respuesta con su propia clave privada de modo que la parte solicitante pueda verificar quién la envió. Es responsabilidad del usuario asignar un nivel de confianza a las claves mantenidas por el administrador del sistema local, la compañía telefónica, la ACM, la asociación de abogados, el gobierno, o cualquier otro que decida entrar en el negocio de administrar claves.

PEM — Correo con confidencialidad mejorada

En contraste con el PGP, que inicialmente fue asunto de una sola persona, nuestro segundo ejemplo, el PEM (*Privacy Enhanced Mail, correo con confidencialidad mejorada*), es un estándar oficial de Internet y se describe en cuatro RFC: el RFC 1421 al RFC 1424. De manera muy general, el PEM cubre el mismo territorio que el PGP: confidencialidad y validación de identificación para sistemas de correo electrónico basados en el RFC 822. Sin embargo, también tiene algunas diferencias respecto al PGP en cuanto a enfoque y tecnología. A continuación describiremos el PEM y luego lo compararemos con el PGP. Para mayor información sobre PEM, véase (Kent, 1993).

Los mensajes enviados usando PEM primero se convierten a una forma canónica para que puedan tener las mismas convenciones de espacios blancos (por ejemplo, tabuladores, espacios al final) y el uso de retornos de carro y avances de línea. Esta transformación se lleva a cabo para eliminar los efectos de los agentes de transferencia de mensajes que modifican los mensajes que

no son de su gusto. Sin canonización, tales modificaciones pueden afectar las dispersiones de los mensajes en sus destinos.

A continuación, se calcula una dispersión del mensaje usando MD2 o MD5. No es opcional, como en PGP. Después se cifra la concatenación de la dispersión y el mensaje usando DES. En vista de las debilidades conocidas de una clave de 56 bits, esta selección ciertamente es sospechosa. El mensaje cifrado puede entonces codificarse con codificación base64 y transmitirse al destinatario. Las listas de correo se reconocen explícitamente.

Como con el PGP, cada mensaje se cifra con una clave de una sola vez que se incorpora en el mensaje. La clave puede protegerse mediante RSA o con DES triple usando EDE. En la práctica, todos usan RSA, por lo que nos concentraremos en él. De hecho, tenemos que hacerlo: el PEM no indica cómo se debe hacer la administración de claves con DES.

La administración de claves es más estructurada que en el PGP. Las claves se certifican mediante autoridades certificadoras que producen certificados indicando el nombre del usuario, su clave pública y la fecha de expiración de la clave. Cada certificado tiene un número de serie único que lo identifica. Los certificados incluyen una dispersión MD5 firmada por la clave privada de la autoridad certificadora. Estos certificados se ajustan a la recomendación ITU X.509 para certificados de claves públicas y, como tales, usan nombres X.400 como el ejemplo de Tom Smith que presentamos antes.

El PGP tiene un esquema parecido (sin el uso del X.509), pero tiene un problema: ¿debe creer un usuario en la autoridad certificadora? El PEM resuelve este problema certificando las autoridades certificadoras, usando lo que se llama PCA (*Policy Certification Authorities, autoridades de certificación de políticas*). Éstas, por su parte, son certificadas por la IPRA (*Internet Policy Registration Authority, autoridad de registro de políticas de Internet*), que es el máximo árbitro sobre quién es bueno y quién es malo.

Cada PCA debe definir una política oficial en el momento de registrarse y archivarla con la IPRA. Estas declaraciones deben ser firmadas por el IPRA y hacerse públicas. Por ejemplo, una PCA puede insistir en que los usuarios bajo su jurisdicción se presenten personalmente con un certificado de nacimiento, licencia de conducir, pasaporte, dos tarjetas de crédito bancarias, un testigo vivo y una clave pública en disquete. Otra PCA podría aceptar registros por correo electrónico de completos extraños. Al hacer pública las declaraciones de políticas, los usuarios tienen una base para decidir en qué autoridades pueden confiar. No se ha tomado ninguna medida para asegurar que las políticas realmente se pongan en práctica.

Se contemplan tres tipos diferentes de autoridades certificadoras. Una autoridad organizacional puede emitir certificados para sus empleados. La mayoría de las compañías operarán su propia autoridad. Una residencial operará en nombre de ciudadanos comunes, de manera parecida a los proveedores de servicios de Internet, que proporcionan servicio a cualquiera que esté dispuesto a pagarlos. Por último, hay un esquema planeado para el registro anónimo. Con todas estas autoridades certificadoras por ahí, debe quedar clara la necesidad de que las PCA las vigilen de cerca.

Aunque es rígidamente jerárquico y burocrático, este esquema tiene la ventaja respecto al PGP de hacer potencialmente práctica la revocación de certificados. Se requiere la revocación de certificados si un usuario quiere cambiar su clave pública, por ejemplo, porque está comprometida o su autoridad certificadora ha sido objeto de robo. La revocación se logra cuando un

usuario dice a la autoridad certificadora que su clave pública está comprometida (o posiblemente al revés). La autoridad certificadora entonces agrega el número de serie del certificado que ha dejado de ser válido a una lista de certificados revocados, lo firma y distribuye la lista a todo mundo.

Cualquier que desee enviar un mensaje PEM a un usuario debe, por tanto, revisar la lista más reciente de revocaciones para ver si la clave pública en caché aún es válida. Este proceso es análogo a que un comerciante revise la lista de tarjetas de crédito robadas antes de aceptar una. Los críticos del PEM argumentan que estar revisando todo el tiempo es demasiado trabajo, por lo que nadie se tomará la molestia. Los que están a su favor argumentan que las computadoras no se aburren; si se programan para revisar todo el tiempo, lo harán.

Algunas de las semejanzas y diferencias entre el PGP y el PEM se listan en la figura 7-51. La mayoría de estos puntos ya han sido cubiertos, pero vale la pena comentar algunos. La validación de identificación parece más importante en el PEM que en el PGP, puesto que es obligatoria en el PEM y opcional en el PGP. También, el PEM lleva la información de validación de identificación fuera de la envoltura de cifrado, lo que significa que la red puede comprobar el origen de cada mensaje. Como consecuencia, los espías pueden registrar quién envía a quién, aunque no pueden leer los mensajes.

Dejando a un lado todas estas diferencias técnicas, hay también una diferencia cultural sorprendente. El PGP, que no es un estándar oficial de Internet, tiene la cultura Internet. El PEM, que es un estándar oficial de Internet, no la tiene. El PGP se basó en lo que Dave Clark llama "consenso aproximado y código operativo". Alguien (Zimmermann) pensó en una solución a un problema muy conocido, la implementó bien y liberó el código fuente para el uso de todos. El PEM comenzó como un estándar oficial de cuatro partes, usando ASN.1 para definir el formato, X.400 para definir los nombres y X.509 para definir los certificados; usa una jerarquía organizacional rígida de tres niveles con varios tipos de autoridades certificadoras, incluyendo declaraciones de políticas certificadas oficialmente y el requisito de que todo mundo confíe en la IPRA. Las implementaciones vinieron después y están muy por debajo del PGP en cuanto a calidad, cantidad y disponibilidad en muchas plataformas. En pocas palabras, el PGP parece un paquete Internet típico, mientras que el PEM presenta la mayoría de las características de un estándar OSI que la gente de Internet odia y los PTT aman. Trate de entender esto.

7.5. NOTICIAS USENET

Una de las aplicaciones más comunes de las redes de cómputo es el sistema mundial de grupos de noticias llamado *noticias de red (net news)*. Con frecuencia, esto se conoce como USENET, que se remonta a una red física de UNIX a UNIX independiente que alguna vez condujo el tráfico usando un programa llamado uucp. Hoy día, una buena parte del tráfico se lleva por la Internet, pero USENET e Internet no son lo mismo. Algunas instalaciones Internet no reciben las noticias de red y otras reciben noticias sin estar en Internet.

En las siguientes secciones describiremos el USENET. Primero lo veremos desde el punto de vista del usuario. Luego describiremos su implementación.

Asunto	PGP	PEM
¿Maneja cifrado?	Sí	Sí
¿Maneja validación de identificación?	Sí	Sí
¿Maneja no repudio?	Sí	Sí
¿Maneja compresión?	Sí	No
¿Maneja canonización?	No	Sí
¿Maneja listas de correo?	No	Sí
¿Usa codificación base64?	Sí	Sí
Algoritmo actual de cifrado de datos	IDEA	DES
Longitud de clave para cifrado de datos (bits)	128	56
Algoritmo actual para administración de claves	RSA	RSA o DES
Longitud de clave para administración de claves	384/512/1024	Variable
Espacio de nombres de usuario	Definido por usuario	X.400
¿Se ajusta al X.509?	No	Sí
¿Hay que confiar en alguien?	No	Sí (IPRA)
Certificación de claves	Ad hoc	Jerarquía IPRA/PCA/CA
Revocación de claves	Fortuita	Mejor
¿Pueden los espías leer los mensajes?	No	No
¿Pueden los espías leer las firmas?	No	Sí
¿Estándar Internet?	No	Sí
Diseñado por	Equipo pequeño	Comité de estándares

Figura 7-51. Comparación del PGP y el PEM.

La cantidad de grupos de noticias es tan grande (probablemente más de 10,000) que se organizan jerárquicamente para poderlos manejar. En la figura 7-52 se muestran los niveles superiores de las jerarquías "oficiales". También existen otras jerarquías, pero por lo regular están dirigidas a consumo regional o están en idiomas diferentes del inglés. Una de las otras jerarquías, *alt*, es especial. *Alt* es para los grupos oficiales lo que un mercado callejero es para una tienda departamental; es una mezcolanza caótica y desregulada de grupos de noticias sobre todos los temas, algunos de los cuales son muy populares, y la mayoría de los cuales son mundiales.

Nombre	Temas
Comp	Computadoras, informática y la industria de la computación
Sci	Ciencias físicas e ingeniería
Humanities	Literatura y humanidades
News	Debates sobre el USENET mismo
Rec	Actividades recreativas, incluye deportes y música
Mic	Todo lo que no cabe en otro lado
Soc	Relaciones y asuntos sociales
Talk	Diatribas, polémicas, debates y argumentos a gran escala
Alt	Árbol alterno que cubre prácticamente todo

Figura 7-52. Jerarquías USENET en orden descendente de relación señal a ruido.

Los grupos *comp* fueron los grupos USENET originales. Estos grupos están poblados de informáticos, profesionales de la computación y aficionados a las computadoras. Cada uno ofrece debates técnicos sobre un tema relacionado con el *hardware* o *software* de cómputo.

Los grupos *sci* y *humanities* están poblados de científicos, estudiantes y aficionados interesados en física, química, biología, Shakespeare, etc. No es sorprendente que la jerarquía *sci* es mucho mayor que *humanities*, pues el concepto mismo de la comunicación electrónica instantánea con colegas de todo el mundo es algo que a la mayoría de los científicos le gusta, pero sobre lo que la mayoría de los humanistas cuando menos tiene reservas. C.P. Snow tenía razón.

La jerarquía *news* se usa para debatir y administrar el sistema de noticias mismo. Los administradores de sistemas pueden encontrar ayuda aquí, y aquí tienen lugar los debates sobre la creación de grupos de noticias nuevos.

Las jerarquías cubiertas hasta el momento tienen un tono profesional, un tanto académico. Eso cambia con *rec*, que trata sobre actividades recreativas y aficiones. No obstante, las personas que publican aquí tienen conocimientos amplios sobre sus intereses respectivos.

A medida que seguimos avanzando, llegamos a *soc*, que tiene muchos grupos de noticias relacionados con política, sexo, religión, diferentes culturas nacionales y genealogía. *Talk* cubre

7.5.1. USENET desde el punto de vista del usuario

Un grupo de noticias es un foro de debate mundial sobre un tema específico. La gente interesada en el tema puede "suscribirse" al grupo de noticias. Los suscriptores pueden usar un tipo especial de agente de usuario, un lector de noticias, para leer todos los artículos (mensajes) publicados en el grupo de noticias. La gente también puede publicar artículos en el grupo de noticias. Cada artículo publicado en un grupo de noticias automáticamente se entrega a todos los suscriptores, en cualquier lugar del mundo donde puedan estar. La entrega típicamente tarda entre unos cuantos segundos y varias horas, dependiendo de lo alejados que estén el transmisor y el receptor de las rutas comunes. En efecto, un grupo de noticias es algo como una lista de correo, pero internamente se implementa de manera diferente. También puede visualizarse como un tipo de multitransmisión de alto nivel.

temas controvertidos y está poblado de gente con opiniones fuertes, pero débil en cuanto a información y cifras. *Alt* es un árbol alterno completo que opera bajo sus propias reglas.

Cada una de las categorías listadas en la figura 7-52 se divide en subcategorías, recursivamente. Por ejemplo, *rec.sport* se ocupa de deportes, *rec.sport.basketball* es sobre baloncesto y *rec.sport.basketball.women* trata el baloncesto femenil. En la figura 7-53 se presenta una muestra de algunos de los grupos de noticias de cada categoría. En muchos casos, la existencia de grupos adicionales puede inferirse cambiando los parámetros obvios. Por ejemplo, *comp.lang.c* es sobre el lenguaje de programación C, pero la *.c* puede cambiarse por prácticamente cualquier otro lenguaje de programación para generar el nombre del grupo de noticias correspondiente.

Existen muchos programas lectores de noticias. Al igual que los lectores de correo electrónico, algunos se basan en el teclado; otros se basan en el ratón. En casi todos los casos, cuando el lector de noticias inicia, revisa un archivo para ver los grupos de noticias a los que está suscrito el usuario; después, por lo regular, presenta un resumen de una línea de cada artículo no leído aún del primer grupo de noticias y espera que el usuario seleccione uno o más. Los artículos seleccionados se presentan en la pantalla, uno a la vez. Tras su lectura, los artículos pueden descartarse, guardarse, imprimirse, etcétera.

Los lectores de noticias también permiten a los usuarios suscribirse y cancelar las suscripciones a los grupos de noticias. El cambio de una suscripción simplemente significa la edición del archivo local que lista los grupos de noticias a los que está suscrito el usuario. Para hacer una analogía, la suscripción a un grupo de noticias es como ver un programa de televisión. Si usted quiere ver un programa cada semana, simplemente lo hace. No necesita registrarse antes con una autoridad central.

Los lectores de noticias también manejan la publicación. El usuario compone un artículo y luego emite un comando o hace clic en un ícono para enviar el artículo. En un lapso de un día, llegarán a casi todos los suscriptores mundiales del grupo de noticias en el que se publicó. Es posible hacer **publicaciones cruzadas** (*crossposts*) de un artículo, es decir, enviarlo a varios grupos de noticias con un solo comando. También es posible restringir la distribución geográfica de una publicación. Un anuncio del coloquio del martes en Stanford probablemente no será de mucho interés, digamos, en Hong Kong, por lo que la publicación puede restringirse a California.

La sociología de USENET es única, por decirlo de alguna manera. Nunca antes ha sido posible que miles de personas que no se conocen tengan debates mundiales sobre una enorme variedad de temas. Por ejemplo, es posible ahora que alguien con un problema lo publique en la red. Al día siguiente, el autor de la publicación puede tener 18 soluciones y, con un poco de suerte, sólo 17 de ellas equivocadas.

Desafortunadamente, algunas personas usan su nuevo poder de comunicación para comunicarse irresponsablemente con un grupo grande. Cuando alguien publica un mensaje que indica: "A la gente como ustedes deberían fusilarla", los ánimos se caldean y sigue un torrente de publicaciones injuriosas, llamado **guerra de llamas** (*flamewar*).

Esta situación puede atacarse de dos maneras, una individual y otra colectiva. Los usuarios individuales pueden instalar un archivo de eliminación (*killfile*), que especifica que los artículos con cierto tema o de cierta persona deben descartarse a su llegada, antes de exhibirse.

SEC. 7.5

NOTICIAS USENET

673

Nombre	Temas
Comp.ai	Inteligencia artificial
Comp.databases	Diseño e implementación de sistemas de bases de datos
Comp.lang.c	El lenguaje de programación C
Comp.os.minix	Sistema operativo educativo MINIX de Tanenbaum
Comp.os.ms-windows.video	Hardware y software de video para Windows
Sci.bio.entomology.lepidoptera	Investigación sobre mariposas y polillas
Sci.geo.earthquakes	Geología, sismología y terremotos
Sci.med.orthopedics	Cirugía ortopédica
Humanities.lit.authors.shakespeare	Dramas y poesía de Shakespeare
News.groups	Potenciales grupos de noticias nuevos
News.lists	Listas relacionadas con el USENET
Rec.arts.poems	Poesía gratuita
Rec.food.chocolate	Sabroso
Rec.humor.funny	¿Has oido el chiste sobre el granjero que...?
Rec.music.folk	Discusiones sobre música folclórica
Misc.jobs.offered	Anuncios de plazas disponibles
Misc.health.diabetes	La vida día a día con diabetes
Soc.culture.estonia	Vida y cultura de Estonia
Soc.singles	Solteros y sus intereses
Soc.couples	Parejas. Graduados de soc.singles
Talk.abortion	Aborto. Sin señal, puro ruido
Talk.rumors	De aquí salen los rumores
Alt.alien.visitors	Lugar para informar de viajes en OVNI
Alt.bermuda.triangle	Si lee esto, desaparece misteriosamente
Alt.sex.voyeurism	Asómate y véalo usted mismo
Alt.tv.simpsons	Bart y compañía

Figura 7-53. Pequeña selección de los grupos de noticias.

La mayoría de los lectores de noticias también permite matar el hilo de una discusión individual. Esta característica es útil cuando parece que una discusión comienza a entrar en un ciclo infinito.

Si se molestan suficientes suscriptores de un grupo con la contaminación del grupo de noticias, pueden proponer la moderación del grupo de noticias. Un grupo de noticias moderado es uno en el que sólo una persona, el moderador, puede publicar artículos en el grupo de noticias. Todas las publicaciones de un grupo moderado se envían automáticamente al moderador, quien publica las buenas y descarta las malas. Algunos temas tienen un grupo de noticias moderado y uno no moderado.

Dado que cada día se suscriben miles de personas a USENET por primera vez, tienden a hacerse las mismas preguntas de principiante una y otra vez. Para reducir este tráfico, muchos grupos de noticias han construido un documento FAQ (*Frequently Asked Questions*, preguntas

frecuentes) que intenta contestar todas las preguntas de los principiantes. Algunos de éstos son bastante exhaustivos y abarcan más de 100 páginas. El encargado del mantenimiento típicamente los publica una o dos veces al mes.

El USENET está lleno de jerga como BTW (*By The Way*, por cierto), ROFL (*Rolling On the Floor and Laughing*, tirado en el piso y muerto de risa) e IMHO (*In My Humble Opinion*, desde mi humilde punto de vista). Mucha gente también usa pequeños símbolos ASCII llamados caritas (*smileys*) o iconos de emoción (*emoticons*). Algunos de los más interesantes se reproducen en la figura 7-54. En casi todos los casos, el girar el libro 90 grados en dirección de las manecillas del reloj hará más claros los símbolos. (Sanderson y Dougherty, 1993) tienen un minilibro con más de 650 caritas.

Carita	Significado	Carita	Significado	Carita	Significado
: -)	Estoy contento	= : -)	Abraham Lincoln	. +)	Nariz grande
: - (Estoy triste/enojado	* : -)	Tío Sam	-)	Papada
: -	No me importa	* <: -)	Santa Claus	- (Bigote
: -)	Estoy guiñando	<: - (Tonto	# -)	Cabello enmarañado
: - (0)	Estoy gritando	-	Australiano	8 -)	Usa lentes
: - (*)	Estoy vomitando	- X	Hombre con corbata de moño	C -)	Cerebro grande

Figura 7-54. Algunas caritas.

Aunque la mayoría de la gente usa su verdadero nombre en las publicaciones, hay quienes desean permanecer anónimos, especialmente cuando publican en grupos de noticias sujetos de controversia y cuando publican anuncios personales en los grupos de noticias relacionados con la búsqueda de un compañero(a). Este deseo ha conducido a la creación de **reenviadores anónimos**, que son servidores que aceptan mensajes de correo electrónico (incluidas publicaciones) y cambian los campos *From:*, *Sender:* y *Reply-To:* para hacer que apunten al reenviador en lugar de al remitente. Algunos de los reenviadores asignan números a los usuarios y reenvían el correo electrónico dirigiéndolo a estos números, para que la gente pueda enviar respuestas de correo electrónico a las publicaciones anónimas como "Mujer blanca soltera 25 busca hombre blanco soltero/divorciado 20-30..." Es dudoso que estos reenviadores puedan guardar sus secretos cuando la policía local sienta curiosidad por la identidad de algún usuario (Barlow, 1995).

A medida que más gente se suscribe al USENET, hay una demanda constante de grupos de noticias nuevos más especializados. En consecuencia, se ha establecido un procedimiento para crear grupos nuevos. Supongamos que a alguien le gustan las cucarachas y quiere hablar con otros fanáticos de las cucarachas; publica un mensaje en *news.groups* nombrando el grupo propuesto, digamos *rec.animals.wildlife.cockroaches* y describiendo la razón por la que son tan importantes (las cucarachas son fascinantes; hay 3500 especies; vienen en rojo, amarillo, verde, café y negro; aparecieron en la tierra mucho antes que los primeros dinosaurios; probablemente fueron los primeros animales voladores, etc.). También se indica si debe ser un grupo moderado o no.

Surge entonces el debate. Cuando termina, se hace una votación por correo electrónico. Los votos se publican, identificando quién votó de qué manera (para evitar fraudes). Si la relación entre los votos sí y los no es mayor que 2:1 y hay cuando menos 100 más a favor que en contra, el moderador de *news.groups* publica un mensaje aceptando el nuevo grupo de noticias. Este mensaje es el aviso a los administradores de sistemas de todo el mundo de que el nuevo grupo ha sido bendecido por los altos poderes y que ahora es oficial.

La creación de un grupo nuevo es menos formal en la jerarquía *alt*, y ésa es, de hecho, la razón de ser de *alt*. Algunos de los grupos de noticias que hay ahí están tan cercanos a los límites legales y morales de lo tolerable que nunca habrían sido aceptados mediante un voto público. En efecto, la gente que los apoyó simplemente pasó de largo el procedimiento normal y creó su propia jerarquía. No obstante, una buena parte de la jerarquía *alt* es bastante convencional.

7.5.2. Implementación de USENET

Algunos de los grupos de noticias más pequeños se implementan como listas de correo. Si alguien quiere publicar un artículo en una de tales listas, lo envía a la dirección de la lista de correo, lo que causa el envío de copias a todas las direcciones de la lista de correo.

Sin embargo, si la mitad de los estudiantes de una universidad grande se suscribiera a *alt.sex*, los servidores de ahí se vendrían abajo por el peso del correo entrante. En consecuencia, USENET generalmente no se implementa usando listas de correo. En cambio, cada instalación (universidad, compañía o proveedor de servicio Internet) almacena el correo entrante en un solo directorio, digamos *news*, con subdirectorios para *comp*, *sci*, etc. Éstos a su vez tienen subdirectorios como *news/comp/os/minix*. Todas las noticias entrantes se depositan en el directorio apropiado. Los lectores de noticias simplemente traen los artículos de ahí según los necesitan. Este arreglo significa que cada instalación sólo requiere una copia de cada artículo, sin importar la cantidad de gente suscrita a su grupo de noticias. Tras unos cuantos días, los artículos expiran y se remueven del disco.

Para entrar en USENET, una instalación debe tener una alimentación de noticias (*newsfeed*) de otra instalación que ya está en USENET. Puede pensarse en el grupo de todas las instalaciones que reciben noticias de red como los nodos de un grafo dirigido. Las líneas de transmisión que conectan pares de nodos forman los arcos del grafo. Este grafo es el USENET. Nótese que estar en Internet no es necesario ni suficiente para estar en USENET.

Periódicamente, cada instalación que desea noticias puede sondear sus alimentadores de noticias, solicitando las noticias nuevas respecto al contacto previo. De haberlas, se recolectan esas noticias y se almacenan en el subdirectorio adecuado de *news*. De esta manera, las noticias se difunden por la red. También es igualmente posible que la alimentación de noticias, en lugar del receptor, tome la iniciativa y haga contacto cuando hay suficientes noticias nuevas. Inicialmente, la mayoría de las instalaciones sondeaban sus alimentadores de noticias, pero ahora es principalmente al revés.

No todas las instalaciones reciben todos los grupos de noticias. Hay varias razones para esto. Primero, la alimentación total de noticias excede los 500 MB diarios y crece con rapidez. El almacenamiento de todo requeriría una gran cantidad de espacio en disco. Segundo, el tiempo y

el costo de transmisión son asuntos importantes. A 28.8 kbps, se requieren 39 horas y una línea telefónica dedicada para transmitir 24 horas de noticias. Incluso a 56 kbps, conseguir todo requiere una línea dedicada durante casi 20 horas al día. De hecho, el volumen total ahora es tan grande que se han creado alimentaciones de noticias vía satélite.

Tercero, no todas las instalaciones se interesan en todos los temas. Por ejemplo, es poco probable que mucha gente de compañías finlandesas quiera leer *rec.arts.manga* (sobre tiras cómicas japonesas). Por último, algunos grupos de noticias son demasiado subidos de color para los gustos de muchos administradores de sistemas, que los proscriben, a pesar de existir un interés local considerable. En diciembre de 1995, la red mundial CompuServe dejó de incluir (temporalmente) todos los grupos de noticias con "sex" en el nombre, porque algún burócrata alemán pensó que sería una buena manera de combatir la pornografía. La protesta resultante fue predecible, instantánea, mundial y muy ruidosa.

Los artículos de noticias tienen el mismo formato que los mensajes de correo electrónico RFC 822, pero con la adición de algunas cabeceras extra. Esta propiedad los hace fáciles de transportar y compatibles con casi todo el software de correo electrónico existente. Las cabeceras de noticias se definen en el RFC 1036. En la figura 7-55 se presenta un artículo de ejemplo.

```
From: Vogel@nyu.edu
Message-ID: <54731@nyu.edu>
Subject: Avistamiento de pájaro
Path: cs.vu.nl!sun4nl!EU.net!news.sprintlink.net!in2.uu.net!pc144.nyu.edu!news
Newsgroups: rec.birds
Followup-To: rec.birds
Distribution: world
Nntp-Posting-Host: nuthatch.bio.nyu.edu
References:
Organization: New York University
Lines: 4
Summary: Adivinen lo que vi

Acabo de ver un aveSTRUZ en la Calle 52 y la Quinta Avenida de Nueva York. ¿Es su época
de migración? ¿Lo vio alguien más?

Jay Vogel
```

Figura 7-55. Ejemplo de artículo noticioso

Son pertinentes algunas palabras sobre las cabeceras de noticias. La cabecera *Path*: es la lista de nodos que atravesó el mensaje para llegar del remitente al destinatario. En cada salto, la máquina reenviadora puso su nombre al frente de la lista. Esta lista proporciona una trayectoria de regreso al remitente. El uso de signos de admiración (llamados *bang*) se remonta a las direcciones USENET, que antedatan al DNS.

La cabecera *Newsgroups*: indica los grupos a los que pertenece el mensaje. Puede contener más de un nombre de grupo de noticias. Cualquier mensaje publicado en forma cruzada en

varios grupos de noticias contendrá todos sus nombres. Dado que se permiten nombres múltiples aquí, se requiere la cabecera *Followup-To*: para indicarle a la gente dónde puede publicar los comentarios y reacciones, a fin de poner la discusión subsecuente en un solo grupo de noticias.

La cabecera *Distribution*: indica la distancia a la que se distribuirá la publicación; puede contener uno o más códigos de estado o país, el nombre de una instalación o red específica, o "world" (todo el mundo).

La cabecera *Nntp-Posting-Host*: es análoga a la cabecera *Sender* del RFC 822. Indica la máquina que publicó el artículo, aun si fue compuesto en otra máquina (NNTP es el protocolo de noticias, que se describe más adelante).

La cabecera *References*: indica que este artículo es una respuesta a un artículo anterior e identifica a ese artículo. Se requiere en todos los artículos que son una continuación y se prohíbe cuando se comienza una discusión nueva.

La cabecera *Organization*: puede usarse para indicar la compañía, universidad o agencia a la que está afiliada el remitente. Los artículos que llenan esta cabecera con frecuencia tienen una nota al final para indicar que, en caso de ser una tontería, no es culpa de la organización.

La cabecera *Lines*: especifica la longitud del cuerpo. Las líneas de cabecera y la línea en blanco que separa la cabecera del cuerpo no se toman en cuenta.

Las líneas *Subject*: atan los hilos de la discusión. Muchos lectores de noticias tienen un comando para permitir que el usuario vea el siguiente artículo sobre el tema actual, en lugar del siguiente artículo que llegó. También los archivos de eliminación y los comandos de eliminación usan esta cabecera para saber lo que deben rechazar.

Por último, el *Summary*: normalmente se usa para resumir un artículo de continuación. En los artículos de continuación, la cabecera *Subject*: contiene "Re:" seguido del tema original.

NNTP — Protocolo de transferencia de noticias de red

Veamos ahora la manera en que se difunden los artículos por la red. El algoritmo inicial simplemente inundaba los artículos por cada línea de USENET. Aunque esto funcionó durante un tiempo, tarde o temprano el volumen de tráfico hizo impráctico este esquema, por lo que tuvo que diseñarse algo mejor.

Su reemplazo fue un protocolo llamado NNTP (*Network News Transfer Protocol*, protocolo de transferencia de noticias de red), que se define en el RFC 977. El NNTP recuerda un poco al SMTP; un cliente emite comandos en ASCII y un servidor emite respuestas en forma de números decimales codificados en ASCII. La mayoría de las máquinas USENET usan ahora el NNTP.

El NNTP se diseñó para dos fines. La primera meta era permitir la propagación de artículos de una máquina a otra a través de una conexión confiable (por ejemplo, TCP). La segunda meta era permitir la lectura remota de noticias a los usuarios cuyas computadoras no pueden recibir noticias. El NNTP se usa ampliamente para ambas cosas, pero nos concentraremos en la manera en que los artículos de noticias se difunden por la red usando NNTP.

Como se mencionó antes, son posibles dos enfoques. En el primero, obtención de noticias, el cliente llama a uno de sus alimentadores de noticias y solicita noticias nuevas. En el segundo,

empuje de noticias, el alimentador de noticias llama al cliente y anuncia que tiene noticias. Los comandos NNTP reconocen ambos enfoques, al igual que la lectura remota de noticias.

Para obtener artículos recientes, un cliente debe primero establecer una conexión TCP con el puerto 119 de uno de sus alimentadores de noticias. Detrás de este puerto está el *daemon NNTP*, esperando clientes todo el tiempo, o bien se crea sobre la marcha según se necesita. Tras establecerse la conexión, el cliente y el servidor se comunican usando una secuencia de comandos y respuestas. Estos comandos y respuestas sirven para asegurar que el cliente reciba todos los artículos que necesita, pero no duplicados, sin importar la cantidad de alimentadores de noticias que use. Los principales comandos usados para mover artículos entre *daemons* de noticias se listan en la figura 7-56.

Comando	Significado
LIST	Dame una lista de todos los grupos de noticias y artículos que tienes
NEWGROUPS fecha hora	Dame una lista de grupos de noticias creados después de fecha/hora
GROUP grp	Dame una lista de todos los artículos en grp
NEWGROUPS grp fecha hora	Dame una lista de artículos nuevos de los grupos especificados
ARTICLE id	Dame un artículo específico
POST	Tengo un artículo para tí que se publicó aquí
IHAVE id	Tengo id de artículo. ¿Lo quieres?
QUIT	Termina sesión

Figura 7-56. Comandos NNTP principales para la difusión de noticias.

Los comandos *LIST* y *NEWGROUPS* permiten que el cliente averigüe qué grupos tiene el servidor. El primero produce la lista completa. El segundo lista sólo los grupos creados después de la fecha y hora especificadas. Si el cliente sabe que la lista es larga, es más eficiente que él lleve el control de lo que tiene cada uno de sus alimentadores de noticias y simplemente solicite actualizaciones. La respuesta de cada uno de estos comandos es una lista, en ASCII, con un grupo de noticias por línea, que proporciona el nombre del grupo de noticias, el número del último artículo que tiene el servidor, el número del primer artículo que tiene el servidor, y una bandera que indica si se permite la publicación en este grupo de noticias o no.

Una vez que el cliente sabe qué grupos de noticias tiene el servidor, puede comenzar a preguntar por los artículos que tiene el servidor (por ejemplo, para grupos viejos cuando se usa *NEWGROUPS*). Los comandos *GROUP* y *NEWSGROUPS* tienen este fin. De nuevo, el primero produce la lista completa y el segundo sólo lista las actualizaciones posteriores a la fecha y hora indicadas, normalmente la hora de la última conexión con este alimentador de noticias. El primer parámetro puede contener asteriscos, para indicar todos. Por ejemplo, *comp.os.** significa todos los grupos de noticias que comienzan con la cadena *comp.os*.

Una vez que el cliente tiene una lista completa de los artículos existentes en los grupos (e incluso antes de tener la lista completa), puede comenzar a preguntar por los artículos que

necesita usando el comando *ARTICLE*. Una vez que están dentro todos los artículos adquiridos, el cliente puede ofrecer artículos que ha adquirido de otros alimentadores de noticias usando el comando *IHAVE*, y artículos que fueron publicados localmente mediante el comando *POST*. El servidor puede aceptarlos o rechazarlos, según desee. Al concluir el cliente, puede terminar la sesión usando *QUIT*. De esta manera, cada máquina tiene control completo sobre los artículos que recibe de cada alimentador de noticias, eliminando todos los artículos duplicados.

Como ejemplo del funcionamiento del NNTP, considere un proveedor de información, *decente.net*, que quiere evitar controversias a toda costa, por lo que el único grupo de noticias que ofrece es *soc.parejas* y *misc.chicos*. Sin embargo, la gerencia no quiere encerrarse y está dispuesta a manejar otros grupos de noticias, siempre y cuando no contengan material potencialmente ofensivo para nadie; por tanto, quiere que se le informe de todos los grupos de nueva creación para poder tomar una decisión informada a nombre de sus clientes. Una posible situación entre *decente.com* actuando como cliente, y su alimentador de noticias, *feeder.com*, actuando como servidor, se muestra en la figura 7-57. Esta situación usa el enfoque de obtención de noticias (el cliente inicia la conexión para solicitar noticias). Las anotaciones entre paréntesis son comentarios y no forman parte del protocolo NNTP.

En esta sesión, *decente.com* pregunta primero si hay noticias para *soc.parejas*. Cuando se le indica que hay dos artículos, obtiene ambos y los almacena en *news/soc/parejas* como archivo aparte. Cada archivo se nombra por su número de artículo. Luego *decente.com* pregunta sobre *misc.chicos* y se le indica que hay un artículo; lo obtiene y lo pone en *news/misc/chicos*.

Habiendo obtenido todas las noticias sobre los grupos que lleva, ahora busca grupos nuevos y se le indica que han aparecido dos grupos nuevos desde la última sesión. Uno de ellos parece prometedor, por lo que se obtienen los artículos. El otro se ve peligroso, por lo que no se toma. (*decente.com* ha hecho una inversión importante en software de inteligencia artificial para poder ser capaz de determinar lo que va a llevar simplemente viendo los nombres.)

Tras haber adquirido todos los artículos que quiere, *decente.com* ofrece a *feeder.com* un artículo nuevo publicado por alguien de esta instalación. Se acepta la oferta y se transfiere el artículo. Ahora *decente.com* ofrece otro artículo, uno que vino de su otro alimentador de noticias. Dado que *feeder.com* ya tiene éste, lo rechaza. Por último, *decente.com* termina la sesión y libera la conexión TCP.

El enfoque de empuje de noticias es parecido. Comienza por la llamada del alimentador de noticias a la máquina que debe recibir las noticias. El alimentador de noticias normalmente lleva el registro de los grupos de noticias a los que están suscritos sus clientes, y comienza por anunciar su primer artículo en el primero de estos grupos de noticias usando el comando *IHAVE*. El receptor potencial entonces revisa sus tablas para ver si ya tiene el artículo, y puede aceptarlo o rechazarlo. Si el artículo es aceptado, se transmite, seguido de una línea que contiene un punto. Entonces el alimentador de noticias anuncia el segundo artículo, al igual que los demás, hasta que se han transferido todas las noticias.

Un problema con ambos, la obtención de noticias y el empuje de noticias, es que usan parada y espera. Por lo regular, se pierden 100 msec en la espera de la respuesta a una pregunta. Con 100,000 o más artículos de noticias nuevos por día, este tiempo perdido se acumula para generar una carga extra sustancial.

```

S: 200 Servidor NNTP feeder.com a su servicio (respuesta a una conexión nueva)
C: NEWNEWS soc.parejas 960901 030000 (¿hay noticias nuevas en soc.parejas?)
S: 230 Continúa lista de 2 artículos
S: <13281@psyc.berkeley.edu> (el artículo 1 de 2 en soc.parejas es de Berkeley)
S: <162721@aol.com> (el artículo 2 de 2 en soc.parejas es de AOL)
S: .
   (fin de lista)
C: ARTICLE <13281@psyc.berkeley.edu> (por favor dame el artículo de Berkeley)
S: 220 Sigue <13281@psyc.berkeley.edu>
S: (artículo completo <13281@psyc.berkeley.edu> se envía aquí)
S: .
   (fin de artículo)
C: ARTICLE <162721@aol.com> (por favor dame el artículo de AOL)
S: 220 Sigue <162721@aol.com>
S: (artículo completo <162721@aol.com> se envía aquí)
S: .
   (fin de artículo)
C: NEWNEWS misc.chicos 960901 030000 (¿hay noticias nuevas en misc.chicos?)
S: 230 Sigue lista de 1 artículo
S: <43222@bio.rice.edu> (1 artículo de Rice)
S: .
   (fin de lista)
C: ARTICLE <43222@bio.rice.edu> (por favor dame el artículo de Rice)
S: 220 Sigue <43222@bio.rice.edu>
S: (artículo completo <43222@bio.rice.edu> se envía aquí)
S: .
   (fin de artículo)
C: NEWGROUPS 960901 030000
S: 231 Siguen 2 grupos nuevos
S: rec.mascotas
S: rec.desnudo
S: .
C: NEWNEWS rec.mascotas 0 0
   (lista todo lo que tienes)
S: 230 Sigue lista de 1 artículo
S: <124@fido.net> (1 artículo de fido.net)
S: .
   (fin de lista)
C: ARTICLE <124@fido.net> (por favor dame el artículo fido.net)
S: 220 sigue <124@fido.net>
S: (se envía el artículo completo aquí)
S: .
C: POST
S: 340
   (por favor envía tus publicaciones)
C: (artículo publicado en decente.com se envía aquí)
S: 240
   (artículo recibido)
C: IHAVE <5321@foo.com>
S: 435
   (Ya lo tengo, por favor no lo envíes)
C: QUIT
S: 205
   (Que tengas un bonito día)

```

Figura 7-57. Manera en que *decente.com* podría adquirir artículos nuevos de su alimentador de noticias.

7.6. LA WORLD WIDE WEB

La World Wide Web es un armazón arquitectónico para acceder a documentos vinculados distribuidos en miles de máquinas de toda la Internet; en cinco años, pasó de ser una manera de distribuir datos sobre física de alta energía a la aplicación que millones de personas piensan que es "La Internet". Su enorme popularidad se deriva del hecho de que tiene una interfaz gráfica atractiva que es fácil de usar por los principiantes y proporciona un enorme cúmulo de información sobre casi cualquier tema concebible, desde aborigenes hasta zoología.

La Web (también conocida como WWW) comenzó en 1989 en el CERN, el Centro Europeo de Investigación Nuclear. El CERN tiene varios aceleradores en los que los científicos de los países europeos participantes llevan a cabo investigaciones sobre física de partículas. Estos equipos con frecuencia tienen miembros de media docena de países o más. La mayoría de los experimentos son altamente complejos, y requieren años de planeación adelantada y construcción de equipo. La Web surgió de la necesidad de lograr que estos grandes grupos de investigadores dispersos internacionalmente colaboraran usando un conjunto siempre cambiante de informes, planos, dibujos, fotos y otros documentos.

La propuesta inicial de una red (*web*) de documentos vinculados surgió del físico del CERN Tim Berners-Lee en marzo de 1989. El primer prototipo (basado en texto) estaba en operación 18 meses después. En diciembre de 1991 se hizo una demostración pública en la conferencia Hypertext '91 en San Antonio, Texas. El desarrollo continuó durante el siguiente año, culminando con la liberación de la primera interfaz gráfica, Mosaic, en febrero de 1993 (Vetter *et al.*, 1994).

Mosaic tuvo tanto éxito que, un año después, su autor, Marc Andreessen, dejó el Centro Nacional de Aplicaciones de Supercómputo, donde se desarrolló Mosaic, para formar una compañía, Netscape Communications Corp., cuya meta fue desarrollar clientes, servidores y otros tipos de software de la Web. Cuando Netscape se volvió pública en 1995, los inversionistas, aparentemente pensando que ésta era la siguiente Microsoft, pagaron 1,500 millones de dólares por las acciones. Esta marca fue aún más sorprendente porque la compañía sólo tenía un producto, operaba con grandes pérdidas y había anunciado en su prospecto que no esperaba tener ganancias en un futuro cercano.

En 1994, el CERN y el M.I.T. firmaron un acuerdo para establecer el World Wide Web Consortium, una organización dedicada al desarrollo de la Web, la estandarización de protocolos y el fomento de interoperabilidad entre las instalaciones. Berners-Lee se convirtió en el director. Desde entonces, cientos de universidades y compañías se han unido al consorcio. El M.I.T. opera la parte de Estados Unidos del consorcio, y el centro francés de investigación, INRIA, opera la parte europea. Aunque hay más libros sobre la Web de los que pueden contarse, el mejor lugar para recibir información actualizada sobre la Web es (naturalmente) la Web misma. La página base (*home page*) del consorcio puede encontrarse en <http://www.w3.org>. Los lectores interesados pueden encontrar ahí vínculos con páginas que cubren todos los documentos y actividades del consorcio.

En las siguientes secciones describiremos la Web desde el punto de vista del usuario y, especialmente, su funcionamiento interno. Puesto que la Web básicamente es un sistema cliente-

servidor, estudiaremos tanto el lado del cliente (es decir, el usuario) como el lado del servidor. Despues analizaremos el lenguaje de escritura de las páginas de la Web (HTML y Java). Por último, examinaremos la manera de encontrar información en la Web.

7.6.1. El lado del cliente

Desde el punto de vista del usuario, la Web consiste en un enorme conjunto a nivel mundial de documentos, generalmente llamados páginas. Cada página puede contener vínculos (apuntadores) con otras páginas relacionadas en cualquier lugar del mundo. Los usuarios pueden seguir un vínculo (por ejemplo, haciendo clic en él), lo que los lleva a la página apuntada. Este proceso puede repetirse indefinidamente, posiblemente atravesando cientos de páginas vinculadas al hacerlo. Se dice que las páginas que apuntan a otras páginas usan **hipertexto**.

Las páginas se ven mediante un programa llamado visor (*browser*); Mosaic y Netscape son dos de los visores más populares. El visor obtiene la página solicitada, interpreta el texto y los comandos de formateo que contiene, y exhibe la página, adecuadamente formateada, en la pantalla. En la figura 7-58(a) se da un ejemplo. Al igual que muchas páginas de la Web, ésta comienza con un título, contiene cierta información y termina con la dirección de correo electrónico del encargado del mantenimiento de la página. Las cadenas de texto que son vínculos a otras páginas, llamadas **hipervínculos** (*hyperlinks*), se resaltan, ya sea mediante subrayado, presentación en un color especial, o ambas cosas. Para seguir un vínculo, el usuario coloca el cursor en el área resaltada (usando el ratón o las teclas de flechas) y la selecciona (haciendo clic con un botón del ratón o pulsando ENTER). Aunque existen algunos visores no gráficos, como Lynx, no son tan comunes como los visores gráficos, por lo que nos concentraremos en estos últimos. También se están desarrollando visores basados en voz.

Los usuarios con curiosidad respecto al Departamento de psicología animal pueden aprender más sobre él haciendo clic en su nombre (subrayado). El visor entonces trae la página a la que está vinculado el nombre y la presenta, como se muestra en la figura 7-58(b). También puede hacerse clic en los elementos subrayados aquí para traer otras páginas, y así se puede continuar. La página nueva puede estar en la misma máquina que la primera, o en otra máquina del otro lado del mundo. El usuario no puede saberlo. La obtención de páginas la hace el visor sin ayuda del usuario. Si el usuario llega a regresar a la página principal, los vínculos ya seguidos tal vez aparezcan con un subrayado punteado (y posiblemente otro color) para distinguirlos de los vínculos que no se han seguido. Nótese que hacer clic en la línea de *información de la Universidad* de la página principal no tiene ningún efecto; no está subrayada, lo que significa que simplemente es texto y no está vinculada con otra página.

La mayoría de los visores tienen muchos botones y características que simplifican la navegación en la Web. Muchos tienen un botón para regresar a la página anterior, un botón para avanzar a la siguiente página (sólo operativo una vez que el usuario ha regresado de ella), y un botón para ir directamente a la propia página base del usuario. Muchos visores tienen un botón o elemento de menú para poner una marca en una página dada y otro para visualizar la lista de marcas, haciendo posible regresar a cualquiera de ellas mediante un solo clic del ratón. Las páginas también pueden almacenarse en disco o imprimirse. Generalmente hay numerosas

BIENVENIDOS A LA PÁGINA BASE DE WWW DE LA UNIVERSIDAD DE EAST PODUNK

- Información sobre la Universidad
 - [Información sobre Inscripciones](#)
 - [Mapa de las instalaciones](#)
 - [Indicaciones para llegar a las instalaciones](#)
 - [El cuerpo estudiantil de la UEP](#)
- Departamentos académicos
 - [Departamento de psicología animal](#)
 - [Departamento de estudios alternativos](#)
 - [Departamento de cocina microbiótica](#)
 - [Departamento de estudios no tradicionales](#)
 - [Departamento de estudios tradicionales](#)

Webmaster@eastpodunk.edu

(a)

DEPARTAMENTO DE PSICOLOGÍA ANIMAL

- [Información sobre posibles carreras](#)
- Personal
 - [Membros del profesorado](#)
 - [Estudiantes de postgrado](#)
 - [Personal no académico](#)
- [Proyectos de investigación](#)
- [Puestos disponibles](#)
- Nuestros cursos más populares
 - [Manejo de herbívoros](#)
 - [Administración de caballos](#)
 - [Negociando con su mascota](#)
 - [Construcción de perreras amables con el usuario](#)
- [Lista completa de cursos](#)

Webmaster@animalpsyc.eastpodunk.edu

(b)

Figura 7-58. (a) Página de Web. (b) Página accedida al hacer clic en *Departamento de psicología animal*.

opciones disponibles para controlar la distribución de la pantalla y ajustar varias preferencias del usuario. En (Berghel, 1996) se comparan nueve visores.

Además de tener texto normal (no subrayado) e hipertexto (subrayado), las páginas de la Web también pueden contener iconos, dibujos de líneas, mapas y fotografías. Cada uno de estos puede vincularse (opcionalmente) con otra página. Hacer clic en uno de estos elementos causa que el visor traiga la página vinculada y la exhiba, igual que ocurre con el texto. En el caso de imágenes como fotografías y mapas, la página que se trae a continuación puede depender de la parte de la imagen en la que se hizo clic.

No todas las páginas pueden visualizarse de la manera convencional. Por ejemplo, algunas páginas consisten en pistas de audio, fragmentos de video, o ambas cosas. El resultado de mezclar páginas de hipertexto con otros medios se llama hipermedia. Algunos visores pueden presentar todos los tipos de hipermedia, pero otros no; en vez de ello, revisan un archivo de configuración para ver la manera de manejar los datos recibidos. Normalmente, el archivo de configuración indica el nombre de un programa, llamado visor externo, o aplicación ayudante, que se ejecutará con la página como entrada. Si no hay un visor configurado, el visor generalmente le pide al usuario que escoja uno. Si no existe ningún visor, el usuario puede indicar al visor que guarde la página entrante en un archivo en disco, o la descarte. Las aplicaciones ayudantes para producir voz hacen posible que incluso los usuarios ciegos accedan a la Web. Otras aplicaciones ayudantes contienen intérpretes de lenguajes de Web especiales, haciendo posible descargar y ejecutar programas desde las páginas de la Web. Este mecanismo hace posible extender la funcionalidad de la Web misma.

Muchas páginas de la Web contienen imágenes grandes, que tardan mucho tiempo en cargarse. Por ejemplo, traer una imagen no comprimida de 640 × 480 (VGA) con 24 bits por pixel (992 KB) requiere unos cuatro minutos a través de un módem de 28.8 kbps. Algunos visores manejan la carga lenta de imágenes obteniendo y presentando primero el texto y luego trayendo las imágenes mismas. Esta estrategia ofrece al usuario algo que leer mientras las imágenes llegan y permite que el usuario cancele el cargado si la página no es lo suficientemente interesante para justificar la espera. Una estrategia alterna es proporcionar una opción para inhabilitar la obtención y presentación automática de imágenes.

Algunos escritores de páginas intentan interesar a usuarios potencialmente aburridos presentando imágenes de una manera especial. Primero, la imagen aparece con baja definición. Luego se llenan gradualmente los detalles. Para el usuario, ver la imagen completa tras unos cuantos segundos, aunque sea con baja definición, con frecuencia es preferible a verla construirse lentamente desde arriba, línea de barrido por línea de barrido.

Algunas páginas de la Web contienen formas que solicitan información al usuario. Las aplicaciones típicas para estas formas son la búsqueda de elementos proporcionados por el usuario en una base de datos, la colocación de un pedido de un producto y la participación en una encuesta de opinión. Otras páginas de la Web contienen mapas que permiten a los usuarios hacer clic en ellos para exhibir acercamientos u obtener información sobre alguna zona geográfica. El manejo de formas y de mapas activos (con clic) requiere un proceso más complejo que simplemente traer una página conocida. Describiremos después la manera de implementar estas características.

Algunos visores usan el disco local para poner en caché las páginas que han traído. Antes de traer una página, se comprueba si ya está en el caché local. De ser así, sólo es necesario verificar si la página está actualizada. De ser así, la página no necesita cargarse nuevamente. Como resultado, normalmente es muy rápido hacer clic en el botón BACK para ver la página previa.

Para ser host de un visor de la Web, una máquina debe estar directamente en Internet o, cuando menos, tener una conexión SLIP o PPP con un enrutador u otra máquina conectada directamente a Internet. Este requisito existe porque la manera en que un visor trae una página es estableciendo una conexión TCP con la máquina en la que está la página, y luego enviando un mensaje a través de la conexión solicitando la página. Un visor no funcionará si no puede establecer una conexión TCP a una máquina arbitraria de Internet.

A veces son impresionantes los extremos a los que llega la gente para acceder a la Web. Cuando menos una compañía ofrece servicio de Web por fax. Un cliente sin acceso a Internet llama al servidor de Web por fax y se conecta usando el teclado del teléfono; luego teclea un código que identifica la página de Web deseada, la cual se envía al fax del cliente.

7.6.2. El lado del servidor

Cada instalación de la Web tiene un proceso servidor que escucha en el puerto TCP 80, esperando conexiones entrantes de los clientes (normalmente visores). Tras establecerse una conexión, el cliente envía una solicitud y el servidor envía una respuesta. Después se libera la conexión. El protocolo que define las solicitudes y respuestas legales se llama HTTP; lo estudiaremos con algún detalle a continuación, pero un ejemplo sencillo de su uso puede proporcionar una idea razonable sobre el funcionamiento de los servidores de la Web. En la figura 7-59 se muestra la manera en que encajan varias de las partes de la Web.

Para este ejemplo, podemos imaginar que el usuario acaba de hacer clic en alguna parte del texto o, tal vez, en un ícono que apunta a la página cuyo nombre (URL — Uniform Resource Locator, localizador uniforme de recursos) es <http://www.w3.org/hypertext/WWW/TheProject.html>. También explicaremos los URL posteriormente en este capítulo. Por el momento, basta con saber que un URL tiene tres partes: el nombre del protocolo (*http*), el nombre de la máquina donde se encuentra la página (www.w3.org) y el nombre del archivo que contiene la página (*hypertext/WWW/TheProject.html*). Los pasos que se ejecutan entre el clic del usuario y la presentación de la página son los siguientes:

1. El visualizador determina el URL (viendo lo que se seleccionó).
2. El visualizador solicita al DNS la dirección IP de www.w3.org.
3. El DNS contesta con 18.23.0.23.
4. El visualizador establece una conexión TCP con el puerto 80 en 18.23.0.23.
5. A continuación, el visualizador emite un comando *GET/hypertext/WWW/TheProject.html*.
6. El servidor www.w3.org envía el archivo *TheProject.html*.

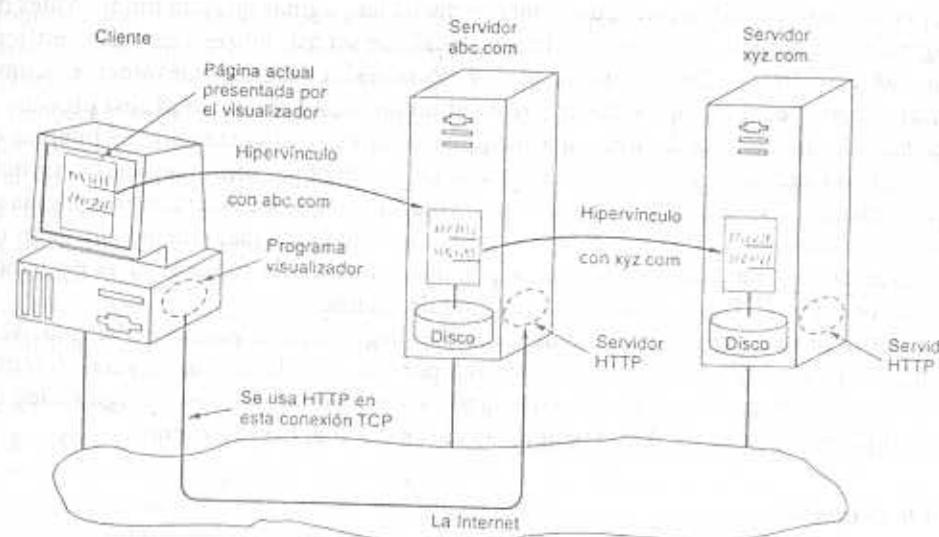


Figura 7-59. Partes del modelo de la Web.

7. Se libera la conexión TCP.
8. El visualizador presenta todo el texto de *TheProject.html*.
9. El visualizador trae y presenta todas las imágenes de *TheProject.html*.

Muchos visualizadores presentan el paso que están ejecutando en el momento en una línea de estado en la parte inferior de la pantalla. De esta manera, cuando el desempeño es malo, el usuario puede ver si se debe a que el DNS no responde, el servidor no responde o simplemente hay congestionamiento en la red durante la transmisión de la página.

Vale la pena indicar que, para cada imagen en línea (ícono, dibujo, fotografía, etc.) de una página, el visualizador establece una conexión TCP nueva con el servidor pertinente para traer la imagen. Sobra decir que si una página contiene muchos íconos, todos en el mismo servidor, el establecer, usar y liberar una conexión nueva para cada uno no es muy eficiente, pero simplifica la implementación. Las modificaciones futuras del protocolo se encargarán del problema de la eficiencia. En (Mogul, 1995) se hace una propuesta.

Puesto que el HTTP es un protocolo ASCII como el SMTP, es bastante fácil que una persona en una terminal (sin usar un visualizador) hable directamente con servidores de la Web. Todo lo que se necesita es una conexión TCP al puerto 80 del servidor. La manera más sencilla de efectuar tal conexión es usar el programa Telnet. En la figura 7-60 se muestra un ejemplo de cómo se puede lograr esto. Aquí, las líneas marcadas con *C:* son tecleadas por el usuario (cliente), las líneas marcadas con *T:* son producidas por el programa Telnet, y las líneas marcadas con *S:* son producidas por el servidor de M.I.T.

```

C: telnet www.w3.org 80
T: Trying 18.23.0.23...
T: Connected to www.w3.org.
T: Escape character is "].
C: GET /hypertext/WWW/TheProject.html HTTP/1.0
C:
S: HTTP/1.0 200 Document follows
S: MIME-Version: 1.0
S: Server: CERN/3.0
S: Content-Type: text/html
S: Content-Length: 8247
S:
S: <HEAD> <TITLE> The World Wide Web Consortium (W3C) </TITLE> </HEAD>
S: <BODY>
S: <H1> <IMG ALIGN=MIDDLE ALT="W3C" SRC="icons/WWW/w3c_96x67.gif">
S: El World Wide Web Consortium <H1> <P>
S:
S: La World Wide Web es el universo de la información accesible por red.
S: El <A HREF="Consortium/"> World Wide Web Consortium </A>
S: existe para hacer realidad el potencial completo de la Web. <P>
S:
S: El W3C trabaja con la comunidad mundial para producir
S: <A HREF="#Specifications"> especificaciones </A> y
S: <A HREF="#Reference"> software de referencia </A>.
S: El W3C es financiado por industriales
S: <A HREF="Consortium/Member/List.html"> miembros </A>
S: pero sus productos están disponibles gratuitamente para todos. <P>
S:
S: En este documento:
S: <menu>
S: <LI> <A HREF="#Specifications"> Especificaciones y áreas de desarrollo de la Web </A>
S: <LI> <A HREF="#Reference"> Software de la Web </A>
S: <LI> <A HREF="#Community"> La World Wide Web y la comunidad de la Web </A>
S: <LI> <A HREF="#Joining"> Participación en el W3C </A>
S: </menu>
S: <P> <HR>
S: Los anfitriones del W3C son
S: el <A HREF="http://www.lcs.mit.edu/"> Laboratorio de informática </A> del
S: <A HREF="http://web.mit.edu"> MIT </A>, y
S: y, en Europa, el <A HREF="http://www.inria.fr/"> INRIA </A>.
S: </BODY>

```

Figura 7-60. Situación ejemplo para obtener una página de la Web.

Se invita al lector a intentar este diálogo personalmente (de preferencia desde un sistema UNIX, puesto que otros sistemas no devuelven el estado de la conexión). Asegúrese de incluir los espacios y la versión del protocolo en la línea *GET*, y la línea en blanco que sigue a la línea *GET*. Como nota, el texto que se recibirá será diferente del que se muestra en la figura 7-60 por cuatro razones. Primero, la salida de ejemplo se resumió y editó para que cupiera en una página. Segundo, se limpió un tanto la salida para evitar poner en ridículo al autor, quien sin duda esperaba que miles de personas examinaran la página formateada, pero que ninguna revisara el HTML que la produjo. Tercero, el contenido de la página cambia constantemente. Cuarto, aquí se presenta una traducción. No obstante, este ejemplo debe dar una idea razonable sobre el funcionamiento del HTTP.

Lo que muestra el ejemplo es lo siguiente. El cliente, en este caso una persona, pero normalmente un visualizador, primero se conecta con cierto *host* y envía un comando solicitando una página en particular y especificando el protocolo y la versión a usar (HTTP/1.0). En la línea 7, el servidor responde con una línea de estado que indica el protocolo usado (el mismo que el cliente) y el código 200, que significa OK. Esta línea va seguida de un mensaje MIME RFC 822, del que se muestran cinco de las líneas de cabecera en la figura (se han omitido varias más por cuestión de espacio). Luego viene una línea en blanco, seguida del cuerpo del mensaje. Para enviar una fotografía, el campo *Content-Type* podría ser

Content-Type: Image/GIF

De esta manera, los tipos MIME permiten el envío de objetos arbitrarios de una manera estándar. Como nota, la cabecera *Content-Transfer-Encoding* de MIME no se requiere, puesto que el TCP permite el envío de corrientes arbitrarias de bytes, incluso imágenes, sin modificación. El significado de los comandos encerrados entre corchetes angulares usados en la página de ejemplo se estudiará más adelante en este capítulo.

No todos los servidores hablan HTTP. En particular, muchos servidores viejos usan FTP, Gopher u otros protocolos. Dado que hay una gran cantidad de información útil disponible en los servidores FTP y Gopher, una de las metas de diseño de la Web fue poner esta información a la disposición de los usuarios de la Web. Una solución es hacer que el visualizador use estos protocolos al hablar con un servidor FTP o Gopher. De hecho, algunos visualizadores usan esta solución, pero hacer que los visualizadores entiendan todos los protocolos posibles aumenta innecesariamente su tamaño.

Con frecuencia se usa una solución diferente: los servidores apoderados (Luotonen y Altis, 1994). Un servidor apoderado es una clase de pasarela que habla HTTP con el visualizador pero FTP, Gopher o algún otro protocolo con el servidor; acepta solicitudes HTTP y las traduce a, digamos, solicitudes FTP, de modo que el visualizador no tenga que entender ningún otro protocolo más que HTTP. El servidor apoderado puede ser un programa que se ejecuta en la misma máquina que el visualizador, pero también puede estar en una máquina independiente en algún lugar de la red, sirviendo a muchos visualizadores. En la figura 7-61 se muestra la diferencia entre un visualizador que puede hablar FTP y otro que usa un apoderado.

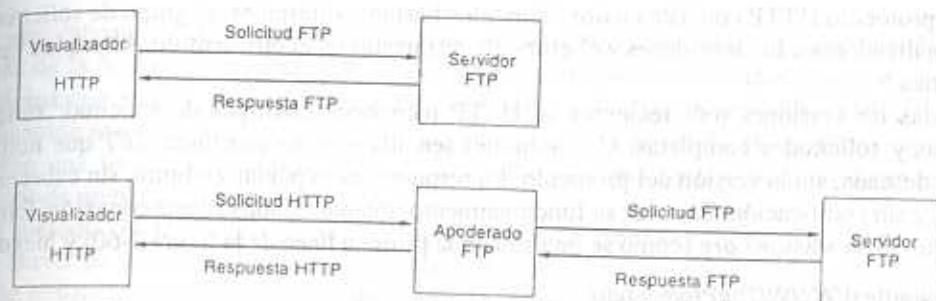


Figura 7-61. (a) Visualizador que habla FTP. (b) Visualizador que no.

Con frecuencia, los usuarios pueden configurar sus visualizadores con apoderados para protocolos que los visualizadores no hablan. De esta manera se aumenta la gama de fuentes de información a las que tiene acceso el visualizador.

Además de actuar como intermediario con protocolos desconocidos, los servidores apoderados tienen varias otras funciones importantes, como el manejo de caché. Un servidor apoderado con caché recolecta y guarda todas las páginas que pasan por él. Cuando un usuario solicita una página, el servidor apoderado verifica si tiene la página. De ser así, puede comprobar si la página aún está vigente. En caso de que la página siga estando vigente, se pasará al usuario; de otro modo, se traerá una copia nueva.

Por último, una organización puede poner un apoderado dentro de su muro de seguridad para que los usuarios puedan acceder a la Web, pero sin tener acceso completo a la Internet. En esta configuración, los usuarios pueden hablar con el servidor apoderado, pero es el servidor apoderado el que se comunica con las instalaciones remotas y trae las páginas a nombre de sus clientes. Este mecanismo puede usarse, por ejemplo, en las escuelas de enseñanza media superior para bloquear el acceso a sitios de la Web que el director considere inadecuadas para las tiernas mentes juveniles.

Para información sobre uno de los servidores de Web más comunes (el *daemon* HTTP de NCSA) y su desempeño, véase (Katz *et al.*, 1994, y Kwan *et al.*, 1995).

HTTP — Protocolo de transferencia de hipertexto

El protocolo estándar de transferencia de la Web es el **HTTP** (*HyperText Transfer Protocol*, protocolo de transferencia de hipertexto). Cada interacción consiste en una solicitud ASCII seguida de una respuesta tipo MIME RFC 822. Aunque es muy común el uso del TCP para la conexión de transporte, no es requerido formalmente por el estándar. Si las redes ATM se vuelven lo bastante confiables, las solicitudes y respuestas HTTP podrían transportarse igualmente en mensajes AAL 5.

El HTTP está evolucionando constantemente. Se usan varias versiones y se están desarrollando otras. El material presentado a continuación es relativamente básico y es poco probable que cambie su concepto, pero algunos detalles podrían ser un poco diferentes en las versiones futuras.

El protocolo HTTP consiste en dos elementos bastante diferentes: el grupo de solicitudes de los visualizadores a los servidores y el grupo de respuestas en el otro sentido. Ahora los veremos por partes.

Todas las versiones más recientes de HTTP reconocen dos tipos de solicitud: solicitudes sencillas y solicitudes completas. Una solicitud sencilla es sólo una línea *GET* que nombra la página deseada, sin la versión del protocolo. La respuesta es la página en bruto, sin cabeceras, sin MIME y sin codificación. Para ver su funcionamiento, intente establecer una conexión Telnet con el puerto 80 de www.w3.org (como se muestra en la primera línea de la figura 7-60) y luego teclee *GET /hypertext/www/TheProject.html*

pero esta vez sin HTTP/1.0. Se devolverá la página sin indicación de su tipo de contenido. Este mecanismo es necesario para la compatibilidad hacia atrás; su uso se reducirá a medida que los visualizadores y los servidores basados en solicitudes completas se vuelvan la regla.

Las solicitudes completas se indican por la presencia de la versión del protocolo en la línea de solicitud *GET*, como en la figura 7-60. Las solicitudes pueden consistir en múltiples líneas, seguidas de una línea en blanco para indicar el final de la solicitud, razón por la que se requirió la línea en blanco en la figura 7-60. La primera línea de una solicitud completa contiene el comando (*GET* es sólo una posibilidad), la página deseada y el protocolo/versión. Las líneas subsiguientes contienen cabeceras RFC 822.

Aunque el HTTP se diseñó para usarse en la *Web*, ha sido intencionalmente más general de lo necesario con miras a aplicaciones futuras orientadas a objetos. Por esta razón, la primera palabra de la línea de solicitud completa es sencillamente el nombre del método (comando) a ejecutar con la página de la *Web* (u objeto general). Los métodos interconstruidos se listan en la figura 7-62. Después de acceder a objetos generales, también pueden estar disponibles métodos adicionales específicos para ese objeto. Los nombres son sensibles a mayúsculas y minúsculas, por lo que *GET* es un método legal, pero *get* no.

El método *GET* solicita al servidor que envíe la página (con lo que queremos decir objeto, en el caso más general), codificada adecuadamente en MIME. Sin embargo, si a la solicitud *GET* le

Método	Descripción
<i>GET</i>	Solicita leer una página de <i>Web</i>
<i>HEAD</i>	Solicita leer la cabecera de una página de <i>Web</i>
<i>PUT</i>	Solicita almacenar una página de <i>Web</i>
<i>POST</i>	Adiciona a un recurso nombrado (p. ej., página de <i>Web</i>)
<i>DELETE</i>	Elimina la página de <i>Web</i>
<i>LINK</i>	Conecta dos recursos existentes
<i>UNLINK</i>	Rompe una conexión existente entre dos recursos

Figura 7-62. Métodos de solicitud HTTP interconstruidos.

sigue una cabecera *If-Modified-Since*, el servidor sólo envía los datos si fueron modificados después de la fecha proporcionada. Usando este mecanismo, un visualizador al que se solicitó una página que está en caché puede solicitarla condicionalmente al servidor, dando la hora de modificación asociada a la página. Si la página en caché aún es válida, el servidor simplemente devuelve una línea de estado anunciando el hecho, eliminando por tanto la carga extra de transferir de nuevo la página.

El método *HEAD* simplemente pide la cabecera del mensaje, sin la página. Este método puede servir para obtener la hora de la última modificación, para recolectar información con fines de indexación, o simplemente para probar la validez de un URL. No existen las solicitudes *HEAD* condicionales.

El método *PUT* es el inverso de *GET*: en lugar de leer una página, la escribe. Este método hace posible construir un conjunto de páginas de la *Web* en un servidor remoto. El cuerpo de la solicitud contiene la página y puede codificarse usando MIME, en cuyo caso las líneas que siguen a *PUT* podrían incluir cabeceras *Content-Type* y de validación de identificación, para demostrar que el solicitante efectivamente tiene permiso de ejecutar la operación solicitada.

Algo parecido a *PUT* es el método *POST*; también lleva un URL pero, en lugar de reemplazar los datos existentes, se "anexa" a ellos en algún sentido generalizado. La publicación de un mensaje en un grupo de noticias y la adición de un archivo a un sistema de boletines electrónicos son ejemplos de anexión en este contexto. Claramente, la intención aquí es hacer que la *Web* asuma la funcionalidad del sistema de noticias USENET.

DELETE hace lo que podría esperarse: elimina la página. Como con *PUT*, la validación de identificación y los permisos desempeñan un papel principal. No hay garantía de que *DELETE* tendrá éxito, puesto que, incluso si el servidor HTTP remoto está dispuesto a borrar la página, el archivo subyacente puede tener un modo que prohíba al servidor HTTP su modificación o eliminación.

Los métodos *LINK* y *UNLINK* permiten establecer conexiones entre páginas existentes u otros recursos.

Cada solicitud recibe una respuesta que consiste en la línea de estado y, posiblemente, información adicional (por ejemplo, toda o parte de una página de la *Web*). La línea de estado puede tener el código 200 (OK) o cualquiera de varios códigos de error, por ejemplo 304 (no modificado), 400 (lista errónea) o 403 (prohibido).

Los estándares HTTP describen las cabeceras y los cuerpos de mensaje con considerable detalle. Basta decir que son muy parecidos a los mensajes MIME RFC 822, por lo que no los veremos aquí.

7.6.3. Escritura de una página de *Web* en HTML

Las páginas de *Web* se escriben en un lenguaje llamado HTML (*HyperText Markup Language*, lenguaje de marcación de hipertexto). El HTML permite a los usuarios producir páginas de *Web* que incluyen texto, gráficos y apuntes a otras páginas de *Web*. Comenzaremos nuestro estudio del HTML con estos apuntes, puesto que son el pegamento que mantiene unida a la *Web*.

Los URL — Localizadores uniformes de recursos

Repetidamente hemos dicho que las páginas de *Web* pueden contener apuntadores a otras páginas de *Web*. Ahora es tiempo de ver la implementación de estos apuntadores. Cuando se creó la *Web*, de inmediato fue evidente para lograr que una página de *Web* apuntara a otra se requerían mecanismos para nombrar y localizar las páginas. En particular, había tres preguntas que debían contestarse antes de poder presentar visualmente una página:

1. ¿Cómo se llama la página?
2. ¿Dónde está la página?
3. ¿Cómo se puede acceder a la página?

Si cada página tuviera asignado de alguna manera un nombre único, no habría ambigüedad al identificar las páginas. No obstante, el problema no se resolvería. Considere un paralelismo entre gente y páginas. En Estados Unidos, casi todos tienen un número de seguro social, que es un identificador único, puesto que no hay dos personas que tengan el mismo. Sin embargo, armados sólo con el número de seguro social, no hay manera de encontrar la dirección del dueño y, ciertamente, no hay manera de saber si se debe escribir a la persona en inglés, español o chino. La *Web* tiene básicamente los mismos problemas.

La solución escogida identifica las páginas de una manera que resuelve los tres problemas a la vez. A cada página se le asigna un **URL** (*Uniform Resource Locator*, localizador uniforme de recursos) que sirve efectivamente como nombre mundial de la página. Los URL tienen tres partes: el protocolo (también llamado esquema), el nombre DNS de la máquina en la que se encuentra la página y un nombre local que indica de manera única la página específica (generalmente sólo un nombre de archivo de la máquina en la que reside). Por ejemplo, el URL del departamento del autor es

<http://www.cs.vu.nl/welcome.html>

Este URL consta de tres partes: el protocolo (*http*), el nombre DNS del *host* (*www.cs.vu.nl*) y el nombre del archivo (*welcome.html*), con cierta puntuación que separa las partes.

Muchas instalaciones cuentan con ciertos atajos para los nombres de archivos. Por ejemplo, *-user/* podría tener una correspondencia con el directorio WWW de *user*, con la convención de que una referencia al directorio mismo implica cierto archivo, digámos *index.html*. Por tanto, la página base del autor puede encontrarse en

<http://www.cs.vu.nl/-ast/>

aunque el nombre real del archivo es distinto. En muchas instalaciones, un nombre de archivo nulo se refiere por omisión a la página base de la organización.

Ahora debe quedar clara la manera en que funciona el hipertexto. Para que pueda hacerse clic en una parte del texto, el escritor de la página debe proporcionar dos elementos de información: el texto visualizable en el que se puede hacer clic y el URL de la página a la que se debe ir si se selecciona el texto. Al seleccionarse el texto, el visualizador busca el nombre del *host*

usando DNS. Armado ahora con la dirección IP del *host*, el visualizador establece entonces una conexión TCP con el *host*, y envía por esa conexión el nombre de archivo usando el protocolo especificado. Eureka. Regresa la página. Esto es precisamente lo que vimos en la figura 7-60.

Este esquema URL es abierto en el sentido de que es fácil tener protocolos distintos de HTTP. De hecho, se han definido los URL de varios otros protocolos comunes, y muchos visualizadores los entienden. En la figura 7-63 se listan formas ligeramente simplificadas de los más comunes.

Nombre	Usado para	Ejemplo
http	Hipertexto (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Archivo local	/usr/suzanne/prog.c
news	Grupo de noticias	news:comp.os.minix
news	Artículo	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Envío de correo electrónico	mailto:kim@acm.org
telnet	Ingreso remoto	telnet://www.w3.org:80

Figura 7-63. Algunos URL comunes.

Repasemos brevemente la lista. El protocolo *http* es el lenguaje nativo de la *Web*, el hablado por los servidores HTTP; maneja todos los métodos de la figura 7-62, al igual que cualquier método específico del objeto que se requiera.

El protocolo *ftp* se usa para acceder a archivos mediante FTP, el protocolo de transferencia de archivos de Internet. El FTP tiene más de dos décadas de existencia y está bien atrincherado. Numerosos servidores FTP de todo el mundo permiten que gente de cualquier lugar de Internet establezca una sesión y descargue los archivos colocados en el servidor FTP. La *Web* no cambia esto; simplemente hace más sencilla la obtención de archivos mediante FTP, puesto que FTP tiene una interfaz un tanto arcaica. Con el tiempo, el FTP probablemente desaparecerá, pues no hay ninguna ventaja particular en que una instalación opere un servidor FTP en lugar de un servidor HTTP, que puede hacer todo lo que un servidor FTP y más (aunque hay algunas discusiones sobre su eficiencia).

Es posible acceder a un archivo local como página de *Web*, ya sea usando el protocolo *file* o, más sencillamente, con sólo nombrarlo. Este enfoque es parecido a usar FTP pero no requiere un servidor. Por supuesto que sólo funciona con archivos locales.

El protocolo *news* permite a un usuario de la *Web* llamar un artículo como si fuera una página de *Web*. Esto significa que un visualizador de la *Web* simultáneamente es un lector de noticias. De hecho, muchos visualizadores tienen botones o elementos de menú para simplificar la lectura de las noticias USENET todavía más que usando los lectores de noticias estándar.

Se reconocen dos formatos para el protocolo *news*. El primero especifica un grupo de noticias y puede usarse para obtener una lista de artículos de una instalación de noticias preconfigurada. El segundo requiere que se indique el identificador de un artículo específico, en este caso *AA0134223112@cs.utah.edu*. El visualizador entonces trae el artículo dado de su instalación preconfigurada de noticias usando el protocolo NNTP.

El protocolo *gopher* corresponde al sistema Gopher, que se diseñó en la Universidad de Minnesota y se bautizó por los equipos atléticos de la escuela, los Golden Gophers (también es una expresión en jerga que significa "go for", es decir, ve y trae). Gopher es anterior a la Web por varios años. Se trata de un esquema de recuperación de información, conceptualmente parecido a la Web misma, pero que sólo maneja texto y no imágenes. Cuando un usuario se conecta con un servidor Gopher, se le presenta un menú de archivos y directorios, cualquiera de los cuales puede vincularse a otro menú Gopher de cualquier lugar del mundo.

La gran ventaja del Gopher sobre la Web es que funciona muy bien con las terminales ASCII de 25 × 80, de las que todavía quedan bastantes, y, dado que está basado en texto, es muy rápido. En consecuencia, hay miles de servidores Gopher en todo el mundo. Usando el protocolo *gopher*, los usuarios de la Web pueden acceder a Gopher y hacer que cada menú Gopher se presente como página de Web con capacidad de clic. Si no conoce Gopher, intente el ejemplo dado en la figura 7-63 o haga que su máquina de búsqueda de la Web busque "gopher".

Aunque el ejemplo dado no lo ilustra, también es posible enviar una consulta completa a un servidor Gopher usando el protocolo *gopher+*. Lo que se presenta es el resultado de hacer consultas en el servidor Gopher remoto.

Los dos últimos protocolos no tienen en realidad el sabor de la obtención de páginas de la Web, y no son reconocidos por todos los visualizadores, pero de todas maneras son útiles. El protocolo *mailto* permite a los usuarios enviar correo electrónico desde un visualizador de la Web. El procedimiento es hacer clic en el botón OPEN y especificar un URL que consista en *mailto:* seguido de la dirección de correo electrónico del destinatario. La mayoría de los visualizadores responderán presentando una forma que contiene espacios para el tema y otras líneas de cabecera, así como para el texto del mensaje.

El protocolo *telnet* sirve para establecer una conexión en línea con una máquina remota. Se usa de la misma manera que el programa Telnet, lo cual no es sorprendente, puesto que la mayoría de los visualizadores simplemente llaman al programa Telnet como aplicación ayudante. Como ejercicio, intente de nuevo el diálogo de la figura 7-60, pero usando ahora un visualizador de la Web.

En pocas palabras, los URL se han diseñado no sólo para permitir a los usuarios navegar por la Web, sino también para entenderse con FTP, noticias, Gopher, correo electrónico y Telnet, haciendo innecesarios los programas especializados de interfaz de usuario para esos otros servicios, e integrando por tanto casi todos los accesos a Internet en un solo programa, el visualizador de la Web. Si no fuera por el hecho de que este esquema fue diseñado por un investigador de física, podría fácilmente pensarse que lo creó el departamento de publicidad de una compañía de software.

A pesar de todas estas agradables propiedades, el uso creciente de la Web ha sacado a la luz una debilidad inherente del esquema URL. Un URL apunta a un *host* específico. En caso de

páginas a las que se hace referencia constante, sería deseable tener varias copias muy distantes, para reducir el tráfico de la red. El problema es que los URL no proporcionan ninguna manera de referirse a una página sin decir simultáneamente dónde está. No hay manera de decir: "quiero la página xyz, pero no me importa de dónde la traigas". Para resolver este problema y hacer posible la duplicación de páginas, el IETF está trabajando en un sistema de identificadores universales de recursos, o URI (*Universal Resource Identifiers*). Puede pensarse en un URI como un URL generalizado. Este tema es el objeto de numerosas investigaciones.

Aunque hemos estudiado sólo los URL absolutos, también existen los URL relativos. La diferencia es análoga a la diferencia entre el nombre de archivo absoluto */usr/ast/foobar* y simplemente *foobar* cuando el contexto está definido sin ambigüedades.

HTML — Lenguaje de marcación de hipertexto

Ahora que tenemos una buena idea del funcionamiento de los URL, es tiempo de ver el HTML mismo. El HTML es una aplicación del estándar ISO 8879, SGML (*Standard Generalized Markup Language*, lenguaje de marcación estándar generalizado), pero especializado en hipertexto y adaptado a la Web.

Como se mencionó antes, el HTML es un lenguaje de marcación, un lenguaje para describir la manera en que debe formatearse un documento. El término "marcación" viene de los viejos días cuando los revisores de texto en realidad marcaban los documentos para indicarle al formador de impresión (en aquellos tiempos un ser humano) las fuentes a usar y otras cosas. Los lenguajes de marcación contienen, por tanto, comandos explícitos para el formateo. Por ejemplo, en HTML, ** significa inicia modo de negritas y ** significa termina modo de negritas. La ventaja de un lenguaje de marcación sobre uno que no tiene marcación explícita es que la escritura de un visualizador para él es cosa directa: el visualizador simplemente tiene que entender los comandos de marcación. TeX y troff son otros ejemplos bien conocidos de lenguajes de marcación.

Los documentos escritos en un lenguaje de marcación se pueden contrastar con los documentos producidos con un procesador de textos WYSIWYG (*What You See Is What You Get*, lo que se ve es lo que se recibe), como MS-Word® o WordPerfect®. Estos sistemas pueden almacenar sus archivos con una marcación oculta integrada para poder reproducirlos después, pero no todos funcionan de esta manera. Los procesadores de textos para Macintosh, por ejemplo, mantienen la información de formateo en estructuras de datos aparte, no como comandos integrados en los archivos de usuario.

Al integrar los comandos de marcación dentro de cada archivo HTML y estandarizarlos, se hace posible que cualquier visualizador de la Web lea y reformatee cualquier página de Web. La capacidad de reformatear las páginas de Web tras su recepción es crucial porque una página puede haberse producido en una ventana de pantalla completa en una pantalla de 1024 × 768 con color de 24 bits, pero puede tener que presentarse en una ventana pequeña de una pantalla de 640 × 480 con color de 8 bits. Los procesadores de textos WYSIWYG patentados no pueden usarse en la Web porque sus lenguajes de marcación internos (si los tienen) no están estandarizados entre los proveedores, máquinas y sistemas operativos, y tampoco manejan reformateo para

ventanas de diferentes tamaños y pantallas de diferentes definiciones. Sin embargo, los programas de procesamiento de textos pueden ofrecer la opción de guardar documentos en HTML en lugar del formato propietario del proveedor, y algunos de hecho ya lo hacen.

Como el HTTP, el HTML está en constante estado de cambio. Cuando Mosaic era el único visualizador, el lenguaje que interpretaba, el HTML 1.0, era el estándar de facto. Cuando llegaron nuevos visualizadores, hubo la necesidad de un estándar Internet formal, por lo que se generó el estándar HTML 2.0. El HTML 3.0 se creó inicialmente como esfuerzo de investigación para agregar muchas características nuevas al HTML 2.0, incluidas tablas, barras de herramientas, fórmulas matemáticas, hojas de estilos avanzados (para definir la distribución de las páginas y el significado de los símbolos), y otras cosas.

La estandarización oficial del HTML está siendo administrada por el WWW Consortium, pero varios proveedores de visualizadores han agregado sus propias extensiones *ad hoc*. Estos proveedores esperan lograr que la gente escriba páginas de Web usando sus extensiones, a fin de que los lectores de estas páginas necesiten el visualizador del proveedor para interpretar adecuadamente las páginas. Esta tendencia no simplifica la estandarización del HTML.

A continuación haremos una introducción breve al HTML, sólo para dar una idea de lo que es. Aunque ciertamente es posible escribir documentos HTML con cualquier editor estándar (y mucha gente lo hace), también es posible usar editores HTML especiales que pueden hacer la mayoría del trabajo (pero que por lo mismo dan al usuario menos control sobre todos los detalles del resultado final).

Una página de Web decorosa consiste en una cabecera y un cuerpo encerrado entre etiquetas (comandos de formateo) <HTML> y </HTML>, aunque la mayoría de los visualizadores no se quejan si faltan estas etiquetas. Como puede verse en la figura 7-64(a), la cabecera está delimitada por las etiquetas <HEAD> y </HEAD>, y el cuerpo, por las etiquetas <BODY> y </BODY>. Los comandos dentro de las etiquetas se llaman directivas. La mayoría de las etiquetas HTML tienen este formato, es decir, <ALGO> para marcar el comienzo de algo, y </ALGO> para marcar su fin. Hay numerosos ejemplos más en HTML. La mayoría de los visualizadores tienen un elemento de menú VIEW SOURCE o algo parecido. Su selección presenta el código fuente HTML de las páginas actuales, en lugar de su salida formateada.

Las etiquetas pueden escribirse tanto en minúsculas como en mayúsculas. Por tanto, <HEAD> y <head> significan la misma cosa, pero el primero resalta mejor para la mayoría de los lectores. La distribución real del documento HTML no es importante. Los analizadores ignoran los espacios extra y los retornos de carro, puesto que tienen que reformatear el texto para acomodarlo en el área de presentación. En consecuencia, pueden agregarse espacios virtuales a voluntad para hacer más legibles los documentos HTML, algo que la mayoría necesita urgentemente. Como consecuencia adicional, no pueden usarse líneas en blanco para separar párrafos, puesto que simplemente se ignoran. Se requiere una etiqueta específica.

Algunas etiquetas tienen parámetros (nombrados). Por ejemplo

es una etiqueta, , con el parámetro *SRC* puesto a *abc* y el parámetro *ALT* igual a *foobar*. Para cada etiqueta, el estándar HTML da una lista de los parámetros permitidos, si los hay, y lo

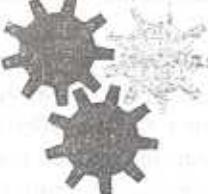
```

<HTML> <HEAD> <TITLE> ADMINÍCULOS AMALGAMADOS, S.A. </TITLE> </HEAD>
<BODY> <H1> Bienvenidos a la página base de AASA </H1>
<IMG SRC="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <BR>
Estamos muy contentos de que haya solicitado visitar la página base de <B> Adminículos
Amalgamados </B>. Esperamos que <i>usted</i> encuentre aquí toda la información que necesita.
<P> A continuación presentamos vínculos con la información sobre nuestro surtido de productos finos.
Puede ordenar electrónicamente (por WWW), por teléfono o por fax. <HR>
<H2> Información sobre nuestros productos </H2>
<UL> <LI> <A HREF="http://widget.com/products/big"> Adminículos grandes </A>
      <LI> <A HREF="http://widget.com/products/little"> Adminículos pequeños </A>
</UL>
<H2> Números telefónicos </H2>
<UL> <LI> Teléfono: 1-800-ADMINIC
      <LI> Fax: 1-415-765-4321
</UL> </CUERPO> </HTML>

```

(a)

Bienvenidos a la página base de AASA



Estamos muy contentos de que haya solicitado visitar la página base de Adminículos Amalgamados. Esperamos que *usted* encuentre aquí toda la información que necesita.

A continuación presentamos vínculos con la información sobre nuestro surtido de productos finos. Puede ordenar electrónicamente (por WWW), por teléfono o por fax.

Información sobre nuestros productos

- [Adminículos grandes](http://widget.com/products/big)
- [Adminículos pequeños](http://widget.com/products/little)

Números telefónicos

- Teléfono: 1-800-ADMINIC
- Fax: 1-415-765-4321

(b)

Figura 7-64. (a) HTML para una página de Web de ejemplo. (b) Página con formato.

que significan. Puesto que cada parámetro se nombra, el orden en que se dan los parámetros no es de importancia.

Técnicamente, los documentos se escriben en el conjunto de caracteres Latin-1 del ISO 8859-1, pero para los usuarios cuyos teclados sólo reconocen ASCII, hay secuencias de escape para los caracteres especiales, como è. La lista de caracteres especiales se proporciona en el estándar. Todos esos caracteres comienzan con un signo & y terminan con un punto y coma. Por ejemplo, è produce è y ´ produce é. Puesto que <, >, y & tienen significados especiales, pueden expresarse sólo con sus secuencias de escape, < y > y & respectivamente.

El elemento principal de la cabecera es el título, delimitado por <TITLE> y </TITLE>, pero pueden estar presentes también ciertos tipos de metainformación. El título mismo no se presenta en la página. Algunos visualizadores lo usan para rotular la ventana de la página.

Veamos ahora algunas de las otras características mostradas en la figura 7-64. Todas las etiquetas usadas en esa figura y algunas más se presentan en la figura 7-65. Las cabeceras se generan con una etiqueta <H_n>, donde *n* es un dígito del intervalo 1 a 6. <H1> es la cabecera más importante; <H6> la menos importante. Es responsabilidad del visualizador presentarlas de manera adecuada en la pantalla. Comúnmente, las cabeceras de número menor se presentarán con una fuente más grande y gruesa. El visualizador puede también seleccionar colores distintos para cada nivel de cabecera. Típicamente, las cabeceras <H1> son grandes y en negritas, con cuando menos una línea en blanco arriba y abajo. En contraste, las cabeceras <H2> tienen una fuente más pequeña y con menos espacio arriba y abajo.

Las etiquetas e <I> se usan para entrar en el modo de negritas y el de cursivas, respectivamente. Si el visualizador no es capaz de presentar negritas y cursivas, debe usar algún otro método para mostrárlas (por ejemplo, un color diferente para cada una o tal vez video inverso). En lugar de especificar estilos físicos como negritas y cursivas, los autores pueden usar estilos lógicos como <DN> (definir), (énfasis débil), (énfasis fuerte) y <VAR> (variables de programa). Los estilos lógicos se definen en un documento llamado hoja de estilos. La ventaja de los estilos lógicos es que, cambiando una definición, pueden cambiarse todas las variables, por ejemplo, de cursivas a una fuente de paso fijo.

El HTML proporciona varios mecanismos para hacer las listas, incluidas listas anidadas. La etiqueta comienza una lista desordenada. Los elementos individuales, que se marcan con la etiqueta en el fuente, aparecen con viñetas (*) al principio. Una variante de este mecanismo es , que es para listas ordenadas. Cuando se usa esta etiqueta, los elementos son numerados por el visualizador. Una tercera opción es <MENU>, que generalmente produce una lista más compacta en la pantalla, sin viñetas y sin números. Aparte del uso de diferentes etiquetas de inicio y fin, , y <MENU> tienen la misma sintaxis y resultados parecidos.

Además de los mecanismos de listas mostrados en la figura 7-65, hay dos más que vale la pena mencionar. <DIR> puede usarse para hacer tablas cortas. También <DL> y </DL> pueden crear listas de definición (glosarios) con entradas de dos partes, que se identifican con <DT> y <DD> respectivamente. La primera es para el nombre, la segunda para su significado. Estas características son sobreseidas en buena medida por el mecanismo de tablas (más general y complejo), que se describe a continuación.

Etiqueta	Descripción
<HTML> ... </HTML>	Declara que la página de Web está escrita en HTML
<HEAD> ... </HEAD>	Delimita la cabecera de la página
<TITLE> ... </TITLE>	Delimita el título (no se presenta en la página)
<BODY> ... </BODY>	Delimita el cuerpo de la página
<H _n > ... </H _n >	Delimita una cabecera nivel <i>n</i>
 ... 	Pone ... en negritas
<I> ... </I>	Pone ... en cursivas
 ... 	Corchetes de una lista desordenada (con viñetas)
 ... 	Corchetes de una lista numerada
<MENU> ... </MENU>	Corchetes de un menú de elementos
	Comienzo de una lista (no hay)
 	Obliga salto de renglón aquí
<P>	Comienzo de párrafo
<HR>	Placa horizontal
<PRE> ... </PRE>	Texto preformato, no reformatear
	Carga una imagen aquí
...	Define un hipervínculo

Figura 7-65. Selección de etiquetas HTML comunes. Algunas tienen parámetros adicionales.

Las etiquetas
, <P> y <HR> indican límites entre secciones del texto. El formato preciso puede determinarse por la hoja de estilos asociada a la página. La etiqueta
 simplemente fuerza una línea nueva. Por lo común, los visualizadores no insertan una línea en blanco tras una
. En contraste, <P> comienza un párrafo y podría, por ejemplo, insertar una línea en blanco y posiblemente una sangría. (En teoría, </P> existe para marcar el fin del párrafo, pero pocas veces se usa; la mayoría de los autores del HTML ni siquiera saben que existe.) Por último, <HR> fuerza una línea nueva y dibuja una línea horizontal a lo ancho de la pantalla.

El HTML 1.0 no tenía capacidad para presentar tablas u otra información formateada. Peor aún, si el escritor de HTML formateaba cuidadosamente una tabla mediante el uso juicioso de espacios y retornos de carro, los visualizadores ignoraban toda la distribución y presentaban la página como si todo el material formateado estuviera sin formatear. Para evitar que los visualizadores alteraran un texto cuidadosamente distribuido, se incluyeron las etiquetas <PRE> y </PRE>; son instrucciones al visualizador para que simplemente presente de manera literal todo lo que está en medio, carácter por carácter, sin cambiar nada. A medida que las tablas y otras características rebuscadas de distribución tengan mayor implementación, disminuirá la necesidad de <PRE>,

excepto para los listados de programa, en los que la mayoría de los programadores no tolerará ningún formateo más que el suyo.

El HTML permite la inclusión de imágenes en línea en una página de *Web*. La etiqueta especifica que se cargará una imagen en la posición actual de la página; puede tener varios parámetros. El parámetro *SRC* indica el URL (o URI) de la imagen. El estándar HTML no especifica los formatos gráficos permitidos. En la práctica, todos los visualizadores reconocen archivos GIF y muchos reconocen también archivos JPEG. Los visualizadores pueden reconocer otros formatos, pero esta extensión es un arma de doble filo. Si un usuario está acostumbrado a un visualizador que reconoce, digamos, archivos BMP, puede incluir éstos en sus páginas de *Web* y luego sorprenderse de que otros visualizadores simplemente ignoren todo su maravilloso arte.

Otros parámetros de son *ALIGN*, que controla la alineación de la imagen con respecto a la línea base del texto (*TOP*, *MIDDLE*, *BOTTOM*), *ALT*, que proporciona texto a usar en lugar de la imagen cuando el usuario ha inhabilitado las imágenes, e *ISMAP*, un indicador de que la imagen es un mapa activo.

Por último, llegamos a los hipervínculos, que usan las etiquetas <A> (ancla) y . Al igual que , <A> tiene varios parámetros, incluidos *HREF* (el URL), *NAME* (el nombre del hipervínculo) y *METHODS* (métodos de acceso), entre otros. Se presenta el texto entre <A> y . Si se selecciona, se sigue el hipervínculo a una nueva página. También se permite poner una imagen ahí, en cuyo caso hacer clic en la imagen también activa el hipervínculo.

Como ejemplo, considere el siguiente fragmento HTML:

```
<A HREF="http://www.nasa.gov"> Página base de la NASA </A>
```

Cuando se presenta una página con este fragmento, lo que aparece en la ventana es

Página base de la NASA

Si el usuario hace clic en este texto, el visualizador de inmediato trae la página cuya URL es <http://www.nasa.gov> y la presenta.

Como segundo ejemplo, considere ahora

```
<A HREF="http://www.nasa.gov"> <IMG SRC="shuttle.gif" ALT="NASA"> </A>
```

Al presentarse, esta página muestra una fotografía (por ejemplo, del transbordador espacial). Al hacer clic en la foto se pasa a la página base de la NASA, igual que al hacer clic en el texto subrayado en el ejemplo anterior. Si el usuario inhabilita la presentación automática de imágenes, el texto NASA se presentará donde iría la fotografía.

La etiqueta <A> puede aceptar un parámetro *NAME* para plantar un hipervínculo, de modo que pueda hacerse referencia a él desde la página. Por ejemplo, algunas páginas de *Web* comienzan con un índice de materias en el que se puede hacer clic. Al hacer clic en un elemento del índice de materias, el usuario salta a la sección correspondiente de la página.

Una característica que el HTML 2.0 no incluyó, y que extrañaron muchos autores de páginas, fue la capacidad de crear tablas a cuyas entradas pudiese hacerse clic para activar hipervínculos.

Como consecuencia, se trabajó mucho para agregar tablas al HTML 3.0. A continuación tenemos una introducción muy breve a las tablas, sólo para captar su sabor básico.

Una tabla HTML consiste en una o más filas, teniendo cada una una o más celdas. Las celdas pueden contener una gran variedad de material, incluidos texto, figuras e inclusive otras tablas. Las celdas pueden unirse, de modo que, por ejemplo, una cabecera puede abarcar varias columnas. Los autores de páginas tienen control limitado sobre la distribución, lo que incluye alineación, estilos de bordes y márgenes de celdas, pero los visualizadores tienen la última palabra al presentar las tablas.

En la figura 7-66(a) se lista una definición de tabla HTML, y en la figura 7-66(b) se muestra una posible presentación. Este ejemplo sólo ilustra unas cuantas características básicas de las tablas HTML. Las tablas comienzan con la etiqueta <TABLE>. Puede proporcionarse información adicional para describir las propiedades generales de la tabla.

La etiqueta <CAPTION> puede usarse para proporcionar un pie de figura. Cada fila comienza con una etiqueta <TR> (fila de tabla). Las celdas individuales se marcan como <TH> (cabecera de tabla) o <TD> (datos de tabla). La distinción se hace para permitir que los visualizadores usen diferentes presentaciones para ellas, como hemos hecho en el ejemplo.

Se permiten muchas otras etiquetas en las tablas, que incluyen maneras de especificar alineaciones horizontales y verticales de las celdas, justificaciones en las celdas, bordes, agrupamiento de celdas, unidades y otras cosas.

Formas

El HTML 1.0 básicamente era de un solo sentido. Los usuarios podían llamar las páginas de los proveedores de información, pero era difícil enviar información en el otro sentido. A medida que más y más organizaciones comerciales comenzaron a usar la *Web*, creció la demanda del tráfico de dos vías. Por ejemplo, muchas compañías querían poder tomar pedidos de productos mediante sus páginas de *Web*, los proveedores de software querían distribuir software por medio de la *Web* y hacer que sus clientes llenaran sus tarjetas de registro electrónicamente, y las compañías que ofrecían búsquedas en la *Web* querían que sus clientes pudieran indicar claves de búsqueda.

Estas demandas condujeron a la inclusión de formas a partir del HTML 2.0. Las formas contienen marcos o botones que permiten a los usuarios proporcionar información o tomar decisiones, y luego devuelven la información al dueño de la página. Las formas usan la etiqueta <INPUT> para este propósito. Esta etiqueta tiene una variedad de parámetros para determinar el tamaño, naturaleza y uso del marco presentado. Las formas más comunes son campos en blanco para aceptar texto del usuario, marcos que pueden marcarse, mapas activos y botones SUBMIT. El ejemplo de la figura 7-67 ilustra algunas de estas opciones.

Comencemos nuestro estudio de las formas repasando este ejemplo. Como todas las formas, ésta se encierra entre las etiquetas <FORM> y </FORM>. El texto no delimitado por etiquetas simplemente se presenta. Se permiten todas las etiquetas normales (por ejemplo,) en una forma. Se usan tres tipos de marcos de entrada en esta forma.

El primer tipo de marco de entrada sigue al texto "Nombre". El marco tiene 46 caracteres de longitud y espera que el usuario escriba una cadena, que luego se almacena en la variable

```
<HTML> <HEAD> <TITLE> Página de ejemplo con tabla </TITLE> </HEAD>
<BODY>
<TABLE BORDER=ALL RULES=ALL>
<CAPTION> Algunas diferencias entre versiones de HTML </CAPTION>
<COL ALIGN=LEFT>
<COL ALIGN=CENTER>
<COL ALIGN=CENTER>
<COL ALIGN=CENTER>
<TR> <TH> Elemento <TH> HTML 1.0 <TH> HTML 2.0 <TH> HTML 3.0
<TR> <TH> Mapas e imágenes activos <TD> <TD> x <TD> x
<TR> <TH> Ecuaciones <TD> <TD> x
<TR> <TH> Formas <TD> <TD> x <TD> x
<TR> <TH> Hipervínculos <TD> x <TD> x <TD> x
<TR> <TH> Imágenes <TD> x <TD> x <TD> x
<TR> <TH> Listas <TD> x <TD> x <TD> x
<TR> <TH> Barras de herramientas <TD> <TD> <TD> x
<TR> <TH> Tablas <TD> <TD> <TD> x
</TABLE> </BODY> </HTML>
```

(a)

Algunas diferencias entre versiones de HTML

Elemento	HTML 1.0	HTML 2.0	HTML 3.0
Mapas e imágenes activos		x	x
Ecuaciones			x
Formas		x	x
Hipervínculos	x	x	x
Imágenes	x	x	x
Listas	x	x	x
Barras de herramientas			x
Tablas			x

Figura 7-66. (a) Tabla HTML de ejemplo. (b) Posible presentación de esta tabla.

customer para procesarse posteriormente. La etiqueta <P> instruye al visualizador que presente el texto y marcos subsiguientes en la siguiente línea, aunque haya espacio en la línea actual. Usando <P> y otras etiquetas de distribución, el autor de la página puede controlar la manera en que se ve la forma en la pantalla.

La siguiente línea de la forma solicita la dirección del usuario, de 40 columnas de ancho, también en una línea individual. Luego viene una línea que solicita la ciudad, el estado y el país. Aquí no se usan etiquetas <P> entre los campos, por lo que el visualizador los presentará todos en la misma línea, si caben. En lo que concierne al visualizador, este párrafo simplemente contiene seis elementos: tres cadenas que alternan con tres marcos. Presenta todo esto de manera

```
<HTML> <HEAD> <TITLE> FORMA DE PEDIDO DE CLIENTES AASA </TITLE> </HEAD>
<BODY>
<H1> Orden de compra de adminículos </H1>
<FORM ACTION="http://widget.com/cgi-bin/widgetorder" METHOD=POST>
Nombre <INPUT NAME="customer" SIZE=46> <P>
Dirección <INPUT NAME="address" SIZE=40> <P>
Ciudad <INPUT NAME="city" SIZE=20> Estado <INPUT NAME="state" SIZE=4>
País <INPUT NAME="country" SIZE=10> <P>
Núm. tarjeta crédito <INPUT NAME="cardno" SIZE=10>
Expira <INPUT NAME="expires" SIZE=4>
M/C <INPUT NAME="cc" TYPE=RADIO VALUE="mastercard">
VISA <INPUT NAME="cc" TYPE=RADIO VALUE="visacard"> <P>
Tamaño de adminículo Grande <INPUT NAME="product" TYPE=RADIO VALUE="expensive">
Pequeño <INPUT NAME="product" TYPE=RADIO VALUE="cheap">
Enviar por mensajería rápida <INPUT NAME="express" TYPE=CHECKBOX> <P>
<INPUT TYPE=SUBMIT VALUE="Enviar orden"> <P>
¡Gracias por ordenar un adminículo AASA, el mejor adminículo del mercado!
</FORM> </BODY> </HTML>
```

(a)

Orden de compra de adminículos

Nombre	<input type="text"/>		
Dirección	<input type="text"/>		
Ciudad	<input type="text"/>	Estado	<input type="text"/>
País	<input type="text"/>		
Núm. tarjeta crédito	<input type="text"/>	Expira	<input type="text"/>
M/C	<input type="radio"/>	Visa	<input type="radio"/>
Tamaño de adminículo	Grande	Pequeño	Enviar por mensajería rápida
<input type="button" value="Enviar orden"/>			

¡Gracias por ordenar un adminículo AASA, el mejor adminículo del mercado!

(b)

Figura 7-67. (a) Código HTML para una forma de pedido. (b) Página formateada.

lineal de izquierda a derecha, pasando a una línea nueva cada vez que la línea actual ya no pueda contener el siguiente elemento. Por tanto, es posible que, en una pantalla de 1024 × 768, las tres cadenas y sus marcos correspondientes aparezcan en la misma línea, pero en una pantalla de 640 × 480 se dividan en dos líneas. En el peor caso, la palabra "País" está al final de una línea y su marco al principio de la siguiente. No hay manera de decirle al visualizador que obligue a que el marco esté adyacente al texto.

La siguiente línea solicita el número de tarjeta de crédito y su fecha de expiración. La transmisión de números de tarjeta de crédito por Internet sólo debe hacerse cuando se han tomado las medidas de seguridad adecuadas. Por ejemplo, algunos, pero no todos, los visualizadores cifran la información enviada por los usuarios. Aun entonces, la comunicación segura y la administración de claves son asuntos complicados y están sujetos a muchos tipos de ataques, como vimos antes.

A continuación de la fecha de expiración encontramos una característica nueva: botones de radio. Éstos se usan cuando debe tomarse una decisión entre dos o más alternativas. El modelo intelectual aquí es un radio de automóvil con media docena de botones para seleccionar estaciones. El visualizador presenta estas opciones de una forma que permite que el usuario las seleccione y las deseccione haciendo clic en ellas (o usando el teclado). Al hacer clic en una de las opciones, se apagan todas las demás del mismo grupo. La presentación visual depende de la interfaz gráfica que se usa. Es responsabilidad del visualizador escoger una forma consistente con Windows, Motif, OS/2 Warp, o cualquier sistema de ventanas que se maneje. El tamaño del adminículo también usa dos botones de radio. Los dos grupos se distinguen por su campo *NAME*, no por alcance estático usando algo como <RADIOBUTTON> ... </RADIOBUTTON>.

Los parámetros de *VALUE* sirven para indicar el botón de radio que se ha presionado. Dependiendo de las opciones de tarjeta de crédito que haya seleccionado el usuario, a la variable *cc* se asignará la cadena "mastercard" o a la cadena "visacard".

Después de los dos grupos de botones de radio, llegamos a la opción de embarque, representada por un marco de tipo *CHECKBOX*, que puede estar encendido o apagado. A diferencia de los botones de radio, de los que sólo se puede seleccionar uno del grupo, cada marco de tipo *CHECKBOX* puede estar encendido o apagado, independientemente de los demás. Por ejemplo, al ordenar una pizza mediante la página de Web de Electropizza, el usuario puede seleccionar sardinas y cebolla y piña (si se atreve), pero no puede seleccionar pequeña y mediana y grande para la misma pizza. Los ingredientes de la pizza se representarían con tres marcos independientes del tipo *CHECKBOX*, mientras que el tamaño de la pizza sería un grupo de botones de radio.

Como nota, en las listas muy largas en las que debe hacerse una selección, los botones de radio son poco prácticos. Por tanto, se proporcionan las etiquetas <SELECT> y </SELECT> para delimitar una lista de alternativas, pero con la semántica de los botones de radio (a menos que se dé el parámetro *MULTIPLE*, en cuyo caso la semántica es la de los marcos marcables). Algunos visualizadores presentan los elementos entre <SELECT> y </SELECT> como menú desplegable.

Hemos visto dos de los tipos interconstruidos de la etiqueta <INPUT>: *RADIO* y *CHECKBOX*. De hecho, ya hemos visto también un tercero: *TEXT*. Como este tipo es el predeterminado, no nos molestamos en incluir el parámetro *TYPE = TEXT*, pero podríamos haberlo hecho. Otros dos tipos son *PASSWORD* y *TEXTAREA*. Un marco *PASSWORD* es igual a un marco *TEXT*, excepto que los caracteres no se presentan cuando se escriben. Un marco *TEXTAREA* es igual a un marco *TEXT*, excepto que puede contener varias líneas.

Regresando al ejemplo de la figura 7-67, ahora nos topamos con un ejemplo de botón *SUBMIT*. Al hacer clic en él, la información del usuario escrita en la forma se envía a la máquina

que proporcionó la forma. Como los demás tipos, *SUBMIT* es una palabra reservada que el visualizador entiende. La cadena *VALUE* aquí es la etiqueta del botón, y se presenta. Todos los marcos pueden tener valores; únicamente necesitamos esa característica aquí. En el caso de los marcos *TEXT*, el contenido del campo *VALUE* se presenta junto con la forma, pero el usuario puede modificarlo o borrarlo. Los marcos *CHECKBOX* y *RADIO* también pueden inicializarse, pero con un campo llamado *CHECKED* (puesto que *VALUE* simplemente da el texto, pero no indica una selección preferida).

El visualizador también entiende el botón *RESET*. Si se hace clic en él, la forma se restablece a su estado inicial.

Vale la pena mencionar dos tipos más. El primero es el tipo *HIDDEN*, que sólo es una salida; no puede hacerse clic en él ni modificarse. Por ejemplo, al trabajar con una serie de páginas en las cuales deben hacerse selecciones, las selecciones previas podrían ser del tipo *HIDDEN*, para evitar que se cambien.

Nuestro último tipo es *IMAGE*, que es para mapas activos (y otras imágenes en las que se puede hacer clic). Cuando el usuario hace clic en el mapa, las coordenadas (*x*, *y*) del píxel seleccionado (es decir, la posición actual del ratón) se almacenan en variables y la forma se devuelve automáticamente al dueño para su procesamiento posterior.

Las formas pueden devolverse de tres maneras: usando el botón *SUBMIT*, haciendo clic en un mapa activo, o tecleando *ENTER* en una forma *TEXT* de un solo elemento. Cuando se devuelve una forma, debe realizarse alguna acción. La acción se especifica mediante los parámetros de la etiqueta <FORM>. El parámetro *ACTION* especifica el URL (o URI) al que debe informarse de la devolución, y el parámetro *METHOD* indica el método a usar. El orden de estos (y todos los demás) parámetros no es significativo.

La manera en que las variables de la forma se devuelven al dueño de la página depende del valor del parámetro *METHOD*. Con *GET*, la única manera de devolver valores es haciendo trampa: se agregan al URL, separados por un signo de interrogación. Este enfoque puede hacer que una URL tenga miles de caracteres de longitud. No obstante, este método se usa con frecuencia porque es sencillo.

Si se usa el método *POST* (véase la figura 7-62), el cuerpo del mensaje contiene las variables de la forma y sus valores. Se usa & para separar los campos; el + representa el carácter de espacio. Por ejemplo, la respuesta a la forma de adminículos podría ser:

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&
product=cheap&express=on
```

La cadena regresaría al servidor como una línea, no tres. Si no se selecciona un *CHECKBOX*, se omite de la cadena. Es responsabilidad del servidor darle sentido a esta cadena.

Afortunadamente ya hay disponible un método estándar de manejo de los datos de una forma. Se llama *CGI* (*Common Gateway Interface*, interfaz de pasarela común). Consideremos una manera común de usarla. Supóngase que alguien tiene una base de datos interesante

(por ejemplo, un índice de las páginas de *Web* por clave y tema) y quiere ponerla a disposición de los usuarios de la *Web*. La manera CGI de hacer disponible la base de datos es escribir un guión (o programa) que hace las veces de interfaz (pasarela) entre la base de datos y la *Web*. A este guión se le asigna un URL, que por convención está en el directorio *cgi-bin*. Los servidores HTTP saben (o se les puede decir) que cuando tengan que invocar un método de una página localizada en *cgi-bin*, deberán interpretar el nombre de archivo como si fuera un guión o programa ejecutable, e iniciarla.

Tarde o temprano, algún usuario abrirá la forma asociada a nuestro guión de administración y lo visualiza. Una vez llenada la forma, el usuario hará clic en el botón SUBMIT. Esta acción causa que el visualizador establezca una conexión TCP con el URL listado en el parámetro *ACTION* de la forma (el guión del directorio *cgi-bin*). Entonces el visualizador invocará la operación especificada por el *METHOD* de la forma, generalmente *POST*. El resultado de esta operación es que se inicia el guión y se presenta (por medio de la conexión TCP, en la entrada estándar) con la cadena grande dada antes. Además, se establecen varias variables de entorno. Por ejemplo, la variable de entorno *CONTENT_LENGTH* indica la longitud de la cadena de entrada.

En este punto, la mayoría de los guiones necesitan analizar su entrada para ponerla en un formato más conveniente. Esto puede lograrse llamando a una de las muchas bibliotecas o procedimientos de guión disponibles. El guión puede entonces interactuar con su base de datos de la manera que desee. Por ejemplo, los mapas activos normalmente usan guiones CGI para emprender diferentes acciones dependiendo del lugar al que apuntó el usuario.

Los guiones CGI también pueden producir salidas y hacer muchas otras cosas además de aceptar entradas de las formas. Si un hipervínculo apunta a un guión CGI, al invocarse ese vínculo se inicia el guión, con varias variables de entorno establecidas para proporcionar información referente al usuario. El guión escribe entonces un archivo (por ejemplo, una página HTML) en la salida estándar, queda donde se envía al visualizador, que lo interpreta. Este mecanismo hace posible que el guión genere páginas de *Web* a la medida en el momento.

Para bien o para mal, algunas instalaciones de la *Web* que contestan consultas tienen una base de datos de anuncios que pueden incluirse selectivamente en la página de *Web* que se construye, dependiendo de lo que busca el usuario. Si el usuario busca "automóvil", podría presentarse un anuncio de General Motors, y si se está buscando "vacaciones", podría aparecer un anuncio de United Airlines. Estos anuncios generalmente incluyen texto e imágenes en las que puede hacerse clic.

7.6.4. Java

El HTML permite describir la manera en que deben exhibirse las páginas estáticas de la *Web*, incluidas tablas y fotografías. Con el recurso *cgi-bin*, también es posible tener una cantidad limitada de interacción de dos sentidos (formas, etc.). Sin embargo, no es posible la interacción rápida con las páginas de la *Web* escritas en HTML. Para hacer posibles páginas de *Web* altamen-

te interactivas, se requiere un mecanismo diferente. En esta sección describiremos uno de tales mecanismos, el lenguaje e intérprete Java^{MR}.

Java se originó cuando algunas personas de Sun Microsystems trataban de desarrollar un lenguaje nuevo adecuado para programar aparatos caseros orientados a información. Después se reorientó el lenguaje hacia la *World Wide Web*. Aunque Java toma prestadas muchas ideas y algo de la sintaxis de C y C++, es un lenguaje nuevo orientado a objetos, incompatible con ambos. A veces se dice que, en grande, Java es como Smalltalk pero, en pequeño, es como C o C++.

La idea principal de usar Java para páginas de *Web* interactivas es que una página de *Web* puede apuntar a un programa Java pequeño, llamado applet (que podría traducirse como "aplicacioncita"). Cuando el visualizador llega a ella, la applet se baja a la máquina cliente y se ejecuta ahí de una manera segura. Debe ser estructuralmente imposible que la applet lea o escriba archivos que no está autorizada a acceder. Debe también ser imposible que la applet introduzca virus o cause algún otro daño. Por estas razones, y para lograr transportabilidad entre máquinas, las applets se compilan para crear un código de bytes después de escribirse y depurarse. Son estos programas en código de bytes a los que apuntan las páginas de *Web*, de manera parecida a como se apunta a las imágenes. Al llegar una applet, se ejecuta interpretativamente en un entorno seguro.

Antes de entrar en los detalles del lenguaje Java, vale la pena decir algunas palabras sobre la utilidad del sistema Java y las razones por las que la gente quiere incluir applets Java en sus páginas de *Web*. Por una parte, las applets permiten que las páginas de *Web* se vuelvan interactivas. Por ejemplo, una página de *Web* puede contener un tablero para jugar gato, Othello o ajedrez, y jugar un juego con el usuario. El programa de juego (escrito en Java) se baja junto con su página de *Web*. Como segundo ejemplo, pueden presentarse formas complejas (por ejemplo, hojas de cálculo), llenando los usuarios elementos y viendo instantáneamente los cálculos.

Es posible que, a la larga, el modelo de gente que compra programas, los instala y los ejecuta localmente será reemplazado por un modelo en el que la gente hace clic en las páginas de *Web* y baja applets que trabajan para ella, posiblemente en colaboración con un servidor o base de datos remoto. En lugar de llenar la declaración de impuestos a mano o usando un programa especial, la gente podrá hacer clic en la página base de la autoridad hacendaria para bajar una applet de impuestos. Esta applet podría hacer algunas preguntas, luego comunicarse con el patrón, banco y corredor de bolsa de la persona a fin de reunir la información necesaria sobre sueldos, intereses y dividendos, llenar la forma y presentarla para verificación y envío.

Otra razón para ejecutar applets en la máquina cliente es que hace posible la adición de animación y sonido a las páginas de *Web* sin tener que llamar a visualizadores externos. El sonido puede reproducirse cuando se carga la página, como música de fondo, o cuando ocurre algún suceso específico (por ejemplo, hacer clic en el gato hace que maúille). Ocurre lo mismo con la animación. Como la applet se ejecuta localmente, aun si se está interpretando, puede escribir en cualquier parte de la pantalla de la manera que quiera y a muy alta velocidad (en comparación con un guión *cgi-bin* remoto).

El sistema Java tiene tres partes:

1. Un compilador de Java a código de bytes.
2. Un visualizador que entiende *applets*.
3. Un intérprete de código de bytes.

El programador escribe la *applet* en Java, luego la compila, obteniendo código de bytes. Para incluir esta *applet* compilada en una página de Web, se ha inventado una nueva etiqueta HTML, <APPLET>. Un uso típico es:

```
<APPLET CODE=game.class WIDTH=100 HEIGHT=200> </APPLET>
```

Cuando el visualizador ve la etiqueta <APPLET>, trae la *applet* compilada *game.class* de la instalación de la página actual de la Web (o, si está presente otro parámetro, *CODEBASE*, del URL que especifica). El visualizador entonces pasa la *applet* al intérprete local de código de bytes para su ejecución (o interpreta la *applet* él mismo, si tiene un intérprete interno). Los parámetros *WIDTH* y *HEIGHT* dan el tamaño de la ventana predeterminada de la *applet*, en píxeles.

De alguna manera, la etiqueta <APPLET> es análoga a la etiqueta . En ambos casos, el visualizador trae un archivo y luego lo entrega a un intérprete (posiblemente interno) para presentarlo en un área limitada de la pantalla; luego continúa procesando la página de Web.

Para algunas aplicaciones que requieren muy alto desempeño, algunos intérpretes Java tienen la capacidad de compilar programas de código de bytes a lenguaje máquina sobre la marcha, según se requiera.

Como consecuencia de este modelo, los visualizadores basados en Java son extensibles de una manera que no lo son los visualizadores de primera generación. Éstos básicamente son intérpretes HTML que tienen módulos interconstruidos para manejar los diferentes protocolos necesarios, como HTTP 1.0, FTP, etc., al igual que decodificadores de varios formatos de imagen. Se muestra un ejemplo en la figura 7-68(a). Si alguien inventa o populariza un formato nuevo, como audio o MPEG-2, estos visualizadores viejos no serán capaces de leer las páginas que los contienen. Cuando mucho, el usuario tendrá que encontrar, bajar e instalar un visualizador externo adecuado.

Con un visualizador basado en Java, la situación es diferente. Al arranque, el visualizador de hecho es una máquina virtual Java vacía, como se muestra en la figura 7-68(b). Al cargar *applets* HTML y HTTP, se vuelve capaz de leer páginas de Web estándar. Sin embargo, a medida que se requieren protocolos y decodificadores nuevos, sus clases se cargan de manera dinámica, posiblemente a través de la red desde instalaciones especificadas en las páginas de Web. Poco tiempo después, el visualizador podría verse como la figura 7-68(c).

Por tanto, si alguien inventa un formato nuevo, todo lo que tiene que hacer la persona es incluir el URL de una *applet* para manejarla en una página de Web, y el visualizador automáticamente traerá y cargará la *applet*. Ningún visualizador de primera generación es capaz de traer e instalar automáticamente visualizadores externos nuevos sobre la marcha. La capacidad de cargar

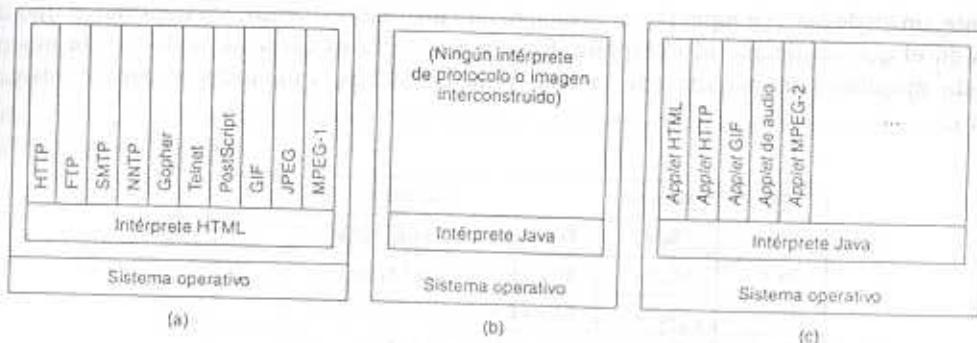


Figura 7-68. (a) Visualizador de primera generación. (b) Visualizador basado en Java al arranque. (c) El visualizador de (b) después de trabajar un rato.

visualizadores dinámicamente significa que la gente puede experimentar fácilmente con formatos nuevos sin primero tener que realizar interminables juntas de estandarización para llegar a un consenso.

Esta capacidad de extensión también se aplica a los protocolos. En algunas aplicaciones se requieren protocolos especiales, por ejemplo, protocolos seguros para aplicaciones bancarias y comerciales. Con Java, estos protocolos pueden cargarse dinámicamente según se requiera, y no hay necesidad de lograr una estandarización universal. Para comunicarse con la compañía X, simplemente bajamos su *applet* de protocolo. Para hablar con la compañía Y, traemos su *applet* de protocolo. No hay necesidad de que X y Y acuerden un protocolo estándar.

Introducción al lenguaje Java

Los objetivos antes listados han conducido a un lenguaje de tipo seguro, orientado a objetos, con capacidad multihilos interconstruida y sin características no definidas o dependientes del sistema. Lo que sigue es una descripción muy simplificada de Java, simplemente para dar una idea de su sabor. Se han omitido muchas características, detalles, opciones y casos especiales en aras de la brevedad. La especificación completa del lenguaje, y mucho más sobre Java, está disponible en la Web misma (por supuesto), en <http://java.sun.com>. En (Campioni y Walrath, 1996, y Van der Linden, 1996) pueden encontrarse tutoriales sobre Java. La historia completa puede encontrarse en (Arnold y Gosling, 1996, y Gosling et al., 1996). Para una comparación breve entre Java y la respuesta de Microsoft a él (Blackbird), véase (Singleton, 1996).

Como mencionamos antes, en pequeño, Java es parecido a C y C++. Las reglas léxicas, por ejemplo, son muy parecidas [por ejemplo, las fichas (*tokens*) se delimitan mediante espacio blanco, y pueden insertarse líneas nuevas entre cualquier par de fichas]. Pueden introducirse comentarios usando la sintaxis de C /* ... */ o la sintaxis de C++ //

Java tiene ocho tipos de datos primitivos, que se listan en la figura 7-69. Cada tipo tiene un tamaño específico, independiente de la implementación local. Por tanto, a diferencia de C, donde un entero puede tener 16, 32 o 64 bits, dependiendo de la arquitectura de la máquina

subyacente, un int de Java siempre tiene 32 bits, ni uno más ni uno menos, sin importar el tipo de máquina en el que se ejecute el intérprete. Esta consistencia es esencial, dado que la misma applet debe ejecutarse en máquinas de 16 bits, 32 bits y 64 bits, y todas deben dar los mismos resultados.

Tipo	Tamaño	Descripción
Byte	1 byte	Entero con signo entre -128 y +127
Short	2 bytes	Entero con signo de 2 bytes
Int	4 bytes	Entero con signo de 4 bytes
Long	8 bytes	Entero con signo de 8 bytes
Float	4 bytes	Número de punto flotante IEEE de 4 bytes
Double	8 bytes	Número de punto flotante IEEE de 8 bytes
Boolean	1 bit	Los únicos valores son verdadero y falso
Char	2 bytes	Carácter en Unicode

Figura 7-69. Tipos de datos Java básicos.

Las variables aritméticas (los primeros 6 tipos) pueden combinarse usando los operadores aritméticos normales (incluyidos ++ y --) y compararse usando los operadores relacionales normales (por ejemplo, <, <=, ==, !=). Se permiten las conversiones entre tipos si tienen sentido.

Java usa caracteres en Unicode de 16 bits en lugar de ASCII, por lo que las variables de caracteres tienen 2 bytes de longitud. Los primeros 127 caracteres Unicode son los mismos que en ASCII, para permitir compatibilidad hacia atrás. Arriba de éstos hay algunos símbolos gráficos y luego los caracteres necesarios para el ruso, el árabe, el hebreo, el japonés (kanji, katakana e hiragana) y prácticamente todos los demás idiomas. Los caracteres no presentes en ASCII pueden representarse mediante \ seguido de cuatro dígitos hexadecimales. Por ejemplo, '\u0ae6' es el cero en Gujarati.

Java permite la declaración de arreglos de una dimensión. Por ejemplo,

```
int[] table;
```

declara un arreglo, *table*, pero no le asigna espacio. Eso puede hacerse después, como en el C++, mediante, por ejemplo

```
table = new int [1024];
```

para asignar un arreglo con 1024 entradas. No es necesario (ni siquiera posible) devolver arreglos que ya no se necesitan; el recolector de basura los toma. Por tanto, no se necesitan las rutinas de biblioteca *malloc* y *free*, altamente susceptibles a errores, para la administración del almacenamiento. Los arreglos pueden inicializarse, y pueden usarse arreglos de arreglos para obtener mayor dimensionalidad, como en C. Están disponibles cadenas, pero se definen por clase, en lugar de ser simplemente arreglos de caracteres que terminan con un byte nulo.

Los enunciados de control de Java se muestran en la figura 7-70. Los primeros nueve tienen esencialmente la misma sintaxis y semántica que en C, excepto que, donde se requiere una expresión booleana, el lenguaje realmente insiste en una expresión booleana. También los postulados break y continue ahora pueden llevar etiquetas que indican cuál de los ciclos etiquetados debe terminarse o repetirse.

Enunciado	Descripción	Ejemplo
Assignment	Asigna un valor	<i>n</i> = <i>i</i> + <i>j</i> ;
If	Decisión booleana	if (<i>k</i> < 0) <i>k</i> = 0; else <i>k</i> = 2* <i>k</i> ;
Switch	Selecciona un caso	switch (<i>b</i>) {case 1: <i>n</i> ++; case 2: <i>n</i> –;}
For	Iteración	for (<i>i</i> = 0; <i>i</i> < <i>n</i> ; <i>i</i> ++) <i>a</i> [<i>i</i>] = <i>b</i> [<i>i</i>];
While	Repetición	while (<i>n</i> < <i>k</i>) <i>n</i> += <i>i</i> ;
Do	Repetición	do { <i>n</i> = <i>n</i> + <i>n</i> } while (<i>n</i> < <i>m</i>);
Break	Enunciado de salida	break <i>etiqueta</i> ;
Return	Regreso	return <i>n</i> ;
Continue	Siguiente iteración	continue <i>etiqueta</i> ;
Throw	Causa excepción	throw new IllegalArgumentException();
Try	Alcance de excepción	try (...) catch (Exception <i>e</i>) {return –1;}
Synchronized	Exclusión mutua	synchronized void update(int <i>s</i>) {...}

Figura 7-70. Enunciados Java. La notación (...) indica un bloque de código.

Los siguientes dos enunciados están en C++, pero no en C. Los enunciados throw y try se encargan del manejo de excepciones. Java define una variedad de excepciones estándar, como un intento de dividir entre cero, y permite a los programadores definir y generar sus propias excepciones. Los programadores pueden escribir manejadores para atrapar excepciones, haciendo innecesario probar constantemente si algo ha funcionado mal (por ejemplo, al leer de un archivo). El enunciado throw genera una excepción, y el enunciado try define un alcance para asociar los manejadores de excepciones a un bloque de código en el que podría ocurrir una excepción.

El enunciado synchronized es nuevo en Java y tiene que ver con el hecho de que los programas Java puedan tener múltiples hilos de control. Para evitar condiciones de competencia, este enunciado delimita un bloque de código (o un procedimiento completo) que no debe tener más de un hilo activo a la vez. Tales bloques de código generalmente se llaman *regiones críticas*. Cuando se ejecuta el enunciado synchronized, el hilo que lo ejecuta debe adquirir el candado asociado a la región crítica, ejecutar el código y luego liberar el candado. Si el candado no está disponible, el hilo espera hasta que queda libre. Al proteger procedimientos completos de esta manera y usar variables de condición, el programador dispone de toda la capacidad de los monitores (Hoare, 1974).

Los programas en Java pueden llamarse con argumentos. El procesamiento de líneas de comandos es parecido al de C, excepto que el arreglo de argumentos se llama *args* en lugar de *argv* y *args[0]* es el primer parámetro, no el nombre del programa. En la figura 7-71 se ilustra un pequeño programa Java que calcula una tabla de factoriales, simplemente para dar una idea de cómo se ve un programa Java pequeño.

```
class Factorial { /* Este programa consiste en una sola clase con dos métodos. */

    public static void main (int argc, String args[] ) { // programa principal
        long i, f, lower = 1, upper = 20; // declaraciones de cuatro longs

        for (i = lower; i <= upper; i++) { // ciclo de menor a mayor
            f = factorial(i); // f = i!
            System.out.println(i + " " + f); // imprime i y f
        }
    }

    static long factorial (long k) { // función factorial recursiva
        if (k == 0)
            return 1; // 0! = 1
        else
            return k * factorial(k-1); // k! = k * (k-1)!
    }
}
```

Figura 7-71. Programa Java para calcular e imprimir de 0! a 20!

A pesar de que ambos son lenguajes orientados a objetos basados en C, Java y C++ difieren en algunos aspectos. Algunas características se quitaron de Java para hacerlo seguro respecto a los tipos y más fácil de leer. Éstas incluyen *#define*, *typedef*, *enum*, *union*, *struct* la sobrecarga de operadores, los apuntadores explícitos, las variables globales, las funciones aisladas y funciones amigas. Casi está de más decir que el enunciado *goto* se ha enviado a ese lugar especial reservado para las características obsoletas de los lenguajes de programación. Se agregaron otras características para dar al lenguaje mayor potencia. Las características agregadas incluyen recolección de basura, multihilos, interfaces de objetos y paquetes.

Orientación a objetos en Java

En los lenguajes de procedimientos como Pascal o C, un programa consiste en un conjunto de variables y procedimientos, sin ningún principio general de organización. En contraste, en los lenguajes orientados a objetos (*cast*) todo es un objeto. Un objeto normalmente contiene algunas variables de estado internas (es decir, escondidas), junto con algunos procedimientos públicos, llamados métodos, para acceder a ellos. Se espera (y puede obligarse a que) los programas que usan el objeto invoquen los métodos para manipular el estado del objeto. De esta manera, el escritor del objeto puede controlar la manera en que los programas usan la informa-

ción del objeto. Este principio se llama encapsulamiento, y es la base de toda la programación orientada a objetos.

Java intenta tomar lo mejor de ambos mundos: puede usarse como lenguaje de procedimientos tradicional o como lenguaje orientado a objetos. El programa en Java de la figura 7-71, por ejemplo, podría haberse escrito igualmente bien en C, y esencialmente de la misma manera. Desde este punto de vista, un subgrupo de Java podría considerarse como una versión depurada de C. Sin embargo, para escribir páginas de Web, es mejor considerar a Java como un lenguaje orientado a objetos, por lo que estudiaremos su orientación a objetos en esta sección.

Un programa en Java consiste en uno o más paquetes, cada uno de los cuales contiene algunas definiciones de clases. Los paquetes pueden accederse remotamente a través de una red, por lo que aquellos destinados para ser usados por mucha gente deben tener nombres únicos. Normalmente se usan nombres jerárquicos, comenzando por el inverso del nombre DNS de su máquina, por ejemplo:

EDU.univ.cs.catie.games.chess

Una definición de clase es una plantilla para producir ejemplares de objetos, cada uno de los cuales contiene las mismas variables de estado y los mismos métodos que todos los otros ejemplares de objetos de su clase. Sin embargo, los valores de las variables de estado de los diferentes objetos son independientes. Las clases, por tanto, son como moldes para cortar galletas: no son las galletas mismas, pero sirven para producir galletas estructuralmente idénticas, produciendo cada molde de galletas una forma diferente de galleta. Una vez producidas, las galletas (objetos) son independientes entre sí.

Los objetos en Java pueden producirse dinámicamente durante la ejecución, por ejemplo mediante

object = new ClassName()

Estos objetos se almacenan en el montículo y los remueve el recolector de basura cuando ya no se necesitan. De esta manera, la administración de almacenamiento en Java es manejada por el sistema, sin necesidad de los temidos procedimientos *malloc* y *free*, ni tampoco de apuntadores específicos.

Cada clase se basa en otra clase. Se dice que una clase de nueva definición es una subclase de la clase en la que se basa, la superclase. Una (sub)clase siempre hereda los métodos de su superclase; puede o no tener acceso directo a las variables internas de la superclase, dependiendo de si la superclase lo quiere o no. Por ejemplo, si una superclase, *A*, tiene los métodos *M1*, *M2* y *M3*, y una subclase *B*, define un método nuevo, *M4*, entonces los objetos creados en *B* tendrán los métodos *M1*, *M2*, *M3* y *M4*. La propiedad por la que una clase automáticamente adquiere todos los métodos de su superclase se llama herencia, y es una propiedad importante de Java. La acción de agregar nuevos métodos a los métodos de la superclase se llama extender la superclase. Como nota, algunos lenguajes orientados a objetos permiten que las clases hereden los métodos de dos o más superclases (herencia múltiple), pero los diseñadores de Java pensaron que esta propiedad era demasiado desordenada e intencionalmente la dejaron fuera.

Dado que cada clase tiene una y sólo una superclase inmediata, el grupo de todas las clases de un programa en Java forma un árbol. La clase de hasta arriba del árbol se llama *Object*, y todas las demás clases heredan sus métodos. Cualquier clase cuya superclase no se menciona explícitamente en su definición es por omisión una subclase de la clase *Object*. La clase *Factorial* de la figura 7-71, por ejemplo, es una subclase de *Object*.

Veamos ahora un ejemplo de los conceptos orientados a objetos que hemos presentado hasta ahora. En la figura 7-72 tenemos un paquete que define dos subclases, *ComplexNumber*, para definir y usar números complejos (es decir, números con una parte real y otra imaginaria), y *test*, para ilustrar la manera en que puede usarse *ComplexNumber*.

```
class ComplexNumber {           // Define una subclase de Object llamada ComplexNumber
    // Hidden data:
    protected double re, im;   // partes real e imaginaria

    // Cinco métodos para administrar los datos ocultos.
    public void complex(double x, double y) {re = x; im = y;}
    public double Real() {return re;}
    public double Imaginary() {return im;}
    public double Magnitude() {return Math.sqrt(re*re + im*im);}
    public double Angle() {return Math.atan(im/re);}
}

class test {                  // Segunda clase, para probar ComplexNumber
    public static void main (String args[]) {
        ComplexNumber c; // declara un objeto de la clase ComplexNumber
        c = new ComplexNumber(); // asigna realmente memoria para c
        c.Complex(3.0, 4.0); // invoca el método Complex para inicializar c
        System.out.println("La magnitud de c es " + c.Magnitude());
    }
}
```

Figura 7-72. Paquete que define dos clases.

Al igual que *Factorial*, la clase *ComplexNumber* se basa en *Object*, puesto que no se nombra ninguna otra superclase en su definición. Cada objeto de la clase *ComplexNumber* representa un número complejo. Cada objeto de esta clase contiene dos variables ocultas, *re* e *im*, ambas números de punto flotante de 64 bits, para representar las partes real e imaginaria, respectivamente. Estas variables no pueden accederse desde fuera de la definición de clase (y sus subclases), puesto que se han declarado como *protected*. Si se hubieran declarado como *private*, habrían sido visibles sólo para *ComplexNumber*, y no para ninguna subclase. Por ahora, *private* habría sido adecuado, pero pronto definiremos una subclase. Si las variables se hubieran decla-

rado como *public*, habrían sido visibles en cualquier lugar en que fuera visible el paquete, destruyendo así gran parte del valor de la programación orientada a objetos. No obstante, existen situaciones en las que se necesita hacer que el estado interno de un objeto sea público.

Se definen cinco métodos para los objetos que pertenecen a la clase *ComplexNumber*. Los usuarios de la clase están restringidos por tanto a las operaciones proporcionadas por estos cinco métodos, y no pueden acceder al estado directamente. Un ejemplo de la manera en que se crean, inicializan y usan los objetos de clase *ComplexNumber* se da en *test*.

Cuando se compila este paquete, el compilador de Java produce dos archivos binarios (de código de bytes), uno para cada una de las clases, y se nombran según su clase. Al teclearse el comando

`java test`

da como resultado la invocación del intérprete Java con la clase *test* como parámetro. El intérprete entonces busca un método llamado *main* y, al encontrarlo, lo ejecuta. El resultado de la ejecución es que se imprime la línea

`La magnitud de c es 5`

Ahora definamos una subclase de *ComplexNumber*, simplemente para ver cómo se hace. Se comienza por importar la clase original, para averiguar qué hace y cuáles son sus métodos. Luego se define una extensión de *ComplexNumber*, que llamaremos *HairyNumber*. La nueva clase hereda automáticamente los cinco métodos presentes en la superclase. Para hacer más interesantes las cosas, definiremos un sexto método, *AddTo*, en la subclase, para sumar un número complejo al objeto, aumentando sus partes real e imaginaria.

La definición de la subclase se muestra en la figura 7-73, junto con otro programa de prueba que ilustra la manera en que puede usarse un objeto que pertenece a la clase *HairyNumber*. Cuando se ejecuta el nuevo programa de prueba, se imprime

`h = (-0.5,6)`

Recuerde que los seis métodos pueden usarse con los objetos *a* y *h*, sin importar dónde se definió cada método. Si definimos ahora otra subclase más basada en *HairyNumber* y le damos, por decir, tres métodos, los objetos producidos a partir de ella tendrán nueve métodos válidos.

Además de agregar métodos nuevos a su superclase, una subclase puede reemplazar los métodos existentes simplemente redifiniéndolos. Por tanto, es posible que una subclase redefina todos los métodos heredados de su superclase, con lo que los objetos que pertenecen a las dos clases no tendrán nada en común. Sin embargo, el hacerlo es de mal gusto, y debe evitarse.

Por último, una clase Java puede definir varios métodos con el mismo nombre pero diferentes parámetros y diferentes definiciones. Cuando el compilador ve una invocación de método que usa este nombre, tiene que basarse en los tipos de los parámetros para determinar el método a usar. Esta propiedad se llama *carga extra* o *polimorfismo*. A diferencia de C++, donde los operadores pueden sobrecargarse también, en Java sólo pueden sobrecargarse los métodos, no los operadores, para hacer más fáciles de entender los programas.

```

import ComplexNumber;           // importa el paquete ComplexNumber

class HairyNumber extends ComplexNumber { // define una clase nueva
    public void AddTo(ComplexNumber z) { // con un método
        re = re + z.Real();
        im = im + z.Imaginary();
    }
}

class test2 {                  // programa de prueba para HairyNumber
    public static void main(String args[]) {
        HairyNumber a, h;          // declara dos HairyNumber

        a = new HairyNumber();     // asigna memoria para a
        h = new HairyNumber();     // asigna memoria para h
        a.Complex(1.0, 2.0);      // asigna un valor a a
        h.Complex(-1.5, 4.0);    // asigna un valor a h
        h.AddTo(a);              // invoca el método AddTo en h
        System.out.println("h = (" + h.Real() + ", " + h.Imaginary() + ")");
    }
}

```

Figura 7-73. Subclase de ComplexNumber que define un método nuevo.

La interfaz de programadores de aplicaciones

Además del lenguaje básico, los diseñadores de Java han definido e implementado unas 200 clases con la versión inicial. Los métodos contenidos en estas clases forman una especie de entorno estándar para los creadores de programas Java. Las clases están escritas en Java, por lo que son transportables a todas las plataformas y sistemas operativos.

Aunque un estudio detallado de todas estas clases y métodos ciertamente está fuera del alcance de este libro, puede ser de interés una descripción breve. Las 200 clases se agrupan en siete paquetes de tamaño desigual, cada uno de los cuales se enfoca hacia algún tema central. Las *applets* que necesiten un paquete en particular pueden incluirlo usando el enunciado *import* de Java. Los métodos contenidos pueden usarse según se requiera. Este mecanismo reemplaza la necesidad de incluir archivos de cabecera en C; también reemplaza la necesidad de bibliotecas, puesto que los paquetes se cargan dinámicamente durante la ejecución al ser invocados.

Los siete paquetes se resumen en la figura 7-74. El paquete *java.lang* contiene clases que pueden verse como parte del lenguaje, pero no lo son técnicamente. Éstas incluyen clases para administrar las clases mismas, hilos y manejo de excepciones. También están aquí las bibliotecas estándar de matemáticas y de cadenas.

Al igual que C, el lenguaje Java no contiene primitivas de E/S. La E/S se hace cargando y usando el paquete *java.io*, análogo a la biblioteca de E/S estándar de C. Se proporcionan

Paquete	Funcionalidad de ejemplo
Java.lang	Clases, hilos, excepciones, matemáticas, cadenas
Java.io	E/S de corrientes y archivos de acceso aleatorio, impresión
Java.net	Sockets, direcciones IP, URL, datagramas
Java.util	Pilas, tablas de dispersión, vectores, hora, fecha
Java.applet	Traer y presentar páginas de Web, audio, clase Object
Java.awt	Eventos, diálogo, menús, fuentes, gráficos, manejo de ventanas
Java.awt.image	Colores, recorte, filtrado y conversión de imágenes
Java.awt.peer	Acceso al sistema de ventana subyacente

Figura 7-74. Los paquetes incluidos en el API estándar.

métodos para leer y escribir cadenas, archivos de acceso aleatorio y hacer el formateo necesario para imprimir. En la figura 7-71 vimos uno de estos métodos, *println*, que hace impresión con formato.

Muy relacionado con la E/S está el transporte por red. Los métodos que buscan y manejan las direcciones IP se localizan aquí. El acceso a los sockets también forma parte de este paquete, al igual que la preparación de datagramas. La transmisión misma se maneja con *java.io*.

La siguiente clase es *java.util*; contiene las clases y métodos para las estructuras de datos comunes, como pilas y tablas de dispersión, de modo que los programadores no tengan que reinventar constantemente la rueda. También se lleva a cabo aquí la administración de fecha y hora.

El paquete *java.applet* contiene parte de la maquinaria básica de las *applets*, incluidos métodos para traer páginas de Web comenzando por sus URL. También tiene métodos para presentar páginas de Web y ejecutar segmentos de audio (por ejemplo, música de fondo). El paquete *java.applet* también contiene la clase *Object*. Todos los objetos heredan sus métodos, a menos que se reemplacen. Estos métodos incluyen hacer clones de un objeto, comparar la igualdad de dos objetos, convertir un objeto en una cadena, y otros más.

Por último, llegamos a *java.awt* y sus dos subpaquetes. AWT significa Abstract Window Toolkit, herramientas de ventana abstracta, y está diseñado para hacer que las *applets* sean transportables entre los sistemas de ventanas. Por ejemplo, ¿cómo debe una *applet* dibujar un rectángulo en la ventana de tal manera que pueda ejecutarse la misma versión compilada (en código de bytes) en UNIX, Windows y Macintosh, aunque cada uno tenga su propio sistema de ventanas? Parte del paquete se encarga de dibujar en la pantalla, por lo que hay métodos para colocar líneas, figuras geométricas, texto, menús, botones, barras de desplazamiento y muchos otros elementos en ella. Los programadores de Java llaman a estos métodos para escribir en la pantalla. Es responsabilidad del paquete *java.awt* hacer las llamadas adecuadas al sistema operativo local para llevar a cabo el trabajo. Esta estrategia significa que *java.awt* tiene que reescribirse

para cada plataforma nueva, pero las *applets* entonces son independientes de la plataforma, lo que es mucho más importante.

Otra tarea importante de esta clase es la administración de eventos. La mayoría de los sistemas de ventanas fundamentalmente son controlados por eventos. Lo que quiere decir esto es que el sistema operativo detecta pulsaciones de teclas, movimientos del ratón, presión y liberación de botones, y otros eventos, y los convierte en llamadas a procedimientos de usuario. En el caso de Java, se proporciona una gran biblioteca de métodos para manejar estos eventos en *java.awt*. Su uso simplifica la escritura de programas que interactúen con el sistema local de ventanas y aun así sean 100% transportables a máquinas con diferentes sistemas operativos y diferentes sistemas de ventanas.

Una parte del trabajo de este paquete se hace en *java.awt.image*, por ejemplo el manejo de imágenes, y en *java.awt.peer*, que permite el acceso al sistema subyacente de ventanas.

Seguridad

Uno de los aspectos más importantes de Java es sus características de seguridad. Cuando se trae una página de Web que contiene una *applet*, ésta automáticamente se ejecuta en la máquina del cliente. Idealmente, no debe caerse ni tirar la máquina del cliente.

Es más, no se requiere mucha imaginación para visualizar a un estudiante emprendedor preparando una página de Web que contiene algún interesante juego nuevo, y luego haciendo publicidad mundialmente a su URL (por ejemplo, publicándola de manera cruzada en todos los grupos de noticias). Lo que no se menciona en la publicación es el pequeño detalle de que la página también contiene una *applet* que, al ejecutarse, de inmediato cifra todos los archivos del disco duro del usuario. Cuando ha terminado, la *applet* anuncia lo que ha hecho y menciona cortésmente que los usuarios que deseen comprar la clave de descifrado deben enviar 1000 dólares en billetes de baja denominación a cierto apartado postal de Panamá.

Además del esquema de millonario instantáneo anterior, hay otros peligros inherentes a permitir la ejecución de código extraño en una máquina. Una *applet* podría andar a la caza de información interesante (correo electrónico almacenado, el archivo de contraseñas, las cadenas del entorno local, etc.) y devolverlas o enviarlas por correo electrónico a través de la red. La *applet* podría también consumir recursos (por ejemplo, llenar discos), presentar fotografías obscenas o consignas políticas en la pantalla, o hacer ruidos irritantes usando la tarjeta de sonido.

Los diseñadores de Java estaban bien conscientes de estos problemas, y erigieron una serie de barreras contra ellos. La primera línea de defensa es un lenguaje seguro respecto a los tipos. Java tiene especificaciones detalladas de los tipos de datos, arreglos verdaderos con verificación de límites y no tiene apuntadores. Estas restricciones hacen imposible que un programador de Java construya un apuntador para leer y escribir localidades arbitrarias de la memoria.

Sin embargo, Trudy, que ha dejado de tratar de descifrar protocolos criptográficos y ha entrado en el negocio mucho más interesante de escribir *applets* maliciosas en Java, simplemente puede escribir o modificar un compilador de C para producir códigos de bytes de Java, pasando por alto todas las salvaguardas proporcionadas por el lenguaje y el compilador de Java.

La segunda línea de defensa es que, antes de ejecutarse una *applet* entrante, debe pasar por un verificador de código de bytes, el cual busca intentos de fabricar apuntadores, ejecutar instrucciones o llamar a métodos con parámetros no válidos, usar variables antes de inicializarlas, etc. Estas comprobaciones supuestamente garantizan que sólo se ejecutarán las *applets* legales, pero Trudy ciertamente trabajará duro para encontrar trucos que el verificador no detecte.

La tercera línea de defensa es el cargador de clases. Puesto que las clases pueden cargarse sobre la marcha, hay el peligro de que una *applet* pueda cargar una de sus propias clases para reemplazar una clase de sistema crítico, pasando por alto las verificaciones de seguridad de esa clase. Este ataque de caballo de Troya se ha hecho imposible dando a cada clase su propio espacio de nombres (una especie de directorio abstracto), y buscando cuidadosamente las clases de sistema antes de buscar clases de usuario. En otras palabras, si el usuario carga una versión maliciosa de *println*, nunca se usará porque siempre se encontrará antes el *println* oficial.

La cuarta línea de defensa es que algunas clases estándar tienen interconstruidas sus propias medidas de seguridad. Por ejemplo, la clase de acceso a archivos mantiene una lista de archivos a los que puede accederse mediante *applets*, y presenta un marco de diálogo cada vez que una *applet* intenta hacer algo que viola las reglas de protección.

A pesar de todas estas medidas, se esperan problemas de seguridad. Primero, puede haber errores en el software de Java que los programadores astutos pueden explotar para pasar por alto la seguridad. El infame gusano de 1988 de Internet usó un error del daemon Finger de UNIX para detener súbitamente miles de máquinas en toda la Internet (Hafner y Markoff, 1991, y Spafford, 1989).

Segundo, si bien puede ser posible evitar que una *applet* haga cualquier cosa excepto escribir en la pantalla, muchas *applets* necesitarán más poder, por lo que, cuando soliciten privilegios adicionales, los usuarios podrían otorgárselos de mala gana (o inocentemente). Por ejemplo, las *applets* podrían necesitar escribir archivos temporales, por lo que los usuarios podrían darles acceso al directorio */tmp*, pensando que no hay nada de importancia ahí. Desgraciadamente, la mayoría de los editores guardan ahí las versiones temporales de los documentos y correo electrónico que se están editando, de modo que las *applets* maliciosas pueden copiarlos y tratar de enviarlos a través de la red. Por supuesto, puede ser posible bloquear el acceso de las *applets* a la red, pero entonces muchas podrían no funcionar, por lo que tendría que otorgárseles también este poder.

Pero incluso en el poco probable caso de que se niegue el acceso a la red a las *applets*, éstas pueden ser capaces de transmitir información usando canales disimulados (Lampson, 1973). Por ejemplo, después de obtener alguna información, una *applet* podría formar una corriente de bits usando el reloj de tiempo real del sistema local. Para enviar un 1, calcula intensamente durante ΔT ; para enviar un 0, simplemente espera ΔT .

Para adquirir esta información, el dueño de la *applet* puede establecer una conexión con la máquina del cliente a fin de leer algunas de sus páginas de Web públicas o procesar con FTP algunos de sus archivos públicos. Vigilando cuidadosamente la tasa de entrada de datos, el dueño de la *applet* puede ver si ésta está haciendo cálculos (y reduciendo, por tanto, la corriente de salida observada) o descansando. Por supuesto, este canal es ruidoso, pero puede manejarse mediante técnicas estándar. La corriente puede dividirse en marcos delimitados por bytes indica-

dores, los marcos individuales pueden usar un código de corrección de errores robusto, y todos los marcos pueden enviarse dos o tres veces. Existen muchos otros canales disimulados, y es extremadamente difícil descubrirlos y bloquearlos a todos. Para mayor información sobre los problemas de seguridad en Java, véase (Dean y Wallach, 1995).

En pocas palabras, Java genera muchas posibilidades y oportunidades nuevas en la *World Wide Web*; permite que las páginas de *Web* sean interactivas y contengan animación y sonido; también permite que los visualizadores sean infinitamente extensibles. Sin embargo, el modelo Java de bajar *applets* también genera algunos problemas de seguridad nuevos y graves que aún no se han resuelto por completo.

7.6.5. Localización de información en la *Web*

Aunque la *Web* contiene una vasta cantidad de información, no siempre es fácil encontrar el elemento adecuado. A fin de hacer más fácil que las personas encuentren las páginas útiles para ellas, varios investigadores han escrito programas que indizan la *Web* de varias maneras. Algunos de éstos ya son tan comunes que se han vuelto comerciales. Los programas que hacen búsquedas en la *Web* a veces se llaman máquinas de búsqueda, arañas, orugas, gusanos, o *knowbots* (robots de conocimientos). En esta sección haremos una breve introducción a este tema. Para mayor información, véase (Pinkerton, 1994, y McBryan, 1994).

Aunque la *Web* es enorme, reducida a su parte más esencial, la *Web* es un grafo enorme, siendo las páginas los nodos y los hipervínculos los arcos. Los algoritmos para visitar todos los nodos de un grafo son bien conocidos. Lo que dificulta la indización de la *Web* es la enorme cantidad de datos que deben manejarse y el hecho de que constantemente están cambiando.

Comencemos nuestro estudio por una meta sencilla: indizar todas las palabras clave de los títulos de las páginas de *Web*. Para nuestro algoritmo necesitaremos tres estructuras de datos. Primero, necesitamos un arreglo lineal grande, *url_table*, que contiene millones de entradas, en última instancia, una por cada página de *Web*. Este arreglo debe mantenerse en memoria virtual, de modo que las partes que no sean de mucho uso se paginen automáticamente a disco. Cada entrada contiene dos apuntadores, uno a la URL de la página y otro al título de la página. Estos dos elementos son cadenas de longitud variable y pueden mantenerse en un montículo (una porción grande no estructurada de memoria virtual a la que pueden agregarse cadenas). El montículo es nuestra segunda estructura de datos.

La tercera estructura de datos es una tabla de dispersión de n entradas de tamaño que se usa como sigue. Cualquier URL puede pasarse por una función de dispersión para producir un entero no negativo menor que n . Todos los URL que se dispersan al valor k se encadenan en una lista vinculada que comienza en la entrada k de la tabla de dispersión. El principal uso de la tabla de dispersión es comenzar por un URL y ser capaz de determinar rápidamente si ya está presente en la tabla *url_table*. Estas tres estructuras de datos se ilustran en la figura 7-75.

La construcción del índice requiere dos fases: búsqueda e indexación. Comencemos con una máquina sencilla para hacer la búsqueda. El corazón de la máquina de búsqueda es un procedimiento recurrente, *process_url*, que toma una cadena URL como entrada y opera como sigue.

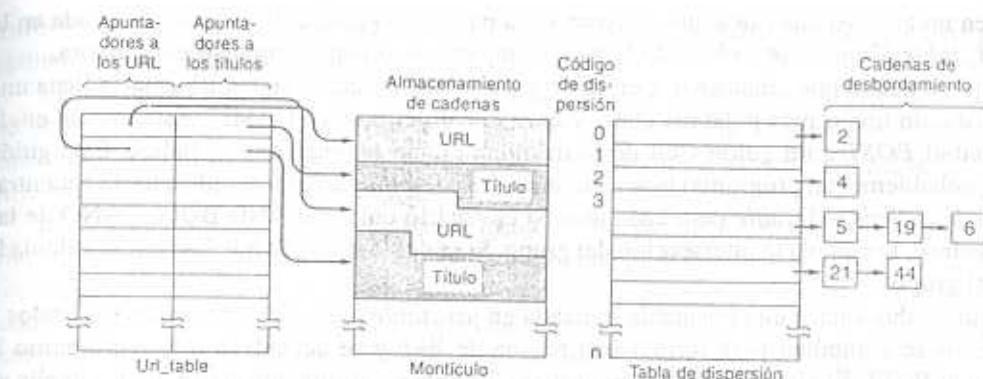


Figura 7-75. Estructuras de datos usadas en una máquina de búsqueda sencilla.

Primero, dispersa el URL para ver si ya está presente en *url_table*. De ser así, ha terminado y regresa de inmediato. Cada URL se procesa sólo una vez.

Si el URL no es conocido aún, se obtiene su página. El URL y el título se copian entonces en el montículo y se generan apuntadores a estas dos cadenas en *url_table*. El URL también se ingresa en la tabla de dispersión.

Por último, *process_url* extrae todos los hipervínculos de la página y llama a *process_url* una vez por hipervínculo, pasando el URL del hipervínculo como parámetro de entrada.

Para ejecutar la máquina de búsqueda, se llama a *process_url* con algún URL inicial. Al regresar, todas las páginas alcanzables desde ese URL han sido ingresadas en *url_table* y se ha completado la fase de búsqueda.

Aunque este diseño es sencillo y correcto en teoría, tiene un problema serio en un sistema tan grande como la *Web*. El problema es que este algoritmo hace una búsqueda primero en profundidad, y en algún momento entrará en recursión tantas veces como la trayectoria no cíclica más larga de la *Web*. Nadie sabe qué tan larga es esta trayectoria, pero probablemente tiene miles de hipervínculos de longitud. Como consecuencia, cualquier máquina de búsqueda que use esta búsqueda primero en profundidad probablemente tendrá un desbordamiento de su pila antes de terminar el trabajo.

En la práctica, las máquinas de búsqueda primero recolectan todos los hipervínculos de cada página que leen, retiran todos los que ya se han procesado, y guardan los demás. Entonces se hace una búsqueda en la *Web* primero en profundidad; es decir, cada vínculo de una página se sigue y se recolectan todos los hipervínculos de todas las páginas a las que apuntan, pero no se siguen en el orden obtenido.

La segunda fase lleva a cabo la indexación de palabras clave. El proceso de indexación recorre linealmente la *url_table*, procesando cada entrada por turno. Para cada entrada, se examina el título y se seleccionan todas las palabras no encontradas en la lista de parada. (La lista de parada evita la indexación de preposiciones, conjunciones, artículos y otras palabras que se encuentran muchas veces pero que son de poco valor.) Por cada palabra seleccionada, se

escribe en un archivo una línea que consiste en la palabra seguida del número de entrada en la tabla *url_table*. Cuando se ha barrido la tabla completa, se ordena el archivo por palabra.

El índice tendrá que almacenarse en disco y puede usarse como sigue. El usuario llena una forma, listando una o más palabras clave y hace clic en el botón SUBMIT. Esta acción envía una solicitud *POST* a un guión CGI de la máquina donde se encuentra el índice. Este guión (o, más probablemente, programa) busca entonces las palabras clave en el índice hasta encontrar el grupo de índices *url_table* para cada una. Si el usuario quiere el AND BOOLEANO de las palabras clave, se calcula la intersección del grupo. Si se desea el OR BOOLEANO, se calcula la unión del grupo.

El guión ahora hace una búsqueda indexada en *url_table* para encontrar todos los títulos y URL. Éstos se combinan para formar una página de Web y se devuelven al usuario como la respuesta al *POST*. El visualizador presenta ahora la página, permitiendo al usuario hacer clic en los elementos que le parezcan interesantes.

¿Suena fácil? No lo es. Tienen que resolverse los siguientes problemas en cualquier sistema práctico:

1. Algunos URL son obsoletos (es decir, apuntan a páginas que ya no existen).
2. Algunas máquinas serán temporalmente inalcanzables.
3. No todas las páginas serán alcanzables desde el URL inicial.
4. Algunas páginas pueden ser alcanzables sólo desde mapas activos.
5. Algunos documentos no pueden indexarse (por ejemplo, fragmentos de audio).
6. No todos los documentos tienen título útil.
7. Podría agotarse la memoria o el espacio de disco de la máquina de búsqueda.
8. El proceso completo podría tardar demasiado.

Los URL obsoletos desperdician tiempo, pero no causan demasiados problemas porque el servidor en el que se supone que están responde de inmediato con un código de error. En contraste, cuando el servidor está inactivo, todo lo que observa la máquina de búsqueda es un gran retardo al establecer la conexión TCP. Para evitar que se quede esperando indefinidamente, debe tener una temporización. Si la temporización es demasiado corta, se perderán varias URL válidas; si es demasiado grande, la búsqueda se hará muy lenta.

La selección del URL inicial ciertamente es un asunto importante. Si la máquina de búsqueda comienza por la página base de algún astrofísico, tarde o temprano podría encontrar todo sobre astronomía, física, química y ciencias espaciales, pero podría pasar de largo páginas sobre medicina veterinaria, inglés antiguo y *rock and roll*. Estos grupos podrían simplemente estar desunidos. Una solución es recolectar un grupo de varios URL tan grande como sea posible, y luego usar cada uno de ellos como página inicial. Los URL iniciales pueden recolectarse de los artículos de noticias de USENET y la versión de la semana pasada de la tabla *url_table*, puesto que algunas de estas páginas podrían haber cambiado recientemente (por ejemplo, uno de los

astrofísicos se casó con una veterinaria, y solemnemente actualizaron sus páginas base para que apuntaran una a la otra).

La indexación funciona bien con texto, pero cada vez más páginas contienen elementos que no son de texto, como fotografías, audio y video. Un enfoque es probar cada URL nuevo encontrado con el método *HEAD*, simplemente para devolver su cabecera MIME. Cualquier cosa que no sea tipo *texto* no se busca.

El 20% de las páginas de Web no tiene título, y muchas de aquellas que sí lo tienen, tienen uno casi inútil ("Página de Pepe"). Una gran mejora del índice básico es incluir no sólo títulos, sino también todo el hipertexto. En este enfoque, al barrerse una página, se registran también todos los hipervínculos, junto con la página de la que vinieron y la página a la que apuntan. Tras completarse la fase de búsqueda, pueden indexarse también todas las hiperpalabras.

Aún más ambicioso es indexar todas las palabras importantes de cada página. Para determinar las palabras importantes, puede calcularse la frecuencia de aparición de todas las palabras no encontradas en la lista de parada (por página de Web). Probablemente valga la pena indexar las 10 o 20 palabras más frecuentes. A fin de cuentas, si la palabra "hígado" es la más común de una página, hay la posibilidad de que la página será de interés para los cirujanos biliares (o para los cocineros). Algunas máquinas de búsqueda (por ejemplo, Lycos) usan esta estrategia.

Por último, la máquina de búsqueda puede quedarse sin memoria o terminarse el tiempo. Una estrategia es rediseñar los algoritmos más cuidadosamente. Un enfoque por entero distinto es hacer lo que hace Harvest y descargar el trabajo (Bowman *et al.*, 1994, 1996). En particular, Harvest proporciona un programa que se ejecuta en servidores cooperativos. Este programa hace toda la búsqueda localmente y transmite de regreso el índice local terminado. En la instalación central, todos los índices locales se integran en el índice maestro. Este enfoque reduce en varios órdenes de magnitud la cantidad de memoria, tiempo de CPU y ancho de banda de red necesarios, pero tiene la desventaja principal de requerir que todos los servidores de la Web cooperen ejecutando software ajeno. Dados los problemas potenciales con los virus y gusanos, al solicitarse a un administrador de sistema: "¿por favor puedes ejecutar este programa en tu máquina por mí?", no debe sorprender que muchos se nieguen.

Es pertinente una pequeña recomendación. Aunque la escritura de una máquina de búsqueda suena fácil, una con fallas puede provocar caos en la red al generar grandes cantidades de solicitudes espurias, no sólo desperdiriendo ancho de banda, sino también abrumando a muchos servidores debido a la carga. Si usted no puede resistir la tentación de escribir su propia máquina de búsqueda, la ética de la red requiere que la restrinja al dominio de su propio DNS local hasta que esté completamente libre de errores.

7.7. MULTIMEDIA

Los multimedia son la vaca sagrada de las redes. Cuando se menciona la palabra, tanto los lunáticos como los de traje comienzan a salivar como si estuvieran ante un gran banquete. Los primeros ven retos técnicos inmensos para proporcionar video (interactivo) a solicitud en cada casa. Los segundos ven enormes ganancias en el asunto. Ningún libro sobre redes estaría com-

pleto sin, al menos, una introducción al tema. Dada la extensión de éste, la nuestra será necesariamente breve. Para información adicional sobre este tema fascinante y potencialmente lucrativo, véase (Buford, 1994; Deloddere *et al.*, 1994; Dixit y Skelly, 1995; Fluckiger, 1995; Minoli, 1995, y Steinmetz y Nahrstedt, 1995).

Literalmente, los multimedia sencillamente son dos o más medios. Si el editor de este libro quisiera unirse a la euforia actual por los multimedia, podría anunciar que el libro usa tecnología multimedia. A fin de cuentas, contiene dos medios: texto y gráficos (las figuras). No obstante, cuando la mayoría de la gente se refiere a multimedia, generalmente quieren decir la combinación de dos o más medios continuos, es decir, medios que tienen que ejecutarse durante cierto intervalo de tiempo bien definido, generalmente con alguna interacción con el usuario. En la práctica, normalmente los dos medios son audio y video, es decir, sonido más imágenes en movimiento. Por esta razón, comenzaremos nuestro estudio con una introducción a la tecnología del audio y el video. Después los combinaremos y pasaremos a los sistemas multimedia reales, incluidos vídeo a solicitud y el sistema multimedia de Internet, MBone.

7.7.1. Audio

Una onda de audio (sonido) es una onda acústica (de presión) de una dimensión. Al entrar una onda acústica en el oído, el tímpano vibra, causando que los pequeños huesos del oído interno vibren con él, enviando pulsos nerviosos al cerebro. Estos pulsos los percibe el escucha como sonido. De manera parecida, cuando una onda acústica incide en un micrófono, el micrófono genera una señal eléctrica, que representa la amplitud del sonido como una función del tiempo. La representación, procesamiento, almacenamiento y transmisión de tales señales de audio es una parte principal del estudio de los sistemas multimedia.

La gama de frecuencias perceptibles por el oído humano va de 20 Hz a 20,000 Hz, aunque algunos animales, notablemente los perros, pueden escuchar frecuencias más altas. El oído escucha de manera logarítmica, por lo que la relación entre dos sonidos de amplitudes A y B se expresa convencionalmente en dB (decibeles) de acuerdo con la fórmula

$$\text{dB} = 20 \log_{10}(A/B)$$

Si definimos como 0 dB el límite inferior de la audibilidad (una presión de unas 0.0003 dinas/cm²) para una onda senoidal de 1 kHz, una conversación ordinaria es de unos 50 dB y el umbral del dolor es de 120 dB, lo que representa una gama dinámica de un factor de un millón. Para evitar cualquier confusión, los A y B indicados antes son *amplitudes*. Si usáramos el nivel de potencia, que es proporcional al cuadrado de la amplitud, el coeficiente del logaritmo sería 10, no 20.

El oído es sorprendentemente sensible a variaciones de sonido que duran apenas unos milisegundos. El ojo, en cambio, no nota cambios en el nivel de luz que duran unos cuantos milisegundos. El resultado de esta observación es que fluctuaciones de apenas unos cuantos milisegundos durante una transmisión multimedia afectan la calidad del sonido percibido más de lo que afectan la calidad percibida de la imagen.

Las ondas de audio pueden convertirse a una forma digital mediante un ADC (*Analog Digital Converter*, convertidor analógico a digital). Un ADC toma un voltaje eléctrico como entrada y genera un número binario de salida. En la figura 7-76(a) vemos un ejemplo de onda senoidal. Para representar esta señal de manera digital, simplemente la muestreamos cada ΔT segundos, como lo muestra la altura de las barras de la figura 7-76(b). Si una onda de sonido no es una onda senoidal pura, sino una superposición de ondas senoidales en las que la componente de más alta frecuencia es f , entonces el teorema de Nyquist (véase el capítulo 2) establece que es suficiente tomar muestras a una frecuencia $2f$. Una frecuencia de muestreo mayor no tiene ningún valor, porque no están presentes las frecuencias mayores que detectaría este muestreo más frecuente.

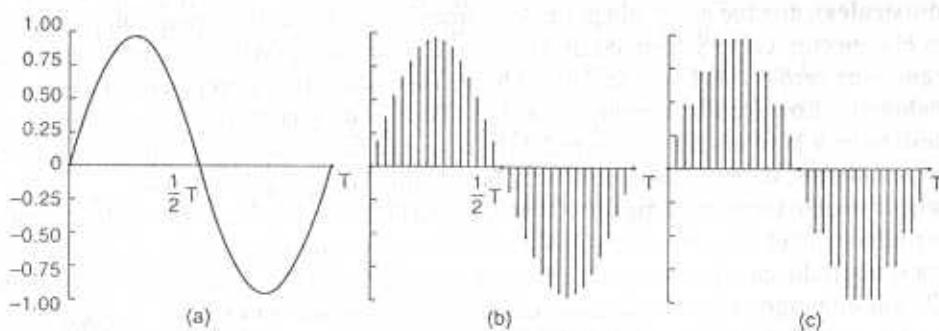


Figura 7-76. (a) Onda senoidal. (b) Muestreo de la onda senoidal. (c) Cuantización de las muestras a 3 bits.

Las muestras digitales nunca son exactas. Las muestras de 3 bits de la figura 7-76(c) permiten sólo ocho valores, de -1.00 a +1.00 en incrementos de 0.25. Una muestra de 8 bits permitirá 256 valores diferentes. Una muestra de 16 bits permitirá 65,536 valores diferentes. El error introducido por la cantidad finita de bits por muestra se llama **ruido de cuantización**. Si éste es demasiado grande, el oído lo detecta.

Dos ejemplos bien conocidos de sonido muestreado son el teléfono y los discos compactos de audio. La modulación de código de pulso, como la usada en el sistema telefónico, emplea muestras de 7 bits (Norteamérica y Japón) y 8 bits (Europa), 8000 veces por segundo. Este sistema da una tasa de datos de 56,000 bps o 64,000 bps. Con sólo 8000 muestras/seg, las frecuencias por arriba de 4 kHz se pierden.

Los CD de audio son digitales, con una tasa de muestreo de 44,100 muestras/seg, suficientes para capturar frecuencias de hasta 22,050 Hz, lo que es bueno para la gente, malo para los perros. Las muestras tienen 16 bits cada una, y son lineales dentro de la gama de amplitudes. Nótese que las muestras de 16 bits permiten sólo 65,536 valores diferentes, aunque la gama dinámica del oído es de aproximadamente 1 millón si se mide en pasos del tamaño del sonido audible más pequeño. Por tanto, el uso de sólo 16 bits por muestra genera algún ruido de cuantización (aunque no se cubre la gama dinámica completa; no se supone que los CD deban lastimar). Con

44,100 muestras/seg de 16 bits cada una, un CD de audio necesita un ancho de banda de 705.6 kbps para monofónico y 1.411 Mbps para estéreo. Si bien esto es menos de lo que necesita el vídeo (véase más adelante), aun así se requiere un canal T1 completo para transmitir sonido estéreo de calidad CD.

El sonido digitalizado puede procesarse fácilmente por software en las computadoras. Existen docenas de programas para computadoras personales que permiten a los usuarios grabar, visualizar, editar, mezclar y almacenar ondas de sonido de fuentes múltiples. Hoy día, casi todas las grabaciones y ediciones profesionales de sonido son digitales.

Muchos instrumentos musicales tienen incluso una interfaz digital. Cuando salieron los primeros instrumentos digitales, cada uno tenía su propia interfaz, pero al poco tiempo se desarrolló un estándar, el MIDI (*Music Instrument Digital Interface*, Interfaz digital de instrumentos musicales), que fue adoptado por prácticamente toda la industria musical. Este estándar especifica el conector, cable y formato de mensaje. Cada mensaje MIDI consiste en un byte de estado seguido de cero o más bytes de datos. Un mensaje MIDI transporta un evento significativo musicalmente. Los eventos comunes son la pulsación de una tecla, el movimiento de un control deslizante y la liberación de un pedal. El byte de estado indica el evento, y los bytes de datos dan parámetros, como la tecla pulsada y la velocidad con la que se pulsó.

Cada instrumento tiene un código MIDI asignado a él. Por ejemplo, un piano de cola tiene el 0, una marimba tiene el 12 y un violín tiene el 40. Esto se requiere para evitar que un concierto para flauta se reproduzca como un concierto para tuba. La cantidad definida de "instrumentos" es de 127. Sin embargo, algunos de éstos no son instrumentos, sino efectos especiales como trinos de pájaro, helicópteros y el aplauso enlatado que acompaña a muchos programas de televisión.

El corazón de todo sistema MIDI es un sintetizador (con frecuencia una computadora) que acepta mensajes y genera música a partir de ellos. El sintetizador entiende los 127 instrumentos, por lo que genera un espectro de potencia diferente para el Do central de una trompeta que para el de un xilófono. La ventaja de transmitir música usando MIDI, en comparación con el envío de una onda digitalizada, es la enorme reducción de ancho de banda, frecuentemente en un factor de 1000. La desventaja del MIDI es que el receptor necesita un sintetizador MIDI para reconstruir la música nuevamente, y diferentes sintetizadores pueden dar reproducciones ligeramente diferentes.

La música, por supuesto, es sólo un caso especial del audio en general, pero es importante. Otro caso especial es la voz. La voz humana tiende a estar en la gama de 600 a 6000 Hz. La voz se compone de vocales y consonantes, que tienen diferentes propiedades. Las vocales se producen cuando la cavidad vocal no está obstruida, produciendo resonancias cuya frecuencia fundamental depende del tamaño y la forma del sistema vocal y la posición de la lengua y quijada del que habla. Estos sonidos son casi periódicos durante intervalos de 30 msec. Las consonantes se producen cuando la cavidad vocal está parcialmente bloqueada. Estos sonidos son menos regulares que las vocales.

Algunos sistemas de generación y transmisión de voz hacen uso de modelos del sistema vocal para reducir la voz a unos cuantos parámetros (por ejemplo, tamaños y formas de varias cavidades), en lugar de simplemente muestrear la forma de onda de la voz.

7.7.2. Vídeo

El ojo humano tiene la propiedad de que, cuando una imagen incide en la retina, se retiene durante algunos milisegundos antes de decaer. Si una secuencia de imágenes incide a 50 o más imágenes/seg, el ojo no nota qué está viendo imágenes discretas. Todos los sistemas de video (es decir, televisión) aprovechan este principio para producir imágenes en movimiento.

Sistemas Analógicos

Para entender los sistemas de video, es mejor comenzar por la antigua y sencilla televisión en blanco y negro. Para representar la imagen bidimensional que está frente a ella como un voltaje unidimensional en función del tiempo, la cámara barre rápidamente un haz de electrones a lo ancho de la imagen y lentamente hacia abajo, registrando la intensidad de la luz a su paso. Al final del barrido, llamado marco, el haz hace un retrazado. Esta intensidad como función de tiempo se difunde, y los receptores repiten el proceso de barrido para reconstruir la imagen. El patrón de barrido que usan tanto la cámara como el receptor se muestra en la figura 7-77. (Como nota, las cámaras CCD integran en lugar de barrer, pero algunas cámaras y todos los monitores hacen un barrido.)

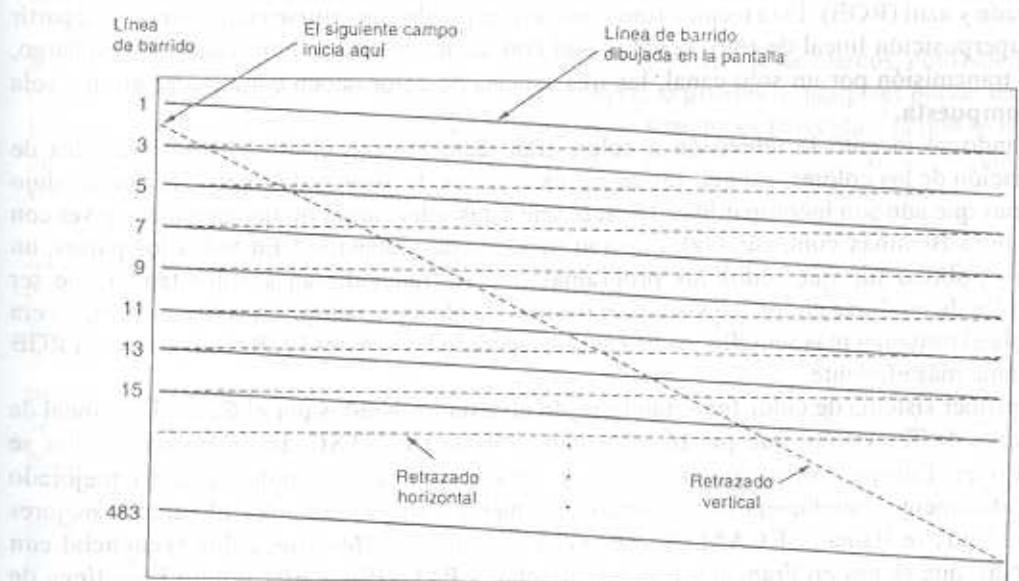


Figura 7-77. Patrón de barrido usado para el video y la televisión NTSC.

Los parámetros de barrido exactos varían de país en país. El sistema usado en Norte y Sudamérica y en Japón tiene 525 líneas de barrido, una relación de aspecto horizontal a vertical

de 4:3, y 30 marcos/seg. El sistema europeo tiene 625 líneas de barrido, la misma relación 4:3, y 25 marcos/seg. En ambos sistemas, no se presentan unas cuantas líneas de arriba y abajo (para aproximar una imagen rectangular a los CRT originales, que eran redondos). Sólo se presentan 483 de las 525 líneas NTSC (y 576 de las 625 líneas de barrido PAL/SECAM). El haz se apaga durante el retrazado vertical, por lo que muchas estaciones (especialmente en Europa) usan este intervalo para difundir Teletexto (páginas de texto que contienen noticias, informes meteorológicos, deportes, precios de acciones, etcétera).

Aunque 25 marcos/seg es suficiente para capturar una imagen continua, con esa tasa de marcos mucha gente, especialmente las personas mayores, percibe un parpadeo de la imagen (porque las imágenes viejas se desvanecen de la retina antes de la aparición de la nueva). En lugar de aumentar la tasa de marcos, lo que requeriría usar más del escaso ancho de banda, se emplea un enfoque diferente. En lugar de presentar las líneas de barrido en orden, primero se presentan las líneas de barrido nenes, y luego las líneas de barrido pares. Cada uno de estos medios marcos se llama campo. Hay experimentos que muestran que, aunque la gente nota el parpadeo a 25 marcos/seg, no lo nota a 50 campos/seg. Esta técnica se llama entrelazado. Se dice que la televisión (o video) no entrelazada es progresiva.

El video de color usa el mismo patrón de barrido que el monocromático (blanco y negro), excepto que, en lugar de presentar la imagen mediante un solo haz en movimiento, se usan tres haces que se mueven al unísono. Se usa un haz para cada uno de los colores primarios aditivos: rojo, verde y azul (RGB). Esta técnica funciona porque puede construirse cualquier color a partir de la superposición lineal de rojo, verde y azul con las intensidades apropiadas. Sin embargo, para la transmisión por un solo canal, las tres señales de color deben combinarse en una sola señal compuesta.

Cuando se inventó la televisión a color, eran técnicamente posibles varios métodos de presentación de los colores, así que varios países tomaron decisiones diferentes, lo que condujo a sistemas que aún son incompatibles. (Nótese qué estas selecciones no tienen nada que ver con VHS contra Betamax contra P2000, que son métodos de grabación.) En todos los países, un requisito político fue que todos los programas que se transmitieran a color tenían que ser susceptibles de recibirse en los televisores existentes de blanco y negro. En consecuencia, no era aceptable el esquema más sencillo, codificar por separado las señales RGB. Tampoco es el RGB el esquema más eficiente.

El primer sistema de color fue estandarizado en Estados Unidos por el Comité Nacional de Estándares de Televisión, que prestó sus siglas al estándar: NTSC. La televisión a color se introdujo en Europa varios años después, y para entonces la tecnología había mejorado significativamente, conduciendo a sistemas con mejor inmunidad contra el ruido y mejores colores. Éstos se llaman SECAM (*SEquentiel Couleur Avec Mémoire*, color secuencial con memoria), que se usa en Francia y Europa Oriental, y PAL (*Phase Alternating Line*, línea de fases alternas), usado en el resto de Europa. La diferencia en la calidad del color entre NTSC y PAL/SECAM ha producido una broma de la industria que sugiere que NTSC significa en realidad Nunca Tendrás Semejanza en los Colores.

Para que las transmisiones puedan verse en receptores de blanco y negro, los tres sistemas combinan linealmente las señales RGB en una señal de luminancia (brillo) y dos de crominancia

(color), aunque usan diferentes coeficientes para construir estas señales a partir de las señales RGB. Resulta interesante que el ojo es mucho más sensible a la señal de luminancia que a las señales de crominancia, por lo que éstas últimas no necesitan transmitirse con tanta precisión. En consecuencia, la señal de luminancia puede difundirse a la misma frecuencia que la vieja señal de blanco y negro, y puede recibirse en los televisores de blanco y negro. Las dos señales de crominancia se difunden en bandas angostas a frecuencias mayores. Algunos aparatos de televisión tienen controles etiquetados brillo, matiz y saturación (o brillo, tinte y color) para controlar estas tres señales por separado. Es necesario el entendimiento de la luminancia y la crominancia para comprender el funcionamiento de la compresión de video.

En los últimos años ha habido un interés considerable en la HDTV (*High Definition Television*, televisión de alta definición), que produce imágenes más nítidas al duplicar (aproximadamente) la cantidad de líneas de barrido. Estados Unidos, Europa y Japón han desarrollado sistemas HDTV, todos diferentes y todos mutuamente incompatibles. Los principios básicos de la HDTV en términos de barrido luminancia, crominancia, etc., son semejantes a los sistemas actuales. Sin embargo, los tres formatos tienen una proporción dimensional común de 16:9 en lugar 4:3 para lograr una correspondencia mejor con el formato usado para el cine (que se graba en película de 35 mm).

Para una introducción a la tecnología de televisión, véase (Buford, 1994).

Sistemas digitales

La representación más sencilla del video digital es una secuencia de marcos, consistiendo cada uno en una malla rectangular de elementos de imagen, o pixeles. Cada píxel puede ser un solo bit, para representar blanco o negro. La calidad de tal sistema es parecida a la que se obtiene al enviar una fotografía a color por fax: espantosa. (Inténtelo si puede; si no, fotocopie una fotografía a color en una máquina copiadora que no maneje tonos.)

El siguiente paso es usar 8 bits por píxel para representar 256 niveles de gris. Este esquema da video en blanco y negro de alta calidad. Para video a color, los sistemas buenos usan 8 bits por cada uno de los colores RGB, aunque casi todos los sistemas los mezclan en video compuesto para su transmisión. Aunque el uso de 24 bits por píxel limita la cantidad de colores a unos 16 millones, el ojo humano no puede diferenciar tantos colores. Las imágenes digitales de color se producen usando tres haces de barrido, uno por color. La geometría es la misma que en el sistema analógico de la figura 7-77, excepto que las líneas continuas de barrido ahora se reemplazan por elegantes filas de pixeles discretos.

Para producir una imagen uniforme, el video digital, al igual que el video analógico, debe presentar cuando menos 25 marcos/seg. Sin embargo, dado que los monitores de computadora de alta calidad barren la pantalla a partir de las imágenes almacenadas en memoria a razón de 75 veces por segundo o más, no se requiere entrelazado y, en consecuencia, normalmente no se usa. Basta con repintar (es decir, redibujar) el mismo marco tres veces consecutivas para eliminar el parpadeo.

En otras palabras, la uniformidad de movimiento la determina la cantidad de imágenes diferentes por segundo, mientras que el parpadeo lo determina la cantidad de veces por

segundo que se pinta el marco. Estos dos parámetros son diferentes. Una imagen fija pintada a 20 marcos/seg no mostrará un movimiento inestable, pero parpadeará porque un marco se desvanece de la retina antes de que aparezca el siguiente. Una película con 20 marcos diferentes por segundo, pintados cada uno cuatro veces seguidas, no parpadeará, pero el movimiento parecerá no uniforme.

El significado de estos dos parámetros se aclara cuando consideramos el ancho de banda necesario para transmitir vídeo digital a través de una red. Todos los monitores de computadora actuales usan la relación de aspecto 4:3 para poder usar tubos de rayos catódicos económicos de producción en masa diseñados para el mercado de televisión de consumidor. Las configuraciones comunes son 640×480 (VGA), 800×600 (SVGA) y 1024×768 (XGA). Una pantalla XGA con 24 bits por píxel y 25 marcos/seg requiere una alimentación a 472 Mbps. Ni siquiera el OC-9 es lo bastante bueno para esto, y la conexión de una portadora SONET OC-9 a la casa de todos no está precisamente en la agenda. La duplicación de esta tasa para evitar el parpadeo es aún menos atractiva. Una mejor solución es transmitir 25 marcos/seg y hacer que la computadora almacene cada uno y lo pinte dos veces. La televisión difundida no usa esta estrategia porque las televisiones no tienen memoria y, en todo caso, las señales analógicas no pueden almacenarse en RAM sin primero convertirse a un formato digital, lo que requeriría *hardware* extra. Como consecuencia, se necesita el entrelazado para la televisión difundida, pero no para el video digital.

7.7.3. Compresión de datos

Debe ser obvio a estas alturas que la transmisión de material multimedia en formato sin compresión es impensable. La única esperanza es que sea posible la compresión masiva. Afortunadamente, un gran número de investigaciones durante las últimas décadas ha conducido a muchas técnicas y algoritmos de compresión que hacen factible la transmisión multimedia. En esta sección estudiaremos algunos métodos para la compresión de datos multimedia, especialmente imágenes. Para mayores detalles, véase (Fluckiger, 1995, y Steininetz y Nahnsiedt, 1995).

Todos los sistemas de compresión requieren dos algoritmos: uno para la compresión de los datos en el origen y otro para la descompresión en el destino. En la literatura, estos algoritmos se conocen como algoritmos de codificación y decodificación, respectivamente. También usaremos esta terminología aquí.

Estos algoritmos tienen ciertas asimetrías cuya comprensión es importante. Primero, en muchas aplicaciones un documento multimedia, digamos una película, sólo se codificará una vez (al almacenarse en el servidor de multimedia), pero se decodificará miles de veces (al ser vista por los clientes). Esta asimetría significa que es aceptable que el algoritmo de codificación sea lento y requiera *hardware* caro, siempre y cuando el algoritmo de decodificación sea rápido y no requiera *hardware* costoso. A fin de cuentas, el operador de un servidor multimedia podría estar dispuesto a rentar una supercomputadora paralela durante algunas semanas para codificar su biblioteca de video completa, pero requerir que los consumidores renten una supercomputadora durante dos horas para ver un video seguramente no tendrá mucho éxito. Muchos sistemas de

compresión prácticos van a los extremos para lograr que la decodificación sea rápida y sencilla, aun al precio de hacer lenta y complicada la codificación.

Por otra parte, para los multimedia de tiempo real, como las videoconferencias, la codificación lenta es inaceptable. La codificación debe ocurrir al momento, en tiempo real. En consecuencia, los multimedia de tiempo real usan algoritmos o parámetros diferentes que el almacenamiento de videos en disco, con frecuencia resultando en una compresión significativamente menor.

Una segunda asimetría es que no es necesario que el proceso de codificación/decodificación se pueda invertir. Por ejemplo, al comprimir, transmitir y descomprimir un archivo de datos, el usuario espera recibir el original, correcto hasta el último bit. En multimedia este requisito no existe. Generalmente es aceptable que la señal de vídeo después de codificar y decodificar sea ligeramente diferente del original. Cuando la salida decodificada no es exactamente igual a la entrada original, se dice que el sistema es con pérdidas (*lossy*). Si la entrada y la salida son idénticas, el sistema es sin pérdidas (*lossless*). Los sistemas con pérdidas son importantes porque aceptar una pequeña pérdida de información puede ofrecer ventajas enormes en la relación de compresión posible.

Codificación entrópica

Los esquemas de compresión pueden dividirse en dos categorías generales: codificación entrópica y codificación por fuente. Los estudiaremos por turno.

La codificación entrópica simplemente manipula las corrientes de bits sin importar lo que significan; es una técnica general, sin pérdidas y completamente reversible que puede aplicarse a todos los tipos de datos. La ilustraremos mediante tres ejemplos.

Nuestro primer ejemplo de codificación entrópica es la codificación por longitud de series. En muchos tipos de datos son comunes las cadenas de símbolos repetidos (bits, números, etc.). Estas pueden reemplazarse por un marcador especial no permitido en los datos en otras condiciones, seguido del símbolo que indica la serie, y luego la cantidad de veces que ocurre. Si el marcador especial mismo ocurre en los datos, se duplica (como en el relleno de caracteres). Por ejemplo, considere la siguiente cadena de dígitos decimales:

Si ahora introducimos A como el marcador y usamos números de dos dígitos para la cuenta de repeticiones, podemos codificar la cadena de dígitos anterior como

315A01284587A11316354674A02265

Aquí la codificación por longitud de serie ha recortado a la mitad la cadena de datos.

Las series son comunes en multimedia. En audio, el silencio con frecuencia se representa mediante series de ceros. En video, series del mismo color ocurren en las tomas del cielo, paredes y muchas superficies planas. Todas estas series pueden comprimirse considerablemente.

Nuestro segundo ejemplo de codificación entrópica es la codificación estadística. Con esto nos referimos al uso de un código corto para representar símbolos comunes y uno largo para

representar los poco frecuentes. El código Morse usa este principio, siendo E un • y Q -- • -, etc. La codificación Huffman y el algoritmo Ziv-Lempel usados por el programa Compress de UNIX también usan codificación estadística.

Nuestro tercer ejemplo de codificación entrópica es la codificación CLUT (*Color Look Up Table*, tabla de búsqueda de colores). Considere una imagen que usa codificación RGB con 3 bytes/píxel. En teoría, la imagen podría contener hasta 2^{24} valores de color diferentes. En la práctica, normalmente contendrá muchos menos valores, especialmente si la imagen es una caricatura o un dibujo generado por computadora, en lugar de una fotografía. Supóngase que sólo se usan en realidad 256 valores de colores. Puede lograrse una compresión de casi un factor de tres construyendo una tabla de 768 bytes que liste los valores RGB de los 256 colores usados, y representando luego cada pixel por el índice de su valor RGB en la tabla. Aquí vemos un ejemplo claro en el que la codificación es más lenta que la decodificación porque la codificación requiere una búsqueda en la tabla, mientras que la decodificación puede hacerse con una sola operación de indización.

Codificación por fuente

Ahora llegamos a la **codificación por fuente**, que aprovecha las propiedades de los datos para producir mayor compresión (generalmente con pérdidas). Aquí también ilustraremos el concepto con tres ejemplos. Nuestro primer ejemplo es la **codificación diferencial**, en la que una secuencia de valores (por ejemplo, muestras de audio) se codifican representando cada uno como la diferencia respecto al valor previo. La modulación por código de pulso diferencial, que vimos en el capítulo 2, es un ejemplo de esta técnica. Es con pérdidas porque la señal puede saltar tanto entre dos valores consecutivos que la diferencia no quepa en el campo proporcionado para expresar las diferencias, por lo que, cuando menos, se registrará un valor incorrecto y se perderá un poco de información.

La codificación diferencial es un tipo de codificación por fuente porque aprovecha la propiedad de que son poco probables los saltos grandes entre puntos de datos consecutivos. No todas las secuencias de números tienen esta propiedad. Un ejemplo en el que falta esta propiedad es una lista generada por computadora de números telefónicos aleatorios que será usada por una empresa de ventas por teléfono para molestar a la gente durante la comida. Las diferencias entre números telefónicos consecutivos de la lista requerirán tantos bits para representarse como los números mismos.

Nuestro segundo ejemplo de codificación por fuente consiste en transformaciones. Gracias a la transformación de señales de un dominio a otro, la compresión puede volverse mucho más fácil. Considere, por ejemplo, la transformación de Fourier de la figura 2-1(e). Aquí una función de tiempo se representa como una lista de amplitudes. Dados los valores exactos de todas las amplitudes, la función original puede reconstruirse perfectamente. Sin embargo, dados sólo los valores de, digamos, las primeras ocho amplitudes redondeadas a dos lugares decimales, aún puede ser posible reconstruir la señal tan bien que el escucha no pueda saber que se ha perdido un poco de información. La ganancia es que la transmisión de ocho amplitudes requiere muchos menos bits que la transmisión de la forma de onda muestreada.

Las transformaciones también se aplican a datos de imágenes de dos dimensiones. Supóngase que la matriz de 4×4 de la figura 7-78(a) representa los valores de la escala de grises de una imagen monocromática. Podemos transformar estos datos restando el valor de la esquina superior izquierda de todos los elementos excepto él mismo, como se muestra en la figura 7-78(b). Esta transformación podría ser útil si se usa codificación de longitud variable. Por ejemplo, los valores entre -7 y +7 podrían codificarse con números de 4 bits, y los valores entre 0 y 255 podrían codificarse como un código especial de 4 bits (-8) seguido de un número de 8 bits.

Figure 7-78 consists of two tables. Table (a) is a 4x4 grid of pixel values labeled 'Valor de pixel'. The values are: Row 1: 160, 160, 161, 160; Row 2: 161, 165, 166, 158; Row 3: 160, 167, 165, 161; Row 4: 159, 160, 160, 160. An arrow points to the value 160 with the label 'Valor de pixel'. Table (b) is the same grid with the top-left value 160 removed from each element. It is labeled '4 pixeles' at the top. The values are: Row 1: 0, 1, 0, 0; Row 2: 1, 5, 6, -2; Row 3: 0, 7, 5, 1; Row 4: -1, 0, 1, 0.

Valor de pixel			
160	160	161	160
161	165	166	158
160	167	165	161
159	160	160	160

4 pixeles			
0	1	0	0
1	5	6	-2
0	7	5	1
-1	0	1	0

Figura 7-78. (a) Valores de píxel para parte de una imagen. (b) Transformación en la que el elemento de la esquina superior izquierda se resta de todos los elementos menos él mismo.

Aunque esta transformación sencilla es sin pérdidas, otras, más útiles, no lo son. Una transformación espacial de dos dimensiones muy importante es la **DCT** (*Discrete Cosine Transformation*, transformación por coseno discreto) (Feig y Winograd, 1992). Esta transformación tiene la propiedad de que, con imágenes sin discontinuidades pronunciadas, la mayor parte de la capacidad espectral está en los primeros términos, permitiendo que los últimos se ignoren sin mucha pérdida de información. Regresaremos a la DCT pronto.

Nuestro tercer ejemplo de codificación por fuente es la **cuantización vectorial**, que también puede aplicarse directamente a los datos de imagen. Aquí, la imagen se divide en rectángulos de tamaño fijo. Además de la imagen misma, necesitamos una tabla de rectángulos del mismo tamaño que los rectángulos de imagen (probablemente construidos a partir de la imagen). Esta tabla se llama **libro de códigos**. Cada rectángulo se transmite buscándolo en el libro de códigos y simplemente mandando el índice en lugar del rectángulo. Si el libro de códigos se crea dinámicamente (es decir, por imagen), debe transmitirse también. Ciertamente, si una pequeña cantidad de rectángulos domina la imagen, son posibles grandes ahorros en ancho de banda.

Un ejemplo de cuantización vectorial se muestra en la figura 7-79. En la figura 7-79(a) tenemos una malla de rectángulos de tamaño no especificado. En la figura 7-79(b) tenemos el libro de códigos. La corriente de salida es sólo la lista de enteros 001022032200400 mostrada en la figura 7-79(c). Cada entero representa una entrada del libro de códigos.

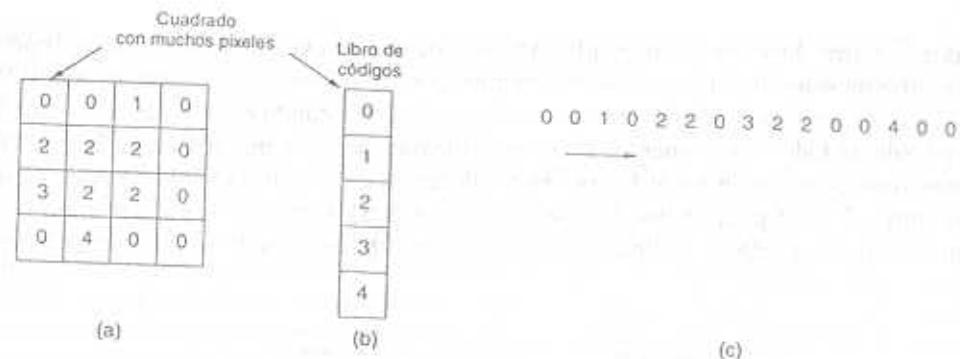


Figura 7-79. Ejemplo de cuantización vectorial. (a) Imagen dividida en marcos. (b) Libro de códigos de la imagen. (c) La imagen codificada.

En cierto sentido, la cuantización vectorial es sólo una generalización de dos dimensiones de la CLUT. Sin embargo, la verdadera diferencia es lo que ocurre si no se encuentra una equivalencia. Son posibles tres estrategias. La primera es simplemente usar la mejor equivalencia. La segunda es usar la mejor equivalencia y agregar alguna información sobre la manera de mejorar la equivalencia (por ejemplo, agregar el valor medio real). La tercera es usar la mejor equivalencia y agregar lo que sea necesario para permitir al decodificador la reconstrucción exacta de los datos. Las primeras dos estrategias tienen pérdidas, pero ofrecen alta compresión. La tercera no tiene pérdidas, pero es menos efectiva como algoritmo de compresión. De nuevo vemos que la codificación (equivalencia de patrones) consume mucho más tiempo que la decodificación (indización en una tabla).

Estándar JPEG

El estándar **JPEG** (*Joint Photographic Experts Group*, grupo conjunto de expertos en fotografía) para la compresión de imágenes fijas de tono continuo (por ejemplo, fotografías) fue desarrollado por expertos en fotografía trabajando con el auspicio conjunto del ITU, la ISO y el IEC, otro grupo de estándares. Es importante para multimedia porque, a primera vista, el estándar multimedia para imágenes en movimiento, MPEG, es simplemente la codificación JPEG de cada marco por separado, más algunas características extra para la compresión intermarcos y la detección de movimiento. El JPEG se define en el Estándar Internacional 10918.

El JPEG tiene cuatro modos y muchas opciones; se parece más a una lista de compras que a un solo algoritmo. No obstante, para nuestros fines sólo es relevante el modo secuencial con pérdidas, y éste se ilustra en la figura 7-80. Es más, nos concentraremos en la manera en que el JPEG se usa normalmente para codificar imágenes de vídeo RGB de 24 bits y dejaremos fuera algunos de los detalles menores por cuestiones de sencillez.

El paso 1 de la codificación de una imagen con el JPEG es la preparación del bloque. Para ser específicos, supongamos que la entrada JPEG es una imagen RGB de 640×480 con 24 bits/píxel, como se muestra en la figura 7-81(a). Puesto que el uso de luminancia y crominancia da



Figura 7-80. Operación del JPEG en el modo secuencial libre.

una mejor compresión, primero calcularemos la luminancia, Y , y las dos crominancias, I y Q (para NTSC), de acuerdo con las siguientes fórmulas:

$$Y = 0.30R + 0.59G + 0.11B$$

$$I = 0.60R - 0.28G - 0.32B$$

$$Q = 0.21R - 0.52G + 0.31B$$

En PAL, las crominancias se llaman U y V , y los coeficientes son diferentes, pero la idea es la misma. SECAM es diferente tanto de NTSC como de PAL.

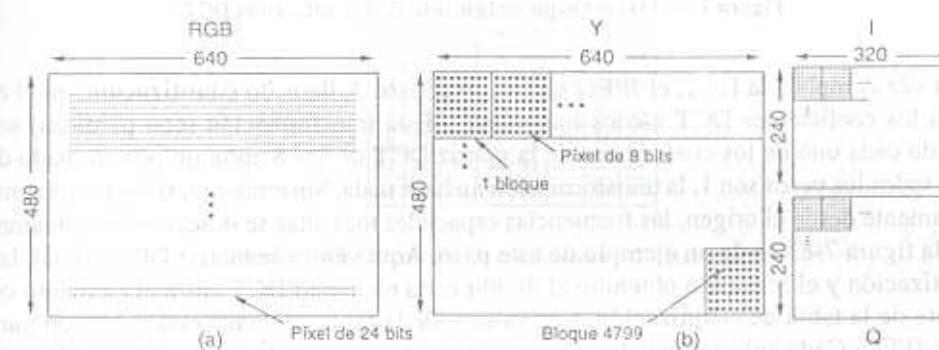


Figura 7-81. (a) Datos de entrada RGB. (b) Tras la preparación de bloques.

Se construyen matrices separadas para Y , I y Q , cada una con elementos en el intervalo de 0 a 255. A continuación se promedian marcos de cuatro píxeles en las matrices I y Q para reducirlos a 320×240 . Esta reducción tiene pérdidas, pero el ojo apenas lo nota, ya que responde a la luminancia más que a la crominancia; no obstante, comprime los datos en un factor de dos. Ahora se resta 128 a cada elemento de las tres matrices para poner el 0 a la mitad de la gama. Por último, cada matriz se divide en bloques de 8×8 . La matriz Y tiene 4800 bloques; las otras dos tienen 1200 bloques cada una, como se muestra en la figura 7-81(b).

El paso 2 del JPEG es aplicar individualmente una transformación coseno discreta (DCT) a cada uno de los 7200 bloques. La salida de cada DCT es una matriz 8×8 de coeficientes DCT. El elemento DCT (0, 0) es el valor medio del bloque. Los otros elementos indican la cantidad de potencia espectral que hay en cada frecuencia espacial. En teoría, una DCT no tiene pérdidas pero, en la práctica, el uso de números de punto flotante y funciones trascendentales siempre introducen

algún error de redondeo que resulta en una pequeña pérdida de información. Estos elementos normalmente decaen rápidamente al alejarse del origen $(0, 0)$, como se sugiere en la figura 7-82.

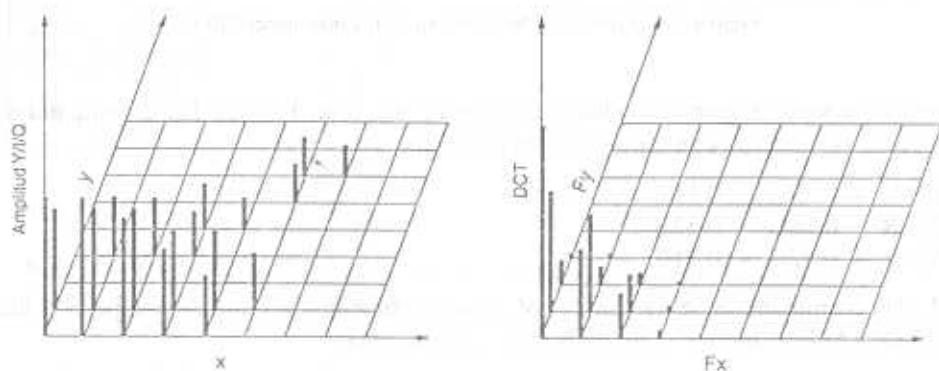


Figura 7-82. (a) Un bloque de la matriz Y . (b) Coeficientes DCT.

Una vez completa la DCT, el JPEG sigue con el paso 3, llamado cuantización, en el que se eliminan los coeficientes DCT menos importantes. Esta transformación (con pérdidas) se hace dividiendo cada uno de los coeficientes de la matriz DCT de 8×8 entre un peso tomado de una tabla. Si todos los pesos son 1, la transformación no hace nada. Sin embargo, si los pesos aumentan marcadamente desde el origen, las frecuencias espaciales más altas se descartarán rápidamente.

En la figura 7-83 se da un ejemplo de este paso. Aquí vemos la matriz DCT inicial, la tabla de cuantización y el resultado obtenido al dividir cada elemento DCT entre el elemento correspondiente de la tabla de cuantización. Los valores de la tabla de cuantización no son parte del estándar JPEG. Cada aplicación debe proporcionar sus propios valores, permitiéndole controlar el equilibrio pérdidas-compresión.

El paso cuatro reduce el valor $(0, 0)$ de cada bloque (el de la esquina superior izquierda) reemplazándolo por el valor de su diferencia respecto al elemento correspondiente del bloque previo. Dado que estos elementos son las medias de sus respectivos bloques, deben cambiar lentamente, por lo que al tomarse sus valores diferenciales se debe reducir la mayoría de ellos a valores pequeños. No se calculan diferenciales de los otros valores. Se llaman a los valores $(0, 0)$ componentes de cc; los otros son las componentes de ca.

El paso 5 hace lineales los 64 elementos y aplica codificación por longitud de serie a la lista. El barrido del bloque de izquierda a derecha y luego de arriba abajo no concentra los ceros, por lo que se usa un patrón de barrido en zigzag, como se muestra en la figura 7-84. En este ejemplo, el patrón de zigzag produce 38 ceros consecutivos el final de la matriz. Esta cadena puede reducirse a una sola cuenta diciendo que hay 38 ceros.

Ahora tenemos una lista de números que representan la imagen (en espacio de transformación). El paso 6 aplica codificación de Huffman a los números para su almacenamiento o transmisión.

Coeficientes DCT								Coeficientes cuantizados							
150	80	40	14	4	2	1	0	150	80	20	4	1	0	0	0
92	75	36	10	6	1	0	0	92	75	18	3	1	0	0	0
52	38	26	8	7	4	0	0	26	19	13	2	1	0	0	0
12	8	6	4	2	1	0	0	3	2	2	1	0	0	0	0
4	3	2	0	0	0	0	0	1	0	0	0	0	0	0	0
2	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla de cuantización

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Figura 7-83. Cálculo de los coeficientes DCT cuantizados.

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figura 7-84. Orden en que se transmiten los valores cuantizados.

El JPEG puede parecer complicado, pero eso es porque *es* complicado. Aun así, dado que con frecuencia produce una compresión de 20:1 o mejor, se usa ampliamente. La decodificación de una imagen JPEG ejecuta hacia atrás el algoritmo. A diferencia de algunos de los algoritmos de compresión que hemos visto, el JPEG es más o menos simétrico: la decodificación tarda tanto como la codificación.

Como detalle interesante, debido a las propiedades matemáticas de la DCT, es posible ejecutar ciertas transformaciones (por ejemplo, rotación de imagen) sobre la matriz transformada sin regenerar la imagen original. Estas transformaciones se estudian en (Shen y Sethi, 1995). También se aplican propiedades semejantes al audio comprimido MPEG (Broadhead y Owen, 1995).

Estándar MPEG

Por último, llegamos al corazón del asunto: los estándares MPEG (*Motion Picture Experts Group*, grupo de expertos en imágenes en movimiento). Éstos son los algoritmos principales usados para comprimir videos y han sido estándares internacionales desde 1993. Puesto que las películas contienen tanto imágenes como sonido, el MPEG puede comprimir tanto audio como video pero, dado que el video usa más ancho de banda y también contiene mayor redundancia que el audio, nos enfocaremos principalmente en la compresión de video MPEG en lo que sigue.

El primer estándar terminado fue el MPEG-1 (Estándar Internacional 11172); su meta fue producir salida con calidad de videograbadora (352×240 para NTSC) usando una tasa de bits de 1.2 Mbps. Puesto que, como vimos antes, el video no comprimido por sí solo puede requerir 472 Mbps, reducirlo a 1.2 Mbps no es nada trivial, aun con esta menor definición. El MPEG-1 puede transmitirse por líneas de transmisión de par trenzado a distancias modestas. El MPEG-1 también se usa para almacenar películas en CD-ROM, en formatos CD-I y CD-Video.

El siguiente estándar de la familia MPEG fue el MPEG-2 (estándar internacional 13818), que se diseñó originalmente para comprimir video con calidad de difusión a 4-6 Mbps, de modo que pudiera caber en un canal de difusión NTSC o PAL. Después se expandió el MPEG-2 para manejar definiciones mayores, incluida HDTV. El MPEG-4 es para videoconferencias de mediana definición con tasas de marco bajas (10 marcos/seg) y a bajo ancho de banda (64 kbps). Esto permite sostener videoconferencias por un solo canal N-ISDN B. Dada la numeración, podría pensarse que el siguiente estándar será el MPEG-8. En realidad, la ISO está numerando los linealmente, no exponencialmente. Originalmente existió el MPEG-3, que estaba dirigido a la HDTV, pero ese proyecto luego se canceló, y la HDTV se agregó al MPEG-2.

Los principios básicos del MPEG-1 y el MPEG-2 son parecidos, pero los detalles son diferentes. A primera vista, el MPEG-2 es un supergrupo del MPEG-1, con características, formatos de marco y opciones de codificación adicionales. Es probable que a la larga el MPEG-1 dominará en las películas en CD-ROM y el MPEG-2 dominará en la transmisión de video a largas distancias. Estudiaremos el MPEG-1 primero y el MPEG-2 después.

El MPEG-1 tiene tres partes: audio, video y sistema, que integra los otros dos, como se muestra en la figura 7-85. Los codificadores de audio y video funcionan independientemente, lo

que hace surgir la cuestión de cómo se sincronizan las dos cadenas en el receptor. Este problema se resuelve teniendo un reloj de sistema de 90 kHz que suministra el valor de tiempo a ambos codificadores. Estos valores son de 33 bits, para permitir que las películas duren 24 horas sin dar la vuelta. Estas marcas de tiempo se incluyen en la salida codificada y se propagan hasta el receptor, que puede usarlas para sincronizar las corrientes de audio y vídeo.



Figura 7-85. Sincronización de las corrientes de audio y video en el MPEG-1.

La compresión de audio MPEG se hace muestreando la forma de onda a 32 kHz, 44.1 kHz o 48 kHz. Se puede manejar sonido monofónico, estéreo disjunto (cada canal comprimido por separado) o estéreo combinado (explotación de la redundancia intercanal). El algoritmo se organiza en tres capas, cada una de las cuales aplica optimizaciones adicionales para obtener mayor compresión (a mayor costo). La capa 1 es el esquema básico. Esta capa se usa, por ejemplo, en el sistema de cinta digital DCC. La capa 2 agrega asignación de bits avanzada al esquema básico. Se usa para audio de CD-ROM y pistas sonoras de cine. La capa 3 agrega filtros híbridos, cuantización no uniforme, codificación Huffman y otras técnicas avanzadas.

El MPEG puede comprimir un CD de *rock and roll* a 96 kbps sin pérdidas perceptibles en la calidad del audio, aun para fanáticos del *rock* que aún no han sufrido pérdidas auditivas. Para un concierto de piano se requieren cuando menos 128 kbps. Son diferentes porque la relación señal a ruido del *rock and roll* es mucho mayor que en un concierto de piano (desde el punto de vista de la ingeniería, al menos).

La compresión de audio se logra ejecutando una transformación de Fourier rápida con la señal de audio para transformarla del dominio del tiempo al dominio de la frecuencia. El espectro resultante se divide entonces en 32 bandas de frecuencia, que se procesan por separado. Cuando hay presentes dos canales estéreo, también puede aprovecharse la redundancia inherente a tener dos fuentes de audio altamente traslapadas. La corriente de audio MPEG-1 resultante se puede ajustar desde 32 kbps hasta 448 kbps. En (Pan, 1995) se presenta una introducción al proceso.

Consideremos ahora la compresión de video MPEG-1. Existen dos clases de redundancia en las películas: espacial y temporal. El MPEG-1 aprovecha ambas. La redundancia espacial puede utilizarse simplemente codificando por separado cada marco mediante JPEG. Este enfoque se ocasiona ocasionalmente, sobre todo cuando se requiere acceso aleatorio a cada marco, como en la

edición de producciones de vídeo. En este modo, se puede lograr un ancho de banda comprimido del orden de 8 a 10 Mbps.

Puede lograrse compresión adicional aprovechando el hecho de que los marcos consecutivos a menudo son casi idénticos. Este efecto es menor que lo que podría pensarse, puesto que muchos cineastas hacen cortes entre escenas cada 3 o 4 segundos (tome el tiempo de una película y cuente las escenas). No obstante, incluso una serie de 75 marcos muy parecidos ofrece el potencial de una reducción importante respecto a la simple codificación de cada marco por separado mediante JPEG.

En las escenas en las que la cámara y el fondo son estacionarios y uno o dos actores se mueven con lentitud, casi todos los pixeles serán idénticos de un marco a otro. Aquí será suficiente la simple resta de cada marco del anterior y la aplicación de JPEG a la diferencia. Sin embargo, en las escenas en las que la cámara hace una panorámica o un acercamiento, esta técnica falla gravemente. Lo que se necesita es una manera de compensar este movimiento. Esto es precisamente lo que hace el MPEG, y es la diferencia principal entre el MPEG y el JPEG.

La salida del MPEG-1 consiste en cuatro tipos de marco:

1. Marcos I (intracodificados): imágenes fijas autocontenidoas codificadas en JPEG.
2. Marcos P (predictivos): diferencia bloque por bloque con el marco anterior.
3. Marcos B (bidireccionales): diferencias con el marco anterior y el siguiente.
4. Marcos D (codificación CD): promedios de bloque usados para avance rápido.

Los marcos I son simplemente imágenes fijas codificadas con JPEG, usando también luminancia de definición completa y crominancia de definición media sobre cada eje. Es necesario hacer que los marcos I aparezcan periódicamente en la corriente de salida por tres razones. Primera, el MPEG-1 puede usarse para multitransmisión, sintonizándose los usuarios a voluntad. Si todos los marcos dependieran de sus antecesores remontándose al primer marco, cualquiera que no recibiera el primer marco no podría decodificar nunca los marcos subsiguientes. Segunda, si un marco se recibiera con error, no sería posible ninguna decodificación posterior. Tercera, sin marcos I, al hacer un avance o retroceso rápido, el decodificador tendría que calcular cada marco por el que pasa para conocer el valor completo de aquel en el que se detiene. Por estas tres razones, se insertan marcos I en la salida una o dos veces por segundo.

Los marcos P, en contraste, codifican las diferencias entre marcos; se basan en la idea de los **macrobloques**, que cubren 16×16 pixeles de espacio de luminancia y 8×8 pixeles de espacio de crominancia. Un macrobloque se codifica buscando en el marco previo algo igual a o ligeramente diferente de él.

Un ejemplo de caso en el que serían útiles los marcos P se ve en la figura 7-86. Aquí vemos tres marcos consecutivos que tienen el mismo fondo, pero en los que cambia la posición de una persona. Los macrobloques que contienen el fondo serán exactamente iguales, pero los macrobloques que contienen la persona estarán desfasados en alguna cantidad desconocida y tendrán que rastrearse.



Figura 7-86. Tres marcos consecutivos.

El estándar MPEG-1 no especifica la manera de efectuar la búsqueda, ni qué tan profunda o buena debe ser para que sirva. Éstas son decisiones de cada implementación. Por ejemplo, una implementación podría buscar un macrobloque en la posición actual, pero del marco anterior, y con todas las demás posiciones desplazadas $\pm \Delta x$ en la dirección x y $\pm \Delta y$ en la dirección y. Para cada posición, podría calcularse la cantidad de equivalencias en la matriz de luminancia. La posición con el puntaje más alto se declararía ganadora, siempre que estuviera por encima de un umbral predefinido. En caso contrario, se diría que falta el macrobloque. Por supuesto, pueden usarse algoritmos mucho más refinados.

Si se encuentra un macrobloque, se codifica tomando la diferencia respecto a su valor en el marco previo (para la luminancia y ambas crominancias). Estas matrices de diferencias son el objeto de la transformación por coseno discreto, cuantización, codificación por longitud de serie y codificación Huffman, al igual que en el JPEG. El valor del macrobloque en la cadena de salida es entonces el vector de movimiento (la distancia que se movió el macrobloque de su posición previa en cada sentido), seguido de la lista de codificación Huffman de los números. Si el macrobloque no se encuentra en el marco previo, se codifica el valor actual con JPEG, igual que en un marco I.

Es claro que este algoritmo es altamente asimétrico. Las implementaciones están en libertad de intentar todas las posiciones del marco previo si lo desean, en un intento desesperado por localizar todos los macrobloques previos. Este enfoque reducirá al mínimo la cadena MPEG-1 codificada al costo de una codificación muy lenta. Este enfoque podría estar bien para la codificación de una sola vez de una cineteca, pero sería terrible para las videoconferencias en tiempo real.

De la misma manera, cada implementación está en libertad de decidir lo que constituye un macrobloque "encontrado". Esta libertad permite a los implementadores competir según la calidad y velocidad de sus algoritmos, pero siempre produce MPEG-1 conformante. Sea cual sea el algoritmo de búsqueda usado, la salida final es la codificación JPEG del macrobloque actual, o la codificación JPEG de la diferencia entre el macrobloque actual y el del marco previo, con un desplazamiento especificado respecto al marco actual.

Hasta ahora, la decodificación del MPEG-1 es directa. La decodificación de marcos I es igual a la decodificación de marcos JPEG. La decodificación de marcos P requiere que el decodificador maneje en *buffer* el marco previo, y luego construya el nuevo en un segundo *buffer*

con base en macrobloques completamente codificados y macrobloques que contienen diferencias respecto al marco previo. El nuevo marco se ensambla macrobloque por macrobloque.

Los marcos B son parecidos a los marcos P, excepto que permiten que el macrobloque de referencia esté en un marco previo o en un marco posterior. Esta libertad adicional permite mejorar la compensación del movimiento y es útil también cuando pasan objetos por delante o detrás de otros objetos. Para ejecutar la codificación de marcos B, el codificador necesita tener a la vez tres marcos decodificados en la memoria: el anterior, el actual y el siguiente. Aunque los marcos B producen la mejor compresión, no todas las implementaciones lo reconocen.

Los marcos D sólo se usan para visualizar una imagen de baja definición al hacer un reembobinado o avance rápido. Hacer la decodificación MPEG-1 normal en tiempo real ya es bastante difícil. Esperar que el decodificador lo haga mientras trabaja a 10 veces su velocidad normal es pedir demasiado. Cada entrada de marco D simplemente es el valor promedio de un bloque, sin mayor codificación, simplificando la presentación en tiempo real. Este mecanismo es importante para permitir que la gente barra un video a alta velocidad en busca de una escena en particular.

Habiendo terminado nuestro tratamiento del MPEG-1, pasemos al MPEG-2. La codificación MPEG-2 es parecida en lo fundamental a la codificación MPEG-1, con marcos I, marcos P y marcos B. Sin embargo, no se reconocen los marcos D. También, la transformación coseno es de 10×10 en lugar de 8×8 , para dar 50% más de coeficientes y, por tanto, mayor calidad. Puesto que el MPEG-2 está dirigido a la televisión difundida al igual que a las aplicaciones en CD-ROM, reconoce imágenes tanto progresivas como entrelazadas, mientras que el MPEG-1 reconoce sólo las imágenes progresivas. También son diferentes otros detalles menores entre los dos estándares.

En lugar de reconocer un solo nivel de definición, el MPEG-2 reconoce cuatro: baja (352×240), principal (720×480), alta-1440 (1440×1152) y alta (1920×1080). La baja definición es para videogramadoras y compatibilidad hacia abajo con el MPEG-1. La principal es la normal para difusión NTSC. Las otras dos son para HDTV.

Además de tener cuatro niveles de definición, el MPEG-2 también maneja cinco perfiles. Cada perfil está pensado para algún área de aplicación. El perfil principal es para propósito general, y probablemente la mayoría de los chips estará optimizada para el perfil principal y el nivel de definición principal. El perfil sencillo es parecido al principal, excepto que excluye el uso de marcos B para simplificar el software de codificación y decodificación. Los otros perfiles tienen que ver con la escalabilidad y la HDTV. Los perfiles son diferentes en términos de la presencia o ausencia de marcos B, definición de crominancia y escalabilidad de la cadena de bits codificada a otros formatos.

La tasa de datos comprimidos de cada combinación de definición y perfil es diferente; va desde unos 3 Mbps hasta 100 Mbps para HDTV. El caso normal es de unos 3 a 4 Mbps. En (Pancha y El Zarki, 1994) se dan algunos datos del desempeño de MPEG.

El MPEG-2 tiene una manera más general de multiplexar audio y video que el modelo MPEG-1 de la figura 7-85; define una cantidad ilimitada de corrientes elementales, incluidos video y audio, pero también corrientes de datos que deben sincronizarse con el audio y el video;

por ejemplo, subtítulos en varios idiomas. Cada una de las corrientes se empaca primero con marcas de tiempo. Se muestra un ejemplo sencillo de dos corrientes en la figura 7-87.

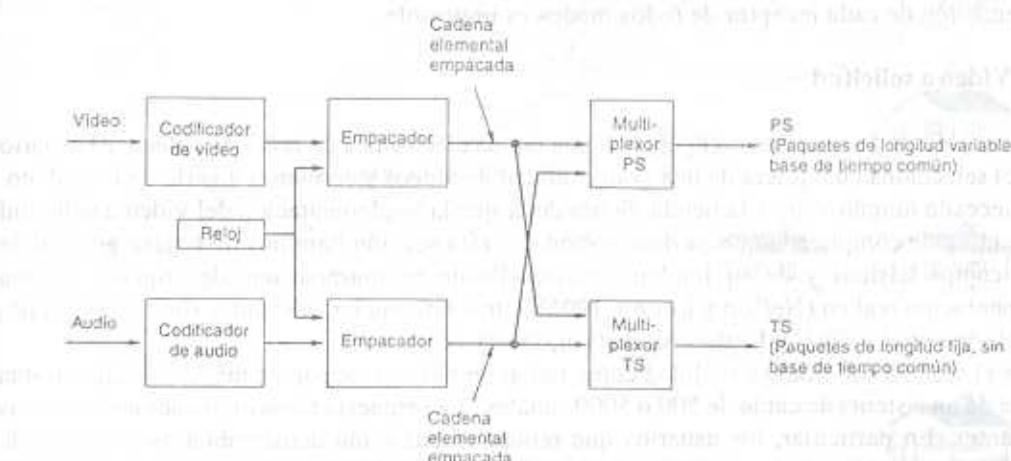


Figura 7-87. Multiplexión de dos corrientes en MPEG-2.

La salida de cada empacador es una PES (Packetized Elementary Stream, corriente elemental empacada). Cada PES tiene unos 30 campos de cabecera e indicadores, incluidas longitudes, identificadores de corriente, control de cifrado, estado de derechos de autor, marcas de tiempo y un CRC.

Las corrientes PES de audio, video y posiblemente datos se multiplexan juntas en una sola corriente de salida para su transmisión. Se definen dos tipos de corriente. La corriente de programa MPEG-2 es parecida a la corriente de los sistemas MPEG-1 de la figura 7-85; se usa para multiplexar varias corrientes elementales que tienen una base común y tienen que presentarse de una manera sincronizada. La corriente de programa usa paquetes grandes de longitud variable.

La otra corriente MPEG-2 es la corriente de transporte. Se usa para multiplexar corrientes (incluidas corrientes de programa) que no usan una base de tiempo común. Los paquetes de corriente de transporte son de longitud fija (188 bytes), para simplificar la limitación del efecto de los paquetes dañados o perdidos durante la transmisión.

Vale la pena indicar que todos los esquemas de codificación que hemos estudiado se basan en el modelo de codificación con pérdidas seguida de transmisión sin pérdidas. Ni el JPEG ni el MPEG pueden recuperarse de paquetes perdidos o dañados sin que se note el problema. Un enfoque diferente para la transmisión de imágenes es transformar éstas de manera que se separe la información importante de la menos importante (como la DCT, por ejemplo). Luego se agrega

una cantidad considerable de redundancia (incluso paquetes duplicados) a la información importante y ninguna a la menos importante. Si se pierden o se alteran algunos paquetes, aún puede ser posible presentar imágenes razonables sin retransmitir. Estas ideas se describen con mayor detalle en (Danskin *et al.*, 1995), y se aplican especialmente a la multitransmisión, en la que la realimentación de cada receptor de todos modos es imposible.

7.7.4. Vídeo a solicitud

El vídeo a solicitud a veces se compara con una tienda electrónica de renta de videos. El usuario (cliente) selecciona cualquiera de una gran cantidad de videos y comienza a verlo de inmediato. No se necesita ningún viaje a la tienda. Sobra decir que la implementación del vídeo a solicitud es un tanto más complicada que su descripción. En esta sección haremos un repaso general de los conceptos básicos y de su implementación. Puede encontrarse una descripción de una implementación real en (Nelson y Linton, 1995). Otras referencias relevantes son (Chang *et al.*, 1994; Hodge *et al.*, 1993, y Little y Venkatesh, 1994).

¿Es el vídeo a solicitud en realidad como rentar un video, o se parece más a seleccionar una película de un sistema de cable de 500 o 5000 canales? La respuesta tiene implicaciones técnicas importantes. En particular, los usuarios que rentan videos están acostumbrados a la idea de poder detener el video, hacer un viaje rápido a la cocina o al baño, y luego continuar a partir de donde detuvieron el video. Los espectadores de televisión no esperan poner los programas en pausa.

Si el vídeo a solicitud va a competir con éxito contra las tiendas de renta de video, puede ser necesario que los usuarios detengan, inicien y rebobinen los videos a voluntad. Para ello, el proveedor de videos tendrá que transmitir una copia individual a cada quien.

Por otra parte, si el vídeo a solicitud se considera más como una televisión avanzada, entonces puede ser suficiente hacer que el proveedor de video inicie cada video popular, digamos, cada 10 minutos, y luego lo exhiba sin parar. Un usuario que desee ver un video popular podría tener que esperar hasta 10 minutos para que comience. Aunque no es posible la pausa/reinicio aquí, un espectador que regrese a la sala tras una interrupción corta puede pasar a otro canal que muestre el mismo video, pero con 10 minutos de retraso. Una parte del material se repetirá, pero no se perderá nada. Este esquema se llama **casi video a solicitud**, y ofrece la posibilidad de un costo mucho menor, puesto que la misma alimentación del servidor de video puede llegar a muchos usuarios al mismo tiempo. La diferencia entre video a solicitud y casi video a solicitud es parecida a la diferencia entre manejar su propio auto y tomar el autobús.

Ver películas (casi) a solicitud es una de una gran gama de servicios nuevos que podrían hacerse realidad una vez que estén disponibles las redes de ancho de banda amplio. El modelo general que usa mucha gente se ilustra en la figura 7-88. Aquí vemos una red de *backbone* de área amplia (nacional o internacional) de alto ancho de banda como centro del sistema. Conectadas a ella hay miles de redes de distribución locales, como TV por cable o sistemas de distribución telefónica. Los sistemas de distribución locales llegan hasta las casas de la gente, donde terminan en **cajas de control**, que de hecho son potentes computadoras personales especializadas.

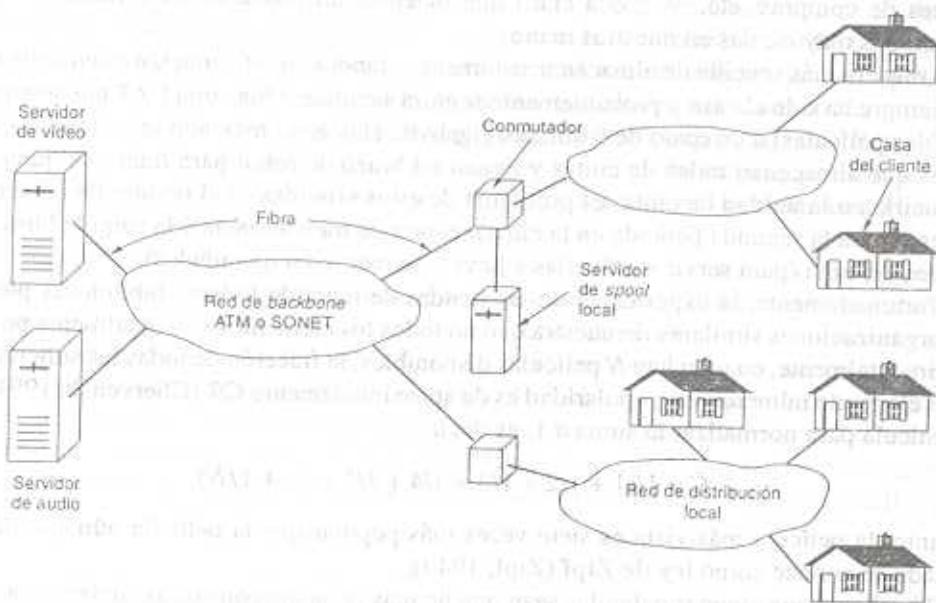


Figura 7-88. Esquema general de un sistema de video a solicitud.

Conectados al *backbone* mediante fibras ópticas de alto ancho de banda hay miles de proveedores de información. Algunos de estos ofrecerán video o audio de pago por evento. Otros ofrecerán servicios especializados, como compras desde casa (con la capacidad de girar una lata de sopa y ver la lista de ingredientes, o ver un fragmento de video sobre el modo de operación de una podadora operada por gasolina). Si duda, pronto habrá la disponibilidad de deportes, noticias, capítulos viejos del "Chapulín Colorado", acceso a la WWW e innumerables otras posibilidades.

También incluidos en el sistema hay servidores *se spool* de E/S locales que permitirán acercar los videos al usuario, a fin de ahorrar ancho de banda durante las horas pico. La manera en que encajarán estas partes y quién será el dueño de qué son temas de intenso debate en la industria. A continuación examinaremos el diseño de las partes principales del sistema: los servidores de video, la red de distribución y las cajas de control.

Servidores de video

Para tener (casi) video a solicitud, necesitamos servidores de video capaces de almacenar y sacar una gran cantidad de películas simultáneamente. La cantidad total de películas que se han filmado se estima en 65,000 (Minoli, 1995). Al comprimirse con MPEG-2, una película normal ocupa unos 4 GB de almacenamiento, por lo que 65,000 de ellas requerirán unos 260 terabytes. Sume a esto todos los programas de televisión, filmaciones de deportes, noticieros, catálogos

parlantes de compras, etc., y queda claro que tenemos un problema de almacenamiento de proporciones mayúsculas en nuestras manos.

La manera más sencilla de almacenar volúmenes grandes de información es en cinta magnética. Siempre ha sido el caso y probablemente seguirá siéndolo. Una cinta DAT puede almacenar 8 GB (dos películas) a un costo de 5 dólares/gigabyte. Hay en el mercado servidores mecánicos grandes que almacenan miles de cintas y tienen un brazo de robot para traer cualquier cinta e introducirla en la unidad de cinta. El problema de estos sistemas es el tiempo de acceso (especialmente para la segunda película en la cinta), la tasa de transferencia y la cantidad limitada de unidades de cinta (para servir n películas a la vez, se requieren n unidades).

Afortunadamente, la experiencia de las tiendas de renta de videos, bibliotecas públicas y otras organizaciones similares demuestra que no todos los elementos son igualmente populares. Experimentalmente, cuando hay N películas disponibles, la fracción de todas las solicitudes que pide el elemento número k en popularidad es de aproximadamente C/k (Chervenak, 1994). Aquí, C se calcula para normalizar la suma a 1, es decir

$$C = 1/(1 + 1/2 + 1/3 + 1/4 + 1/5 + \dots + 1/N)$$

Por tanto, la película más vista es siete veces más popular que la película número siete. Este resultado se conoce como ley de Zipf (Zipf, 1949).

El hecho de que algunas películas sean mucho más populares que otras sugiere una solución posible en forma de jerarquía de almacenamiento, como se muestra en la figura 7-89. Aquí, el desempeño aumenta a medida que se sube en la jerarquía.



Figura 7-89. Jerarquía de almacenamiento de un servidor de video.

Una alternativa a la cinta es el almacenamiento óptico. Los CD-ROM actuales contienen sólo 650 MB, pero la siguiente generación podrá contener unos 4 GB, lo que los hará adecuados para la distribución de películas MPEG-2. Aunque los tiempos de búsqueda son lentos en comparación con los discos magnéticos (100 msec contra 10 msec), su bajo costo y alta confiabilidad hace que los servidores ópticos con miles de CD-ROM sean una buena alternativa a la cinta para las películas de mayor uso.

Siguen los discos magnéticos. Éstos tienen tiempos de acceso cortos (10 msec), altas tasas de transferencia (10 MB/seg) y capacidades sustanciales (10 GB), lo que los hace adecuados para guardar películas que en realidad se están transmitiendo (en oposición a simplemente estar almacenadas en caso de que alguien las quiera). Su desventaja principal es el alto costo de almacenar películas a las que pocas veces se accede.

Hasta arriba de la pirámide de la figura 7-89 está la RAM. La RAM es el medio de almacenamiento más rápido, pero también el más caro; es el más adecuado para películas de las que se están enviando partes distintas a destinos diferentes al mismo tiempo (por ejemplo, vídeo a solicitud a 100 usuarios que comenzaron en momentos diferentes). Cuando los precios de la RAM caigan a 10 dólares por megabyte, una película de 4 GB ocupará 40,000 dólares de RAM, por lo que tener 100 películas en RAM costará 4 millones de dólares por los 400 GB de memoria. Aun así, para un servidor de vídeo de 10 millones de dólares, este costo bien podría valer la pena si cada película tiene suficientes clientes simultáneos.

Dado que un servidor de vídeo en realidad es sólo un dispositivo masivo de E/S en tiempo real, necesita una arquitectura de *hardware* y *software* diferente de la de una PC o estación de trabajo UNIX. La arquitectura del *hardware* de un servidor de video típico se ilustra en la figura 7-90. El servidor tiene una o más CPU RISC de alto desempeño, cada una con un poco de memoria local, una memoria principal compartida, un caché de RAM masivo para películas populares, una variedad de dispositivos de almacenamiento para guardar las películas, y algún *hardware* de red, normalmente una interfaz óptica con una red ATM (o SONET) a OC-3 o mayor. Estos subsistemas se conectan mediante un *bus* de velocidad extremadamente alta (cuando menos 1 GB/seg).

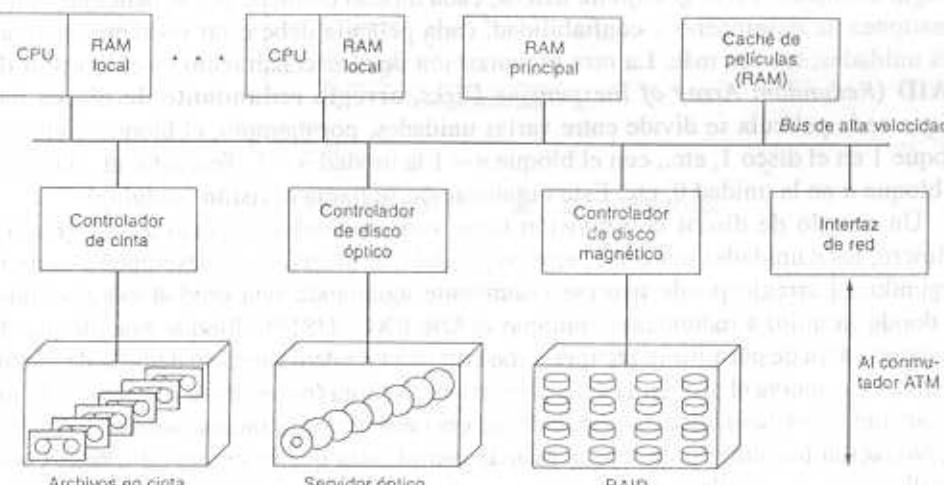


Figura 7-90. Arquitectura del *hardware* de un servidor de video típico.

Ahora veamos brevemente el *software* de un servidor de video. Las CPU se usan para aceptar solicitudes de los usuarios, localizar películas, mover datos entre dispositivos, facturar al cliente y muchas funciones más. Algunas de éstas no son de sincronización crítica, pero muchas otras sí, por lo que algunas de las CPU, si no todas, tendrán que ejecutar un sistema operativo de tiempo real, como un micronúcleo de tiempo real. Estos sistemas normalmente dividen el trabajo en tareas pequeñas, con un plazo de terminación conocido. El planificador puede ejecu-

tar entonces un algoritmo del tipo del siguiente plazo de terminación más cercano o el algoritmo de tasa monótonica (Liu y Layland, 1973).

El *software* de la CPU también define la naturaleza de la interfaz que el servidor presenta a los clientes (servidores de *spool* o cajas de control colocadas sobre el televisor). Son comunes dos diseños. El primero es un sistema tradicional de archivos, en el que los clientes pueden abrir, leer, escribir y cerrar archivos. Aparte de las complicaciones generadas por la jerarquía de almacenamiento y las consideraciones de tiempo real, uno de tales servidores puede tener un sistema de archivos modelado según el de UNIX.

El segundo tipo de interfaz se basa en el modelo de la videograbadora. Los comandos al servidor solicitan que abra, ejecute, haga pausa, avance rápido y rebobine archivos. La diferencia con el modelo UNIX es que, una vez que se ha emitido un comando *PLAY*, el servidor simplemente sigue sacando datos a velocidad constante, sin requerirse comandos nuevos. El corazón del *software* del servidor de video es el *software* de administración de disco, que tiene dos tareas principales: colocar películas en el disco magnético cuando tienen que traerse de almacenamiento óptico o cinta, y manejar solicitudes de disco para las muchas corrientes de salida. La colocación de las películas es importante, puesto que puede afectar en gran medida el desempeño.

Dos posibles maneras de organizar el almacenamiento en disco son la granja de discos y el arreglo de discos. En la granja de discos, cada unidad contiene pocas películas completas. Por cuestiones de desempeño y confiabilidad, cada película debe estar presente en cuando menos dos unidades, tal vez más. La otra organización de almacenamiento es el arreglo de discos o RAID (*Redundant Array of Inexpensive Disks*, arreglo redundante de discos baratos), en el que cada película se divide entre varias unidades, por ejemplo, el bloque 0 en el disco 0, el bloque 1 en el disco 1, etc., con el bloque $n - 1$ la unidad $n - 1$. Tras esto, el ciclo se repite, con el bloque n en la unidad 0, etc. Esta organización se llama división (*striping*).

Un arreglo de discos con división tiene varias ventajas respecto a una granja de discos. Primero, las n unidades pueden operar en paralelo, aumentando el desempeño en un factor de n . Segundo, el arreglo puede hacerse redundante agregando una unidad extra a cada grupo de n , donde la unidad redundante contiene el OR EXCLUSIVO bloque por bloque de las otras unidades, a fin de permitir la recuperación completa de datos en caso de falla de una unidad. Por último, se resuelve el problema del equilibrio de la carga (no se requiere colocación manual para evitar que todas las películas populares se encuentren en la misma unidad). Por otra parte, la organización por arreglo de discos es más complicada que la granja de discos y es altamente sensible a diversas fallas; también es poco adecuada para operaciones de videograbadora como rebobinado y avance rápido de una película. Un estudio de simulación que compara las dos organizaciones se da en (Chervenak *et al.*, 1995).

Muy relacionada con la colocación de bloques está la localización de bloques de disco. El esquema UNIX de tener un árbol desbalanceado de bloques de disco a los que se apunta mediante inodos generalmente es inaceptable, puesto que los archivos de video son enormes, por lo que la mayoría de los bloques sólo puede localizarse a través de un bloque indirecto triple, lo que significa muchos accesos a disco extra (Tanenbaum, 1992). En vez de ello, es común vincular los bloques en una lista de vínculo sencillo o doble. A veces se usa un índice (inodo) tipo UNIX para permitir acceso aleatorio.

La otra tarea del *software* del disco es dar servicio a todas las corrientes de salida en tiempo real y cumplir sus restricciones de temporización. Una corriente de video MPEG-2 a 25 marcos/seg requiere obtener y transmitir unos 14 kB cada 40 mseg, pero la cantidad real varía considerablemente dado que los marcos I, P y B tienen diferentes relaciones de compresión. En consecuencia, para mantener una tasa de salida uniforme, se requieren buffers en ambas terminales de la corriente.

En la figura 7-91 vemos una escalera que muestra la cantidad total de datos leídos del disco para una corriente de video dada (suponiendo que la película está en disco). Dicha cantidad aumenta en saltos discretos, un salto por cada bloque leído. Sin embargo, la transmisión debe ocurrir a una tasa más uniforme, de modo que el proceso de lectura debe mantenerse adelantado respecto al proceso de transmisión. El área sombreada de la figura indica datos que se han obtenido del disco pero aún no se transmiten.

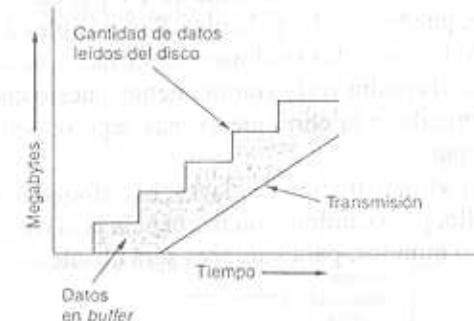


Figura 7-91. Proceso de buffer del disco en el servidor.

Los discos normalmente se programan usando el algoritmo del elevador, que comienza el movimiento del brazo hacia adentro y sigue avanzándolo hasta llegar al cilindro más interno, procesando todas las solicitudes que puede cumplir por orden de cilindro. Al llegar tan lejos como puede, el brazo invierte su sentido y comienza a moverse hacia afuera, procesando de nuevo en orden todas las solicitudes pendientes que encuentra en el camino. Aunque este algoritmo reduce el tiempo de búsqueda, no ofrece garantías en cuanto al desempeño en tiempo real, por lo que no es útil para un servidor de video.

Un mejor algoritmo es llevar el registro de todas las corrientes de video y hacer una lista del siguiente bloque que cada una necesita. Entonces, estos números de bloque se ordenan y se leen los bloques en orden por cilindro. Al leerse el último bloque, la siguiente ronda comienza con la obtención del número del bloque que ahora está a la cabeza de cada corriente. Estos números se ordenan y se leen en orden por cilindro, continuando así el proceso. Este algoritmo mantiene un desempeño de tiempo real para todos los cilindros, pero también reduce al mínimo el tiempo de búsqueda en comparación con un algoritmo puro de servicio según orden de aparición.

Otro problema de *software* es el control de admisión. Si entra una solicitud para una corriente nueva, ¿puede aceptarse sin arruinar el desempeño de tiempo real de las corrientes existentes? Un

algoritmo que puede servir para tomar una decisión examina el peor caso para ver si se garantiza que sea posible ir de k a $k + 1$ corrientes, con base en las propiedades conocidas de la CPU, la RAM y el disco. Otro algoritmo simplemente examina todas las propiedades estadísticas.

Otro asunto importante relacionado con el *software* del servidor es la manera de manejar la presentación en pantalla durante un avance o retroceso rápido (para que la gente pueda hacer búsquedas visuales). Los marcos D proporcionan la información necesaria para el MPEG-1 pero, a menos que se marquen y se almacenen de alguna manera especial, el servidor no podrá encontrarlos sin decodificar la corriente completa y, normalmente, los servidores no llevan a cabo decodificación MPEG durante la transmisión. En el MPEG-2 se requerirá algún otro mecanismo, cuando menos para facilitar el encuentro y decodificación de los marcos I.

Por último, el cifrado también es un asunto importante. Al multitransmitir películas (por ejemplo, si la red de distribución local es un sistema de TV por cable), se requiere cifrado para asegurar que sólo los clientes puedan ver las películas. Son posibles dos enfoques: el precifrado y el cifrado en el momento. Si las películas se almacenan cifradas, entonces cualquiera que logre averiguar la clave de una película podrá verla gratuitamente, puesto que siempre se usa la misma clave. El cifrado independiente de cada corriente es más seguro, pero también más costoso en términos de recursos de cómputo.

También es importante la administración de claves. El enfoque normal es cifrar al momento mediante un algoritmo sencillo, pero cambiar con frecuencia la clave, de modo que, si un intruso puede descifrar la clave en 10 minutos, para entonces será obsoleta.

La red de distribución

La red de distribución es el grupo de conmutadores y líneas entre el origen y el destino. Como vimos en la figura 7-88, esta red consiste en un *backbone* SONET o ATM (o ATM sobre SONET), conectado a la red local de distribución. Generalmente el *backbone* es conmutado y no la red de distribución local.

Los requisitos principales impuestos al *backbone* son alto ancho de banda y baja fluctuación. En un *backbone* SONET puro, es sencillo lograr esto: el ancho de banda está garantizado y la fluctuación es cero, puesto que la red es síncrona. En un *backbone* ATM, o ATM sobre SONET, la calidad del servicio es muy importante; se controla mediante el algoritmo de cubeta con fugas y todas las demás técnicas que estudiamos con detalle en el capítulo 5, por lo que no repetiremos ese análisis aquí. Para información adicional sobre el MPEG en tiempo real por *backbones* ATM, véase (Dixit y Skelly, 1995, y Morales *et al.*, 1995). A continuación nos enfocaremos en la red de distribución local, asunto que apenas hemos tocado hasta ahora.

La distribución local es caótica, pues las diferentes compañías usan redes distintas en diferentes regiones. Las compañías telefónicas, las de televisión por cable y los nuevos participantes están convencidos de que el que llegue primero será el gran ganador, por lo que ahora estamos viendo una proliferación en la instalación de nuevas tecnologías. Los cuatro esquemas principales de distribución local de vídeo a solicitud tienen las siglas ADSL, FTTC, FITH y HFC. Ahora los explicaremos.

El ADSL (*Asymmetric Digital Subscriber Line*, línea asimétrica digital de suscriptor) fue el primer competidor de la industria telefónica por el premio de la distribución local (Chen y Waring, 1994). La idea es que a prácticamente cada casa de Estados Unidos, Europa y Japón ya entra un par trenzado de cobre (para el servicio telefónico analógico). Si estos alambres pudieran usarse para el video a solicitud, las compañías telefónicas podrían ganar.

El problema, claro está, es que estos alambres no pueden manejar ni siquiera MPEG-1 a través de su recorrido normal de 10 km, ya no se diga MPEG-2. La solución ADSL aprovecha los avances en el procesamiento digital de señales para eliminar electrónicamente los ecos y otros ruidos de la línea. Como se muestra en la figura 7-92, cada suscriptor ADSL recibe una unidad ADSL casera que contiene un chip para el procesamiento de señales digitales. El teléfono y la caja de control se conectan a la unidad ADSL. En el otro extremo del lazo local, se conecta otra unidad ADSL. Ésta puede estar en la oficina terminal de la compañía telefónica o, si el lazo local es demasiado grande, en el extremo de una fibra óptica cercana a la casa.

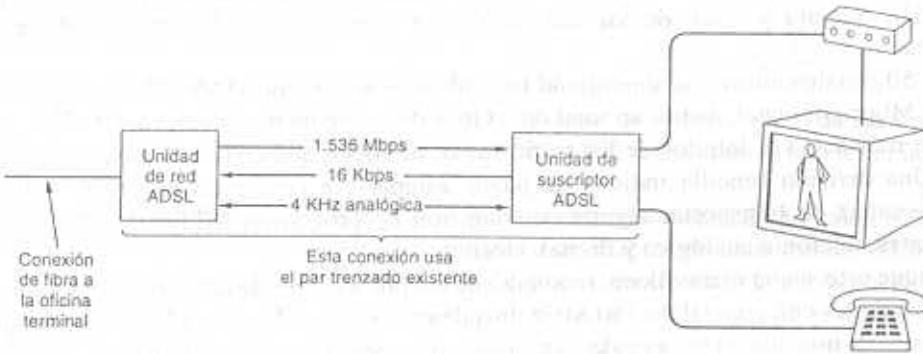


Figura 7-92. ADSL como red de distribución local.

El ADSL-1 ofrece un canal de enlace descendente de 1.536 Mbps (T1 menos el bit 193), pero sólo un canal de enlace ascendente de 16 kbps. Además está ahí el viejo canal analógico telefónico de 4 kHz (o, en algunos casos, dos canales digitales N-ISDN). La idea es que el enlace ascendente tenga suficiente ancho de banda para que el usuario solicite películas, y el enlace descendente tenga bastante ancho de banda para enviarlas codificadas en MPEG-1. El ADSL debe verse más como una solución rápida y sin complicaciones que una solución a largo plazo, pero ya se está instalando en varias ciudades. También se está trabajando en versiones mejoradas, llamadas ADSL-2 y ADSL-3. Esta última permite transportar MPEG-2 por lazos locales de hasta 2 km.

El segundo diseño de las compañías telefónicas es el FTTC (*Fiber To The Curb*, fibra hasta la acera). Vimos este diseño en la figura 2-23(a). En FTTC, la compañía telefónica tiende fibra óptica de la oficina terminal a cada barrio residencial, terminando en un dispositivo llamado

ONU (*Optical Network Unit*, unidad de red óptica). La ONU se llama "caja de unión" en la figura 2-23(a). En una ONU pueden terminar hasta 16 lazos locales de cobre. Estos lazos son lo bastante cortos para operar T1 o T2 dúplex integral en ellos, permitiendo películas MPEG-1 y MPEG-2, respectivamente. Además son posibles videoconferencias para trabajadores remotos y negocios pequeños, pues el FTTC es simétrico.

La tercera solución de las compañías telefónicas es tender fibra a todas las casas. Este sistema se llama FTTH (*Fiber To The Home*, fibra hasta la casa). En él, todos mundo puede tener una portadora OC-1, OC-3 o inclusive mayor, si se requiere. El FITH es muy caro y tardará años en hacerse realidad, pero ciertamente abrirá una vasta gama de nuevas posibilidades cuando finalmente ocurra.

ADSL, FTTC y FTTH son redes de distribución local punto a punto, lo cual no es sorprendente dada la organización del sistema telefónico actual. Un enfoque completamente diferente es el HFC (*Hybrid Fiber Coax*, híbrido fibra/coaxial), la solución preferida que están instalando los proveedores de TV por cable y que se ilustra en la figura 2-23(b). La propuesta es la siguiente. Los cables coaxiales actuales de 300 a 450 MHz serán sustituidos por cables coaxiales de 750 MHz, elevando la capacidad de 50 a 75 canales de 6 MHz a 125 canales de 6 MHz. Setenta y cinco de los 125 canales se usarán para la transmisión de televisión analógica.

Los 50 canales nuevos se modularán individualmente usando QAM-526, que proporciona unos 40 Mbps por canal, dando un total de 2 Gbps de ancho de banda nuevo. Los *head-ends* se moverán más hacia el interior de los vecindarios, de modo que cada cable pase por sólo 500 casas. Una división sencilla indica que puede asignarse a cada casa un canal dedicado de 4 Mbps, capaz de transportar alguna combinación de programas MPEG-1, MPEG-2, datos ascendentes, telefonía analógica y digital, etcétera.

Aunque esto suena maravilloso, requiere que los proveedores de cable reemplacen todos los cables existentes con coaxial de 750 MHz, instalen nuevos *head-ends* y eliminan todos los amplificadores de una vía; en pocas palabras, que reemplacen el sistema completo de TV por cable. En consecuencia, la cantidad de infraestructura nueva es comparable con la que necesitan las compañías telefónicas para FTTC. En ambos casos, el proveedor local de red tiene que tender fibra a los barrios residenciales. También en ambos casos, la fibra termina en un convertidor optoelectrónico. En FTTC, el segmento final es un lazo local punto a punto que usa pares trenzados. En HFC, el segmento final es cable coaxial compartido. Técnicamente, estos sistemas no son tan diferentes, como suelen asegurar sus patrocinadores.

No obstante, hay una diferencia real que es importante indicar. HFC usa un medio compartido sin conmutación ni enrutamiento. Cualquier información que se ponga en el cable puede ser retirada por cualquier suscriptor sin mayores trámites. FTTC, que es completamente conmutado, no tiene esta propiedad. Como resultado, los operadores HFC quieren que los servidores transmitan cadenas cifradas, de modo que los clientes que no hayan pagado una película no puedan verla. A los operadores FTTC no les interesa el cifrado, puesto que agrega complejidad, disminuye el desempeño y no proporciona seguridad adicional a su sistema. Desde el punto de vista de la compañía que opera un servidor de vídeo, ¿es buena idea cifrar? Un servidor operado por una

compañía telefónica o una de sus subsidiarias o socios podría intencionalmente decidir no cifrar sus videos, indicando como razón la eficiencia, pero en realidad para provocar pérdidas a sus competidores HFC.

Con todas estas redes locales de distribución es probable que se equipe a cada vecindario con uno o más servidores de *spool*. Éstos son, de hecho, simplemente versiones más pequeñas de los servidores de video que estudiamos antes. La gran ventaja de estos servidores locales es que, dado que las redes de distribución local son pequeñas y generalmente no son conmutadas, no introducen fluctuaciones como lo haría una red de *backbone ATM*.

Dichos servidores locales pueden precargarse con películas de manera dinámica o por reservación. Por ejemplo, al seleccionar un usuario una película, podría transmitirse el primer minuto al servidor local en menos de 2 segundos con OC-3. Tras 55 segundos, el siguiente minuto podría enviarse al servidor local en 2 segundos, etc. De esta manera, el tráfico a través del *backbone ATM* ya no tiene que estar libre de fluctuaciones, haciendo posible el uso del servicio ABR en lugar de CBR, más costoso.

Si la gente indica con suficiente adelanto al proveedor las películas que desea ver, pueden bajarse al servidor local durante horas no pico, obteniéndose aún más ahorros. Esta observación probablemente llevará a los operadores de red a contratar ejecutivos de aviación para diseñar las tarifas. Pueden preverse tarifas en las que las películas ordenadas con 24 a 72 horas de adelanto para ser vistas la tarde de un martes o jueves antes de las 6 p.m. o después de las 11 p.m. tendrán 27% de descuento. Las películas ordenadas el primer domingo del mes antes de las 8 a.m. para ser vistas un miércoles por la tarde de un día cuya fecha sea un número primo tendrán 43% de descuento, etcétera.

La selección del grupo de protocolos para video a solicitud aún está en el aire. ATM claramente es la tecnología a escoger, pero ¿qué protocolo de adaptación debe usarse? El AAL 1 se diseñó para video, por lo que es un candidato fuerte, pero corresponde a la categoría de servicio CBR. Dedicar el máximo ancho de banda que podría necesitarse es caro, especialmente dado que el MPEG es inherentemente tráfico VBR, por lo que el circuito virtual tendrá que sobredimensionarse.

El AAL 2 no está acabado (y probablemente nunca lo estará), y el AAL 3/4 es demasiado torpe, así que el AAL 5 es el único contendiente que queda. AAL 5 no está atado a un servicio CBR, y el envío de un bloque grande de MPEG en cada mensaje sería extremadamente eficiente, pudiéndose dedicar casi 100% del ancho de banda de usuario para la corriente de video. Por el lado negativo, el AAL 5 hace detección de errores. El que se descarte un bloque completo debido a un error de un bit es muy poco atractivo, especialmente dado que la mayoría de los errores son de un solo bit a la mitad de los datos. Como consecuencia, hay quienes proponen cambiar el AAL 5 de modo que permita que las aplicaciones soliciten todos los bloques, junto con un bit que indique si la cifra de comprobación fue correcta o no.

La pila de protocolos de video a solicitud que acabamos de esbozar se ilustra en la figura 7-93. Por encima de la capa AAL, vemos el programa MPEG y la capa de transporte de corrientes. Luego vienen la codificación y decodificación de audio y video MPEG, respectivamente. Por último, tenemos hasta arriba la aplicación.

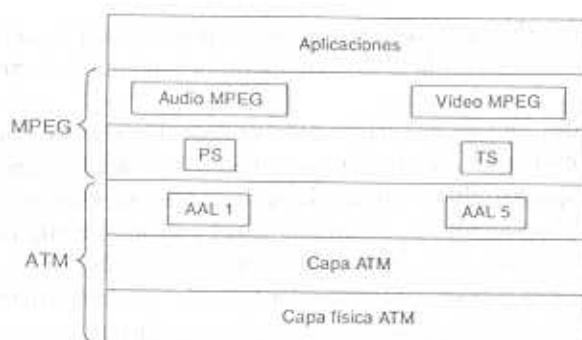


Figura 7-93. Una pila de protocolos de video a solicitud.

Cajas de control

Todos los métodos de distribución local anteriores llevan en última instancia una o más corrientes MPEG a la casa. Para decodificarlas y verlas, se requiere una interfaz de red, un decodificador MPEG y otros componentes electrónicos. Son posibles dos enfoques.

En el primer enfoque, la gente usa sus computadoras personales para decodificar y ver películas. Para ello sería preciso comprar una tarjeta especial que contenga unos cuantos *chip* especiales y un conector para hacer interfaz con la red local de distribución. Las películas aparecen en el monitor de la computadora, posiblemente en una ventana. Este enfoque es más barato (todo lo que se requiere es una tarjeta y el *software*), usa un monitor no entrelazado de alta definición, tiene una interfaz de usuario manejada por ratón, y puede integrarse fácilmente con la WWW y otras fuentes de información y entretenimiento orientadas a computadora. Por otra parte, las PC generalmente tienen pantallas pequeñas, se ubican en estudios o en cuartos pequeños en lugar de salas y tradicionalmente las usa una sola persona a la vez. También emiten bastante menos luz que los aparatos de televisión.

En el segundo enfoque, el operador de red local renta o vende al usuario una caja de control, a la que se conectan la red y el aparato de televisión. Este enfoque tiene la ventaja de que todo mundo tiene una televisión y no todos una PC, y muchas de las PC son viejas, peculiares, o no son adecuadas para la decodificación MPEG. Además, la televisión normalmente está ubicada en un cuarto adecuado para un grupo de gente.

Por el lado negativo, el monitor tiene un cinescopio entrelazado de baja definición (lo que lo hace inadecuado para material orientado a texto, como la WWW); además, tiene una interfaz de usuario horrenda (el control remoto), lo que hace virtualmente imposible que el usuario haga otra cosa que no sea seleccionar opciones de menús sencillos. Incluso escribir el nombre de una película es un asunto laborioso, no se diga entablar un diálogo solicitando al servidor que busque todas las películas de cierto actor, director o compañía productora durante cierto período de

tiempo. Por último, no es fácil producir las cajas de control con el desempeño requerido a un precio aceptable (que se piensa será de unos cuantos cientos de dólares).

Considerando todos estos factores, la mayoría de los sistemas de video a solicitud han optado por el modelo de caja de control, principalmente porque los estrategas de mercadeo de masas odian excluir a cualquier cliente potencial (gente sin PC). También, puede haber ganancias en la renta o venta de cajas de control. No obstante, el mercado para las tarjetas de PC es bastante grande, por lo que sin duda se producirán también estas tarjetas.

Las funciones primarias de la caja de control son la interfaz con la red de distribución local, la decodificación de la señal MPEG, la sincronización de las corrientes de audio y video, la producción de una señal compuesta NTSC, PAL o SECAM para el aparato televisor, la recepción de señales del control remoto, y el manejo de la interfaz de usuario. Algunas funciones adicionales podrían incluir interfaces con equipos de sonido estereofónico, teléfonos y otros dispositivos. Hay actualmente una gran batalla en la industria sobre la funcionalidad que debe incluirse en la caja de control y la que debe estar en la red. El resultado está por verse.

Una arquitectura posible de una caja de control sencilla se muestra en la figura 7-94. El dispositivo consta en CPU, ROM, RAM, controlador de E/S, decodificador MPEG e interfaz de red. Opcionalmente puede agregarse un *chip* de seguridad para el descifrado de películas y el cifrado de menajes de salida (números de tarjeta de crédito para compras desde casa, etcétera).

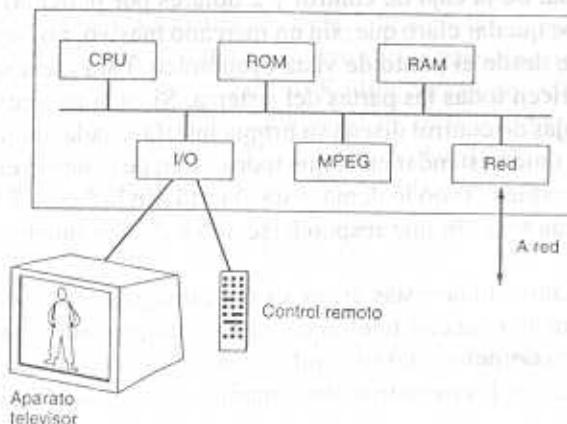


Figura 7-94. Arquitectura de hardware de una caja de control sencilla.

Un problema importante del video a solicitud es la sincronización de audio/video y el manejo de la fluctuación. La adición de unos 500 kB de RAM permite contar con un segundo de buffer de MPEG-2, pero con un costo adicional en un dispositivo que los fabricantes esperan vender en unos cuantos cientos de dólares, cuando mucho.

Puesto que la caja de control es sólo una computadora, necesitará *software*, probablemente un sistema operativo de tiempo real basado en micronúcleo, almacenado en ROM. Para proporcionar flexibilidad y adaptabilidad, probablemente sería conveniente permitir la carga de otro *software* desde la red. Esta posibilidad genera un problema: ¿qué ocurre cuando el dueño de una caja de control basada en MIPS quiere jugar un juego escrito para una caja de control basada en SPARC? El uso de un lenguaje interpretado como Java resuelve el problema de la compatibilidad, pero disminuye severamente el desempeño en un entorno de tiempo real en el que es crucial el alto desempeño.

Estándares

No puede ignorarse el aspecto económico del video a solicitud. Un servidor grande de video fácilmente puede costar más que un *mainframe*, ciertamente unos 10 millones de dólares. Supóngase que el servidor da servicio a 100,000 hogares, cada uno de los cuales ha rentado una caja de control de 300 dólares. Ahora agréguese 10 millones de dólares en equipo de red con un periodo de depreciación de cuatro años, y el sistema tiene que generar 10 dólares por casa por mes. A cinco dólares/película, todos tendrían que comprar dos películas al mes, solamente para que el operador recupere la inversión (sin tomar en cuenta salarios, mercadeo y todos los demás costos). El que realmente vaya a ocurrir esto, no se puede saber.

Las cifras indicadas antes pueden redistribuirse de varias maneras (por ejemplo, cargando 6 dólares por renta mensual de la caja de control y 2 dólares por película), y los costos cambian todo el tiempo, pero debe quedar claro que, sin un mercado masivo, no hay modo de que el video a solicitud tenga sentido desde el punto de vista económico. Para crear un mercado masivo, es esencial que se estandaricen todas las partes del sistema. Si cada proveedor de video, operador de red y fabricante de cajas de control diseña su propia interfaz, nada interactuará con el resto del sistema. Hasta ahora, el único estándar en el que todos están de acuerdo es con el uso de MPEG-2 para la codificación de video. Todo lo demás está abierto. En la figura 7-95 se listan algunas de las muchas preguntas que tendrán que responderse antes de que pueda construirse un sistema nacional.

Si pueden estandarizarse todas estas áreas, es fácil imaginar la producción de equipos que consistan en una caja con un conector telefónico, un monitor, teclado y ratón que puedan usarse para ver videos, efectuar cómputo, o tal vez ambas cosas a la vez. Entonces será realidad la muy mencionada convergencia de las industrias de cómputo, comunicación y entretenimiento.

7.7.5. MBone — *Backbone* de multidifusión

Mientras todas estas industrias hacen grandes (y muy publicitados) planes para el video digital (inter)nacional a solicitud, la comunidad de Internet ha estado implementando calladamente su propio sistema digital de multimedia, MBone (*Multicast Backbone, backbone* de multidifusión). En esta sección haremos un repaso breve a sus características y su funcionamiento. Para un libro completo sobre MBone, véase (Kumar, 1996). Para artículos sobre MBone, véase (Eriksson, 1994, y Macedonia y Brutzman, 1994).

¿Qué tecnología usará el <i>backbone</i> (SONET, ATM, SONET + ATM)?
¿A qué velocidad operará el <i>backbone</i> (OC-3, OC-12)?
¿Cómo se efectuará la distribución local (HFC, FTTC)?
¿Cuál será el ancho de banda ascendente (16 kbps, 1.5 Mbps)?
¿Se cifrarán las películas? De ser así, ¿cómo?
¿Habrá corrección de errores (obligatoria, opcional, ausente)?
¿Quién será el dueño de la caja de control (usuario, operador de red)?
¿Será la telefonía parte del sistema (análogica, N-ISDN)?
¿Se manejarán aplicaciones de hipertexto de alta resolución (por ejemplo, www)?

Figura 7-95. Algunas de las áreas en las que se necesitan estándares.

Puede pensarse en el MBone como una radio y televisión de Internet. A diferencia del video a solicitud, donde lo más importante es solicitar y ver películas precompresionadas almacenadas en un servidor, MBone se usa para difundir audio y video en vivo de forma digital por todo el mundo a través de Internet. MBone ha estado en operación desde comienzos de 1992. Se han difundido muchas conferencias científicas, especialmente las reuniones del IETF, así como eventos científicos de importancia noticiosa, como varios lanzamientos del transbordador espacial. Alguna vez se difundió un concierto de los Rolling Stones por MBone. El que esto tenga un interés científico está abierto a debate. También hay *software* disponible para aquellos que deseen grabar digitalmente una difusión MBone (Hofstader, 1995).

La mayor parte de las investigaciones acerca del MBone se han orientado a la manera de multitransmitir eficientemente a través de Internet (orientada a datagramas). Se ha hecho poco acerca de la codificación de audio o video. Las fuentes MBone están en libertad de experimentar con MPEG o cualquier otra tecnología que deseen. No hay estándares de Internet respecto al contenido o la codificación.

Desde el aspecto técnico, MBone es una red virtual sobrepuerta a Internet. Consiste en islas capaces de multitransmisión conectadas mediante túneles, como se muestra en la figura 7-96. En esta figura, el MBone consiste en seis islas, A a F, conectadas mediante siete túneles. Cada isla (por lo común, una LAN o grupo de LAN interconectadas) apoya la multitransmisión por *hardware* a sus *hosts*. Los túneles propagan paquetes MBone entre las islas. En el futuro, cuando todos los enrutadores sean capaces de manejar directamente tráfico de multitransmisión, dejará de necesitarse esta superestructura, pero por el momento lleva a cabo el trabajo.

Cada isla contiene uno o más enrutadores especiales, llamados enrutadores m o *mrouters* (enrutadores de multitransmisión). Algunos de éstos son en realidad enrutadores normales, pero otros sencillamente son estaciones de trabajo UNIX que ejecutan *software* especial de nivel de usuario (pero como la raíz). Los enrutadores m se conectan lógicamente mediante túneles. En el pasado, los paquetes MBone se enviaban por un túnel de enrutador m a enrutador m (general-

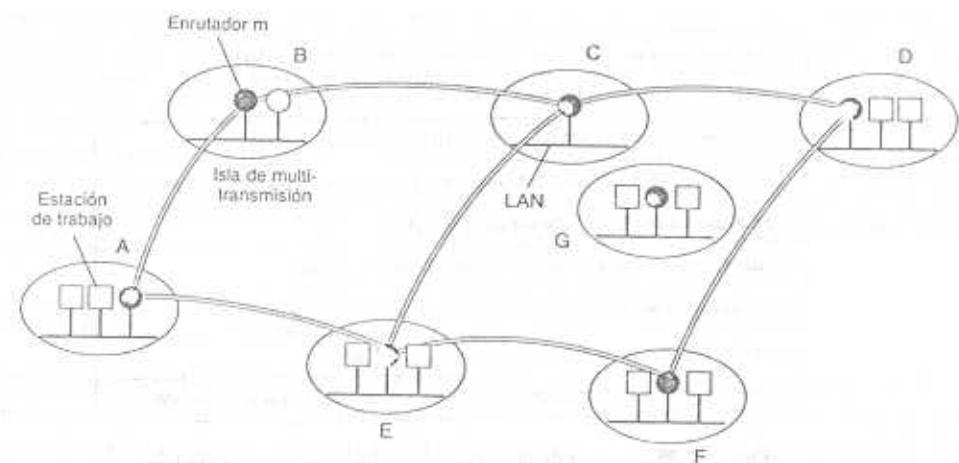


Figura 7-96. El MBone consiste en islas de multitransmisión conectadas mediante túneles.

mente a través de uno o más enrutadores que no entendían de MBone), usando enrutamiento no estricto desde el origen. Hoy día, los paquetes MBone se encapsulan en paquetes IP y se envían como paquetes de unitransmisión normales a la dirección IP del enrutador m del destino. Sin embargo, si todos los enrutadores que intervienen apoyan la multitransmisión, no se necesita el proceso de túnel.

Los túneles se configuran de manera manual. Por lo general, un túnel corre por una trayectoria para la que existe una conexión física, pero esto no es un requisito. Si por accidente se cae la trayectoria física subyacente de un túnel, los enrutadores m que usan el túnel ni siquiera lo notarán, puesto que la Internet reenrutará automáticamente todo el tráfico IP entre ellos por otras líneas.

Al aparecer una isla nueva que desea unirse al MBone, como en el caso de G de la figura 7-96, su administrador envía un mensaje anunciando su existencia a la lista de correo de MBone. Los administradores de las instalaciones cercanas entonces se ponen en contacto con ella para acordar el establecimiento de túneles. A veces se reordenan los túneles existentes para aprovechar la nueva isla y optimizar la topología. A fin de cuentas, los túneles no existen físicamente, se definen mediante tablas en los enrutadores m y pueden agregarse, eliminarse o moverse con sólo cambiar estas tablas. Por lo común, cada país en el MBone tiene un *backbone* con islas regionales conectadas a él. Normalmente se configura el MBone con uno o dos túneles que cruzan el Océano Atlántico y el Pacífico, haciendo al MBone de escala global.

Por tanto, en cualquier momento, el MBone consiste en una topología específica de túneles e islas, independiente de la cantidad de direcciones de multitransmisión en uso en el momento y de quiénes las están escuchando u observando. Esta situación es muy parecida a una subred normal (física), por lo que se aplican los mismos algoritmos de enrutamiento. En consecuencia, el MBone inicialmente usaba un algoritmo de enrutamiento, el *DVMRP* (*Distance Vector Multicast Routing Protocol*, protocolo de enrutamiento multitransmisión por vector de distancia),

basado en el algoritmo de vector de distancia Bellman-Ford. Por ejemplo, en la figura 7-96, la isla C puede enrutar a A vía B o vía E (o tal vez vía D). Esto lo decide C tomando los valores que le dan los nodos sobre sus respectivas distancias a A, y sumando después su distancia a ellos. De esta manera, cada isla determina la mejor ruta a todas las demás islas. Sin embargo, las rutas en realidad no se usan de esta manera, como veremos en breve.

Consideremos ahora la manera en que ocurre la multitransmisión. Para multitransmitir un programa de audio o video, una fuente primero debe adquirir una dirección de multitransmisión clase D, que actúa como la frecuencia o el número de canal de una estación. Las direcciones clase D se reservan usando un programa que busca direcciones de multitransmisión libres en una base de datos. Pueden ocurrir simultáneamente muchas multitransmisiones, y un *host* puede "sintonizarse" con aquella en la que está interesado si escucha en la dirección de multitransmisión adecuada.

Periódicamente, cada enrutador m envía un paquete de difusión IGMP limitado a su isla, preguntando quién está interesado en qué canal. Los *hosts* que desean (continuar) recibiendo uno o más canales envían como respuesta otro paquete IGMP. Estas respuestas se reparten en el tiempo, para evitar sobrecargar la LAN local. Cada enrutador m mantiene una tabla de los canales que debe poner en sus LAN, para evitar el desperdicio de ancho de banda al multitransmitir canales que nadie quiere.

Las multitransmisiones se propagan por el MBone según sigue. Cuando una fuente de audio o video genera un paquete nuevo, lo multitransmite a su isla local usando el recurso de multitransmisión del *hardware*. Este paquete lo recoge el enrutador m local, que entonces lo copia en todos los túneles a los que está conectado.

Cada enrutador m que recibe uno de tales paquetes por medio de un túnel lo revisa para ver si llegó por la mejor ruta, es decir, la ruta que, según su tabla, debe usarse para llegar al origen (como si fuera el destino). Si el paquete llegó por la mejor ruta, el enrutador m copia el paquete en todos sus otros túneles. Si el paquete llegó por una ruta subóptima, se descarta. Por ejemplo, en la figura 7-96, si las tablas de C le indican que use B para llegar a A, entonces cuando un paquete multitransmisión de A a C llega vía B, se copia de ahí a D y a E. Sin embargo, cuando un paquete de multitransmisión de A a C llega vía E (que no es la mejor trayectoria), simplemente se descarta. Este algoritmo sólo es el de reenvío por trayectoria invertida que vimos en el capítulo 5; aunque no es perfecto, es bastante bueno y muy sencillo de implementar.

Además de usar reenvío por trayectoria invertida para evitar inundar la Internet, también se usa el campo *tiempo de vida* del IP para limitar el alcance de la multitransmisión. Cada paquete comienza con algún valor (determinado por el origen). A cada túnel se le asigna un peso. Un paquete sólo pasa a través de un túnel si tiene suficiente peso; de otra manera, se descarta. Por ejemplo, los túneles transoceánicos se configuran normalmente con un peso de 128, por lo que los paquetes pueden limitarse al continente de origen dándoles un *tiempo de vida* inicial de 127 o menos. Tras pasar por el túnel, se resta el peso del túnel del campo de *tiempo de vida*.

Aunque el algoritmo de enrutamiento de MBone funciona, se han realizado muchas investigaciones para mejorarlo. Una propuesta mantiene la idea del enrutamiento por vector de distancia, pero hace jerárquico el algoritmo, agrupando en regiones las instalaciones MBone y enrutando primero hacia ellas (Thyagarajan y Deering, 1995).

Otra propuesta es usar una forma modificada del enrutamiento por estado de enlace en lugar de enrutamiento por vector de distancia. En particular, un grupo de trabajo del IETF está modificando el OSPF para adecuarlo a la multitransmisión dentro de un solo sistema autónomo. El OSPF de multitransmisión resultante se llama MOSPF (Moy, 1994). Lo que hacen las modificaciones es que el mapa completo construido por el MOSPF lleva el registro de las islas y túneles de multitransmisión, además de la información normal de enrutamiento. Si se conoce la topología completa, es directo el cálculo de la mejor trayectoria desde cualquier isla a cualquier otra usando los túneles. Por ejemplo, puede usarse el algoritmo de Dijkstra.

Una segunda área de investigación es el enrutamiento entre los AS. Aquí, otro grupo de trabajo del IETF está desarrollando un algoritmo llamado PIM (*Protocol Independent Multicast, multitransmisión independiente del protocolo*) (Huitema, 1995). El PIM viene en dos versiones, dependiendo si las islas son densas (casi todos quieren ver) o dispersas (casi nadie quiere ver). Ambas versiones usan las tablas de enrutamiento de unitransmisión estándar, en lugar de crear una topología superpuesta como lo hacen DVMRP y MOSPF.

En el PIM denso, la idea es recortar las trayectorias inútiles. El recortado funciona como sigue. Cuando llega un paquete multitransmisión por el túnel "equivocado", se devuelve un paquete de recorte por el túnel, indicando al transmisor que cese el envío de paquetes desde la fuente en cuestión. Al llegar un paquete por el túnel "correcto", se copia en todos los demás túneles que no han sido previamente recortados. Si todos los demás túneles se han recortado y no hay interés por el canal en la isla local, el enrutador m devuelve un mensaje de recorte por el canal "correcto". De esta manera, la multitransmisión se adapta automáticamente y sólo va a donde se le quiere.

El PIM disperso funciona de manera diferente. Aquí la idea es evitar la saturación de Internet porque tres personas de Berkeley quieren realizar una llamada en conferencia por una dirección clase D. El PIM disperso opera estableciendo puntos de reunión. Cada una de las fuentes de un grupo de multitransmisión PIM disperso envía sus paquetes a los puntos de reunión. Cualquier instalación interesada en unirse solicita a uno de los puntos de reunión que establezca un túnel a ella. De esta manera, todo el tráfico PIM disperso se transporta por unitransmisión en lugar de multitransmisión.

En resumen, los multimedia son un campo emocionante y de rápido movimiento. Diariamente se anuncian tecnologías y aplicaciones nuevas, pero el área en conjunto sin duda será importante durante las siguientes décadas.

7.8. RESUMEN

Las redes de cómputo son inherentemente inseguras. Para mantener la información en secreto, ésta debe cifrarse. Los protocolos de cifrado pertenecen a dos clases generales: de clave secreta (por ejemplo, DES, IDEA) y de clave pública (por ejemplo, RSA). El uso de estos protocolos es directo; la parte difícil es la administración de claves.

Además de proporcionar secreto, los protocolos criptográficos también pueden proporcionar validación de identificación, de modo que cuando Alicia piensa que se está comunicando con Benjamín, en realidad se está comunicando con Benjamín, y no con Trudy. Por último, la

criptografía también puede servir para permitir la firma de mensajes de manera que el transmisor no pueda repudiarlos una vez enviados.

Los nombres en Internet usan un sistema distribuido de base de datos, el DNS. El DNS contiene registros con las direcciones IP, centrales de correo y otra información. Mediante consultas al servidor DNS, un proceso puede relacionar un nombre de dominio de Internet con la dirección IP usada para comunicarse con ese dominio.

A medida que crecen las redes, se vuelven más difíciles de administrar. Por esta razón, se han desarrollado sistemas y protocolos especiales de administración de redes, de los cuales el más común es el SNMP. Este protocolo permite a los administradores comunicarse con agentes que están dentro de los dispositivos para leer su estado y emitirles comandos.

Cuatro de las aplicaciones principales de las redes son el correo electrónico, las noticias USENET, la World Wide Web y los multimedia (vídeo a solicitud y MBone). La mayoría de los sistemas de correo electrónico usan el sistema de correo definido en los RFC 821 y 822. Los mensajes enviados por este sistema usan cabeceras de sistema en ASCII para definir las propiedades del mensaje. Estos mensajes se envían usando SMTP. Existen dos sistemas para hacer seguro el correo electrónico, el PGP y el PEM.

Las noticias USENET consisten en miles de grupos de noticias sobre todos los temas. La gente puede suscribirse a los grupos de noticias de manera local, y entonces puede publicar mensajes en todo el mundo usando el protocolo NNTP, que tiene algún parecido con el SMTP.

La World Wide Web (red mundial) es un sistema para vincular documentos de hipertexto. Cada documento es una página escrita en HTML, posiblemente con hipervínculos con otros documentos. Un visualizador puede presentar en pantalla un documento estableciendo una conexión TCP con su servidor, solicitando el documento y cerrando después la conexión. Al seleccionar el usuario un hipervínculo, puede traerse por la misma vía ese documento. De este modo, los documentos de todo el mundo se vinculan para formar una red (*web*) inmensa.

Los multimedia son la nueva estrella en el firmamento de las redes; permiten la digitalización de audio y vídeo, y su transporte electrónico para exhibirse. La mayoría de los proyectos multimedia usa los estándares MPEG y transmite los datos por conexiones ATM. El MBone es un servicio experimental de radio y televisión digital de alcance mundial que opera en Internet.

PROBLEMAS

1. Descifre el siguiente cifrado monoalfabético. El texto normal, que consiste sólo en letras, es un fragmento bien conocido de un poema de Lewis Carroll, que está en inglés.

kfd ktbd fzim cubd kfd pzyiom mztx ku kzyg ur bzha kfthcm
ur mfudm zhx mftum zhx mdzythc pzq ur ezzszedm zhx gthcm
zhx pfa kfd mdz tm sutythe fuk zhx pfdkfdi ntcn fzld pthcm
sok pztk z stk kfd uamkdim citdx sdruid pd fzld uoi efzk
rui mubd ur om zid uok ur sidzfkf zhx zyj ur om zid rzk
hu foia mztx kfd ezindhkdi kfda kfzhgdx ftb boef rui kfzk

2. Descifre el siguiente cifrado por transposición columnar. El texto normal (en inglés) se tomó de un libro de texto sobre computación bastante conocido, por lo que "computer" es una palabra probable. El texto normal consiste por entero en letras (sin espacios). El texto cifrado se dividió en bloques de cinco caracteres para hacerlo más legible.

aauan cylre rurnn dltme aeebp ytust iceat npmey iicgo gorch srsoc
nnii imiha oofpa gsivt tpsit lboir otoex

3. En la figura 7-4, las cajas P y las cajas S se alternan. Aunque este arreglo es atractivo estéticamente, ¿es más seguro que tener primero todas las cajas P y luego todas las cajas S?
4. Suponga que se ha cifrado un mensaje usando DES en modo de encadenamiento de bloque cifrado. Un bit de texto cifrado del bloque C_i cambia accidentalmente de 0 a 1 durante la transmisión. ¿Cuánto texto normal se alterará como resultado?
5. Considere nuevamente el encadenamiento de bloque cifrado. En lugar de transformarse un solo bit 0 en bit 1, se inserta un bit 0 extra en la corriente de texto cifrado después del bloque C_i . ¿Cuánto texto normal se alterará como resultado?
6. Diseñe un ataque al DES con base en el conocimiento de que el texto cifrado consiste exclusivamente en letras ASCII mayúsculas más espacios, comas, puntos, punto y coma, retornos de carro y saltos de línea. No se sabe nada sobre los bits de paridad del texto normal.
7. Compare el encadenamiento de bloque cifrado con el modo de realimentación de cifrado en términos de la cantidad de operaciones de cifrado necesarias para transmitir un archivo grande. ¿Cuál es más eficiente, y por cuánto?
8. Usando el criptosistema RSA de clave pública, con $a = 1$, $b = 2$, etc.,
- Si $p = 7$ y $q = 11$, liste cinco valores legales de d .
 - Si $p = 13$, $q = 31$ y $d = 7$, encuentre e .
 - Usando $p = 5$, $q = 11$ y $d = 27$, encuentre e y cifre "abcdefghijkl".
9. Se está usando el intercambio de claves Diffie-Hellman para establecer una clave secreta entre Alicia y Benjamín. Alicia envía a Benjamín (719, 3, 191). Benjamín responde con (543). El número secreto de Alicia, x , es 16. ¿Cuál es la clave secreta?
10. Modifique ligeramente un mensaje del protocolo de la figura 7-14 para hacerlo resistente al ataque por reflexión. Explique por qué funciona su cambio.
11. En el protocolo de rana de boca amplia, ¿por qué se envía A en texto normal junto con la clave de sesión cifrada?
12. En el protocolo de rana de boca amplia indicamos que es un riesgo de seguridad iniciar cada mensaje de texto normal con 32 bits cero. Supóngase que cada mensaje comienza con un número aleatorio por usuario, de hecho una segunda clave secreta conocida sólo por su usuario y el KDC. ¿Impide esto el ataque por texto normal conocido?
13. En el protocolo Needham-Schroeder, Alicia genera dos retos, R_A y R_{A2} . Esto parece demasiado. ¿Habría bastado con uno solo?
14. En el protocolo de validación de identificación de clave pública de la figura 7-21, en el mensaje 3, R_B se cifra con K_S . ¿Es necesario este cifrado, o habría sido adecuado devolver R_B en texto normal?

15. El protocolo de firma de la figura 7-22 tiene la siguiente debilidad. Si Benjamín se cae, puede perder el contenido de su RAM. ¿Qué problemas causa esto y qué puede hacerse para evitarlos?
16. Tras la confesión de Elena a Marilyn respecto al engaño en el asunto de la cátedra de Tomás, Marilyn decidió evitar este problema dictando sus mensajes futuros a una grabadora y haciendo que su nueva secretaria sólo los mecanograffe. Marilyn planeaba examinar los mensajes en su terminal, una vez mecanografiados, para asegurarse de que contuvieran sus palabras exactas. ¿Puede la nueva secretaria usar todavía el ataque de cumpleaños para falsificar un mensaje, y de ser así, cómo? *Sugerencia:* Si puede.
17. Las terminales de punto de venta que usan tarjetas con tira magnética y códigos PIN tienen una falla mortal: un comerciante sin escrúpulos puede modificar su lector de tarjetas para capturar y almacenar toda la información sobre la tarjeta al igual que el código PIN, a fin de efectuar transacciones adicionales (falsas) en un futuro. La siguiente generación de terminales de punto de venta usará tarjetas con una CPU completa, teclado y pantalla miniatura en la tarjeta. Diseñe un protocolo para este sistema que no puedan violar los comerciantes sin escrúpulos.
18. De acuerdo con la información dada en la figura 7-27, ¿está `little-sister.cs.vu.nl` en una red clase A, B o C?
19. En la figura 7-27 no hay un punto después de `rowboat`. ¿Por qué no?
20. ¿Cuál es el *OBJECT IDENTIFIER* del objeto `tcp`?
21. Es necesario transmitir un entero SNMP cuyo valor es 200. Muestre la representación binaria de los bits enviados en la sintaxis de transferencia ASN.1.
22. ¿Cuál es la representación de la cadena de 11 bits binarios '11100001111' en la sintaxis de transferencia ASN.1?
23. Suponga que ha sido contratado por un proveedor de puentes para escribir código que se ajuste al SNMP para uno de sus puentes. Usted lee todos los RFC y aún tiene preguntas, por lo que sugiere al IAB que se especifique en un solo lugar una gramática formal y completa del lenguaje empleado para describir las variables SNMP. La reacción del IAB es estar de acuerdo y encargarle a usted la tarea. ¿Debe agregarse la gramática al RFC 1442 o al RFC 1213? ¿Por qué? *Sugerencia:* No necesita buscar en los RFC; se da bastante información en el texto.
24. Algunos sistemas de correo electrónico reconocen un campo de cabecera *Content Return*, que especifica si el cuerpo de un mensaje debe devolverse en el caso de no poderse entregar. ¿Pertenece este campo al sobre o a la cabecera?
25. Los sistemas de correo electrónico necesitan directorios para poder buscar las direcciones de correo electrónico de la gente. Para construir tales directorios, deben dividirse los nombres en componentes estándar (por ejemplo, nombre y apellido), a fin de hacer posible la búsqueda. Explique algunos de los problemas que deben resolverse para que sea aceptable un estándar mundial.
26. Un archivo binario tiene 3072 bytes de longitud. ¿Qué tan grande será si se codifica usando codificación base64, con la inserción de un par CR+LF tras cada 80 bytes enviados y al final?
27. Considere el esquema de codificación MIME entrecomillado-imprimible. Mencione un problema que no se haya analizado en el texto y proponga una solución.
28. Dé dos razones por las que el PGP comprime los mensajes.

29. Suponga que alguien establece un *daemon* de vacaciones y luego envía un mensaje justo antes de terminar la sesión. Desafortunadamente, el receptor ha estado de vacaciones durante una semana y también tiene activo un *daemon* de vacaciones. ¿Qué ocurre después? ¿Continuarán yendo y viéndolo los mensajes enlatados hasta que alguien regrese?
30. Suponiendo que todos en la Internet usaran PGP, ¿podría enviarse un mensaje PGP a una dirección Internet arbitraria y ser decodificado correctamente por todos los interesados? Explique su respuesta.
31. PGP no reconoce la canonización como lo hace PEM. ¿Por qué no?
32. Adivine lo que podría significar la carita : -X (a veces escrita como :-#).
33. ¿Cuánto tiempo se requiere para distribuir el equivalente a un día de noticias a través de un canal satelital de 50 Mbps?
34. ¿Cuáles de los comandos listados en la figura 7-56 son redundantes en teoría?
35. Una red grande consiste en una matriz de máquinas $n \times n$. Todos los nodos internos tienen cuatro vecinos; los de las orillas (esquinas) tienen tres (dos) vecinos. Si se publica un artículo de m bytes en alguna máquina mediante NNTP, ¿cuántos bytes de ancho de banda se consumen para hacerlo llegar a todas las demás máquinas (ignorando la carga extra NNTP y contando sólo los bytes de mensaje)?
36. Repita el problema anterior, pero calcule ahora el ancho de banda aproximado que se necesitará para distribuir el mensaje usando una lista de correo. ¿Qué tanto mayor es respecto del problema anterior?
37. Al enviarse páginas de Web, tienen como prefijo cabeceras MIME. ¿Por qué?
38. ¿Cuándo se requieren visores externos? ¿Cómo sabe el visualizador cuál debe usar?
39. Imagine que alguien del departamento de informática de Stanford acaba de escribir un programa nuevo que quiere distribuir mediante FTP. Pone el programa en el directorio FTP *ftp/pub/freebies/newprog.c*. ¿Cómo será el URL de este programa?
40. En la figura 7-60, se asigna 1 al parámetro *ALT* de la etiqueta . ¿En qué condiciones lo usa el visualizador, y cómo?
41. ¿Cómo se confiere en HTML la posibilidad de hacer clic en una imagen? Dé un ejemplo.
42. Muestre la etiqueta <A> que se necesita para hacer que la cadena "ACM" sea un hipervínculo a <http://www.acm.org>.
43. Diseñe una forma para una compañía nueva, Interburger, que permita ordenar hamburguesas vía Internet. La forma debe incluir el nombre del cliente, su dirección y ciudad, así como una selección de tamaños (gigante o inmensa) y una opción de queso. Las hamburguesas se pagarán al contado a la entrega, por lo que no se requiere información de tarjeta de crédito.
44. Java no tiene estructuras como las de C ni registros como los de Pascal. ¿Hay alguna otra manera de lograr el mismo efecto de agrupar un conjunto de variables diferentes para formar un solo tipo de datos? ¿De haberla, cuál es?
45. Usando las estructuras de datos de la figura 7-75, liste los pasos exactos necesarios para examinar un URL nuevo y comprobar si ya está en *url_table*.

46. Supóngase que, en su esfuerzo por orientarse más al mercado, la KGB se vuelve comercial y contrata a una agencia de publicidad que le diseña una página de Web. Su compañía ha sido contratada como consultor externo para implementarla. Escriba el HTML para producir la siguiente página de Web.

BIENVENIDOS A LA PÁGINA BASE DE WWW DE LA KGB

En virtud de su reciente privatización, la KGB se complace en anunciar la disponibilidad comercial de muchos artículos y servicios de calidad previamente disponibles sólo para los grandes gobiernos.

[Precios competitivos! Servicio discreto asegurado]

- Productos
 - Armamento nuclear (pequeño, mediano, grande, Jumbo)
 - Satélites espía (vigile a sus vecinos)
 - Aeronaves supersónicas de bajo perfil de radar (sobrevele las casas de sus amigos sin ser visto)
- Servicios
 - Colocación de espías en la organización que deseas
 - Golpes de estado (corporativos y gubernamentales)
 - Asistencia para establecer su propio laboratorio de armas bacteriológicas
- Especiales con descuento
 - Obras seleccionadas de Felix Dzerzhinsky (edición limitada)
 - Fotografías aéreas de Algnistán (1984)
 - Tanques búlgaros de calidad (95% de descuento)

Webmaster@kgb.ru

47. En C y C++, el tamaño de un entero no está especificado por el lenguaje. En Java, sí lo está. Cite un argumento a favor del enfoque de C y otro a favor del enfoque de Java.
48. Suponga que la Web contiene 10 millones de páginas, cada una con un promedio de 10 hipervínculos. Para traer una página se requieren 100 msec en promedio. ¿Cuál es el tiempo máximo para indizar la Web completa?
49. Un disco compacto contiene 650 MB de datos. ¿Se usa compresión en los CD de audio? Explique su razonamiento.
50. ¿Cuál es la tasa de bits para transmitir color VGA sin compresión con 8 bits/píxel a 40 marcos/seg?
51. En la figura 7-76(c) ocurre ruido de cuantización debido al uso de muestras de 3 bits. La primera muestra, en 0, es exacta, pero las siguientes no. ¿Cuál es el porcentaje de error para las muestras en 1/32, 2/32 y 3/32 del periodo?
52. Un error de 1 bit en un marco MPEG, ¿puede tener efectos que alcancen más allá del marco en el que ocurrió el error? Explique su respuesta.
53. Considere el ejemplo del servidor de video con 100,000 clientes dado en el texto. Suponga que la mitad de todas las películas se sirven de 8 p.m a 10 p.m. ¿Cuántas películas tiene que transmitir el

servidor a la vez durante este periodo de tiempo? Si cada película requiere 4 Mbps, ¿cuántas conexiones OC-12 a la red necesita el servidor?

54. Suponga que la ley de Zipf se cumple para accesos a un servidor de video de 10,000 películas. Si el servidor contiene las 1000 películas más populares en disco magnético y las 9000 restantes en disco óptico, escriba una expresión para la fracción de las referencias que estarán en disco magnético. Escriba un programa pequeño para evaluar numéricamente esta expresión.
55. Los paquetes PES del MPEG contienen un campo que indica el estado de derechos de autor de la transmisión actual. ¿Para qué podría servir este campo?

8

LISTA DE LECTURAS Y BIBLIOGRAFÍA

Hemos terminado nuestro estudio de las redes de computación, pero éste es sólo el comienzo. No se han tratado muchos temas interesantes con el detalle que merecen, y se han omitido otros por falta de espacio. En este capítulo ofrecemos algunas sugerencias de lectura y una bibliografía, para el beneficio de los lectores interesados en continuar sus estudios de las redes de cómputo.

8.1. SUGERENCIAS PARA LECTURA ADICIONAL

Hay una extensa literatura sobre todos los aspectos de las redes de computación y los sistemas distribuidos. Cuatro publicaciones que con frecuencia publican artículos sobre esta área son *IEEE Transactions on Communications*, *IEEE Journal on Selected Areas in Communications*, *Computer Communication Review* y *Computer Networks and ISDN Systems*. Muchas otras publicaciones también contienen artículos ocasionales sobre el tema.

El IEEE también publica dos revistas, *IEEE Network Magazine* y *IEEE Communications Magazine*, que contienen investigaciones, tutoriales y estudios de casos sobre las redes. La primera está orientada hacia la arquitectura, los estándares y el software, y la última tiende hacia la tecnología de comunicaciones (fibra óptica, satélites, etcétera).

Además, hay varias conferencias anuales o bienales que con frecuencia atraen ponencias sobre redes y sistemas distribuidos, en particular, *SIGCOMM '9x*, *The International Conference on Distributed Computer Systems*, *The Symposium on Operating Systems Principles* y *The N-th*

servidor a la vez durante este periodo de tiempo? Si cada película requiere 4 Mbps, ¿cuántas conexiones OC-12 a la red necesita el servidor?

54. Suponga que la ley de Zipf se cumple para accesos a un servidor de video de 10,000 películas. Si el servidor contiene las 1000 películas más populares en disco magnético y las 9000 restantes en disco óptico, escriba una expresión para la fracción de las referencias que estarán en disco magnético. Escriba un programa pequeño para evaluar numéricamente esta expresión.
55. Los paquetes PES del MPEG contienen un campo que indica el estado de derechos de autor de la transmisión actual. ¿Para qué podría servir este campo?



8

LISTA DE LECTURAS Y BIBLIOGRAFÍA

Hemos terminado nuestro estudio de las redes de computación, pero éste es sólo el comienzo. No se han tratado muchos temas interesantes con el detalle que merecen, y se han omitido otros por falta de espacio. En este capítulo ofrecemos algunas sugerencias de lectura y una bibliografía, para el beneficio de los lectores interesados en continuar sus estudios de las redes de cómputo.

8.1. SUGERENCIAS PARA LECTURA ADICIONAL

Hay una extensa literatura sobre todos los aspectos de las redes de computación y los sistemas distribuidos. Cuatro publicaciones que con frecuencia publican artículos sobre esta área son *IEEE Transactions on Communications*, *IEEE Journal on Selected Areas in Communications*, *Computer Communication Review* y *Computer Networks and ISDN Systems*. Muchas otras publicaciones también contienen artículos ocasionales sobre el tema.

El IEEE también publica dos revistas, *IEEE Network Magazine* y *IEEE Communications Magazine*, que contienen investigaciones, tutoriales y estudios de casos sobre las redes. La primera está orientada hacia la arquitectura, los estándares y el software, y la última tiende hacia la tecnología de comunicaciones (fibra óptica, satélites, etcétera).

Además, hay varias conferencias anuales o bienales que con frecuencia atraen ponencias sobre redes y sistemas distribuidos, en particular, *SIGCOMM '9x*, *The International Conference on Distributed Computer Systems*, *The Symposium on Operating Systems Principles* y *The N-th*

Data Communications Symposium. Además, el IEEE ha publicado varios volúmenes de reimpresiones de investigaciones sobre redes en formato accesible en rústica.

A continuación se presenta una lista de sugerencias para lectura adicional, ordenada según los capítulos de este libro con el que se relacionan.

8.1.1. Introducción y obras generales

Jabbari *et al.*, "Network Issues for Wireless Communication"

Esta introducción a los sistemas de radio celular cubre control de llamadas, enrutamiento, señalización y otros aspectos de los sistemas modernos de comunicación móvil.

Comer, *The Internet Book*

Cualquiera que desee una introducción fácil a Internet debe buscarla aquí. Comer describe la historia, crecimiento, tecnología, protocolos y servicios de Internet en términos que pueden entender los principiantes, pero gran parte del material cubierto por este libro también es de interés para los lectores más técnicos.

Kwok, "A Vision for Residential Broadband Service"

Si desea conocer cómo piensa Microsoft que debe organizarse el video a solicitud, este artículo es para usted. En él, el arquitecto principal de ATM de Microsoft explica el punto de vista de su compañía. En resumen, la idea de Microsoft es: el camino a seguir es ATM a los hogares. Olvidemos todas las soluciones "realistas" (es decir, *ad hoc*), como el ADSL, y hágámoslo bien.

Le Boudec, "The Asynchronous Transfer Mode: A tutorial"

El ATM es una tecnología nueva en crecimiento, y este documento es una introducción detallada a él. Se cubren la capa física, la capa ATM y la capa AAL. Además, la sección final analiza el debate sobre el ATM.

Pahlavan *et al.*, "Trends in Local Wireless Networks"

Las LAN inalámbricas sin duda serán cada vez más importantes en el futuro. En este artículo, los autores estudian lo más avanzado y las tendencias en el uso del espectro y las tecnologías para las LAN inalámbricas.

Siu and Jain, "A Brief Overview of ATM"

Se cubren muchas de las características de los sistemas ATM en este documento introductorio, pero el enfoque es hacia la simulación de LAN y la administración de tráfico; también sirve como introducción a una edición especial de *Computer Communication Review* dedicada a la tecnología ATM.

Stallings, *Data and Computer Communications*, 4th ed.

En lugar de cubrir sólo los modelos de referencia OSI y TCP/IP, Stallings incluye un tercero, el SNA, por si no basta con los otros. Los tres se describen en el capítulo 10.

8.1.2. La capa física

Awdeh and Mouttaf, "Survey of ATM Switch Architectures"

Cualquier interesado en aprender más sobre el diseño de los commutadores ATM debe buscar aquí. Tras una introducción a los commutadores en general y a las estrategias de *buffer*, los autores estudian muchos tipos de commutadores de barras cruzadas, trayectoria disjunta y banianos. El artículo también proporciona más de 200 referencias a otros artículos.

Bellamy, *Digital Telephony*

Todo lo que siempre quiso saber sobre el sistema telefónico y más, contenido en este libro de referencia. Particularmente interesantes son los capítulos sobre transmisión y multiplexión, commutación digital, fibra óptica e ISDN.

De Prycker, *Asynchronous Transfer Mode*, 2nd ed.

El capítulo 4 contiene abundante información sobre los commutadores ATM. Los principios se ilustran mediante numerosos ejemplos, incluidos los commutadores por *knockout*, Roxanne, Coprin y Athena.

Held, *The Complete Modem Reference*, 2nd ed.

Aquí está todo lo que conceiblemente puede querer saber sobre los módems, desde las reglas de estandarización de los gobiernos de Estados Unidos y Canadá, a las técnicas y estándares de modulación, y la manera de identificar problemas en un módem enfermo.

IEEE *Communications Mag.*, Jan. 1995, "Wireless Personal Communications"

Esta edición especial contiene siete artículos sobre diferentes aspectos de la comunicación personal inalámbrica. En conjunto, los artículos cubren propagación, métodos de acceso, principios de los receptores, aspectos de los sistemas y asuntos sobre redes.

Metcalfe, "Computer/Network Interface Design: Lessons from Arpanet & Ethernet"

Aunque los ingenieros han estado construyendo interfaces de red durante décadas, con frecuencia nos preguntamos si han aprendido algo de toda esta experiencia. En este artículo, el diseñador de Ethernet describe la manera de construir una interfaz de red, y lo que se debe hacer una vez construida. No hace concesiones, indicando lo que hizo mal y lo que hizo bien.

Padgett *et al.*, "Overview of Wireless Personal Communications"

Introducción a los sistemas celulares e inalámbricos de comunicación, y comparación entre ambos. Se cubre el estándar de Estados Unidos y el europeo.

Palais, *Fiber Optic Communication*, 3rd ed.

Los libros sobre fibra óptica tienden a estar orientados al especialista, pero éste es más accesible que la mayoría; cubre las guías de ondas, fuentes de luz, detectores de luz, acopladores, modulación, ruido y muchos otros temas.

Pandya, "Emerging Mobile and Personal Communications Systems"

Para una introducción corta y agradable a los sistemas de comunicación personales, vale la pena ver este artículo. Una de las nueve páginas contiene una lista de 70 siglas usadas en las otras ocho páginas.

Partridge, *Gigabit Networking*

Además de describir varios tipos de conmutadores ATM, el capítulo 5 también compara buffers de entrada y buffers de salida, y deriva fórmulas para el desempeño de ambos.

Spragins et al., *Telecommunications Protocols and Design*

El capítulo 2 contiene una buena introducción a la tecnología de transmisión, incluidos alambres de cobre, fibra óptica, radio celular y satélites. También tiene estudios extensos de los límites de Nyquist y de Shannon, y comenta sus implicaciones.

8.1.3. La capa de enlace de datos

Black, *Data Link Protocols*

Éste es un libro completo dedicado a la capa de enlace de datos; tiene un enfoque práctico, con una gran cantidad de material sobre HDLC, LLC, PPP y otros protocolos comercialmente importantes.

Holzmann, *Design and Validation of Computer Protocols*

Los lectores interesados en los aspectos más formales de los protocolos de enlace de datos (y similares) deben leer este libro. En él se cubren la especificación, modelado, diseño correcto y pruebas de tales protocolos.

Spragins et al., *Telecommunications Protocols and Design*

Los lectores interesados en aprender más sobre códigos de detección de errores y corrección de errores deben ver el capítulo 6 de este libro. La obra también cubre los principios de los protocolos elementales de enlace de datos más o menos en el mismo nivel que nuestro libro. El capítulo 7 continúa el estudio y analiza con detalle varios protocolos de enlace de datos.

Walrand, *Communication Networks: A First Course*

El capítulo 4 cubre los protocolos de enlace de datos, con énfasis en el análisis de desempeño. También se tratan los enfoques de máquina de estados finitos y de red de Petri para el diseño correcto de protocolos.

8.1.4. La subcapa de acceso al medio

Abeymundara and Kamal, "High-Speed Local Area Networks and Their Performance"

Dado que las LAN de alta velocidad son de interés precisamente por su alta velocidad, es bienvenido un artículo en el que se estudia y analiza el desempeño. En éste, el enfoque es en los

diferentes tipos de LAN de bus, anillo, árbol y estrella, así como sus características de retardo y de utilización.

Jain, *FDDI Handbook—High-Speed Networking Using Fiber and other Media*

Para un tratamiento detallado del FDDI (incluidos agradables tutoriales sobre fibra óptica y SONET), este libro es una buena decisión. Además de secciones grandes sobre el hardware y software de FDDI, el libro tiene una sección sobre el desempeño e incluso consejos para la compra de cables de fibra óptica.

Perlman, *Interconnections: Bridges and Routers*

El libro de Perlman es el lugar para buscar un informado pero entretenido tratamiento de los puentes (y enrutadores). El autor diseñó los algoritmos usados en el árbol de extensión IEEE 802 al igual que los algoritmos de enrutamiento del DECnet, y evidentemente es un experto en el tema.

Stallings, *Local and Metropolitan Area Networks*, 4th ed.

Las tres LAN IEEE 802 forman el núcleo de este libro, pero también hay material sobre otras LAN y MAN.

Walrand, *Communication Networks: A First Course*

Al igual que el libro de Stallings mencionado antes, el capítulo 5 de éste cubre el material 802 básico, más FDDI y DQDB. La orientación es hacia el análisis del desempeño de los protocolos.

8.1.5. La capa de red

Comer, *Internetworking with TCP/IP*, Vol. 1, 3rd ed.

Comer ha escrito el libro definitivo sobre la serie de protocolos TCP/IP. Los capítulos 4 a 11 tratan el IP y protocolos relacionados de la capa de red. Los otros capítulos se encargan principalmente de las capas superiores, y también valen la pena.

Huitema, *Routing in the Internet*

Si quiere saber todo lo que hay por saber sobre el enrutamiento en Internet, éste es el libro para usted. Se tratan con lujo de detalle tanto los algoritmos pronunciables (por ejemplo, RIP, CIDR y MBONE), como los impronunciables (por ejemplo, OSPF, IGRP, EGP y BGP). También están aquí las características nuevas, como multitransmisión, IP móvil y reservación de recursos.

Perlman, *Interconnections: Bridges and Routers*

En el capítulo 9, Perlman describe muchos de los asuntos implicados en diseño de algoritmos de enrutamiento unitransmisión y multitransmisión, tanto en las WAN como en las redes de varias LAN, así como su implementación en varios dispositivos. Es claro que al autor le importa el tema, habiendo dado a la sección 9.13.10 el título "Mi opinión sobre la multitransmisión de capa de red estilo IP".

Sterbenz *et al.*, "Report on the IEEE ComSoc Gigabit Networking Workshop"

Antes de que sean utilizables las redes de gigabits, deben resolverse cuestiones básicas. Una que es crucial es si estas redes usarán ATM, TCP/IP o ambos. Para entender mejor estos temas, el IEEE organizó un taller de trabajo en abril de 1995, del que se presenta un resumen aquí. Vale la pena leer la crítica del ATM que hace Schulzrinne si usted cree que el ATM es la solución a los problemas de telecomunicación del mundo.

Stevens, *TCP/IP Illustrated*, Vol. 1

Los capítulos 3 a 10 son un estudio detallado del IP y protocolos relacionados (ARP, RARP e ICMP), ilustrados con ejemplos.

Yang and Reddy, "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks"

Los autores han diseñado una taxonomía para los algoritmos de control de congestionamiento. Las categorías principales son ciclo abierto con control de origen, ciclo abierto con control de destino, ciclo abierto con realimentación explícita y ciclo cerrado con realimentación implícita. Los autores usan esta taxonomía para describir y clasificar 23 algoritmos existentes.

8.1.6. La capa de transporte

Comer, *Internetworking with TCP/IP*, Vol. 1, 3rd ed.

Como se mencionó antes, Comer ha escrito el trabajo definitivo sobre la serie de protocolos TCP/IP.

Mogul, "IP Network Performance"

A pesar del título de este artículo, trata más el desempeño del TCP y las redes en general, que el desempeño del IP en particular; está lleno de pautas y reglas empíricas útiles.

Stallings, *Data and Computer Communications*, 4th ed.

El capítulo 12 trata los protocolos de transporte y cubre los servicios y mecanismos de manera abstracta, al igual que, con detalle, los protocolos de transporte OSI y TCP.

Stevens, *TCP/IP Illustrated*, Vol. 1

Los capítulos 17-24 dan un tratamiento detallado del TCP, ilustrado con ejemplos.

8.1.7. La capa de aplicación

Anderson, R., "Why Cryptosystems Fail"

Según Anderson, la seguridad de los sistemas bancarios es mala, pero no debido a que intrusos astutos violen el DES desde sus PC. Los problemas reales van desde los empleados

deshonestos (un oficinista que cambia la dirección de correo de un cliente a la suya propia para interceptar el número de tarjeta bancaria y PIN) a los errores de programación (dar a todos los clientes el mismo número de PIN). Lo que es especialmente interesante es la respuesta que dan los bancos al confrontar un error: nuestros sistemas son perfectos y, por tanto, todos los errores deben ser causados por errores del cliente o un fraude.

Berghel, "The Client Side of the Web"

Introducción informal a los visualizadores de la Web y a las características que pueden apoyar. Los temas principales son manejo de HTML/HTTP, desempeño, capacidad de reconfiguración, integración con los equipos de escritorio y herramientas de navegación. Se hace la comparación de estos aspectos entre nueve visualizadores populares.

Berners-Lee *et al.*, "The World Wide Web"

Una perspectiva de la Web y hacia dónde va, desde el punto de vista de la persona que la inventó. El artículo se enfoca hacia la arquitectura de la Web, HTTP y HTML, así como las direcciones futuras.

Carl-Mitchell and Quarterman, *Practical Internetworking with TCP/IP and UNIX*

El capítulo 5 presenta una buena introducción a los nombres y el DNS, incluidas autoridades de nombres, arquitectura operativa y base de datos DNS.

Choudhury *et al.*, "Copyright Protection for Electronic Publishing on Computer Networks"

Aunque numerosos libros y artículos describen los algoritmos criptográficos, pocos describen la manera en que pueden usarse para evitar que los usuarios distribuyan más los documentos que tienen permitido descifrar. Este artículo describe una variedad de mecanismos que podrían ayudar a proteger los derechos de autor en el área electrónica.

Furht *et al.*, "Design Issues for Interactive Television Systems"

El video a solicitud presenta muchos problemas técnicos complicados que se relacionan con arquitectura del sistema, topología de red, diseño de servidores y diseño de cajas de control. En este artículo, los autores presentan un tutorial sobre algunos de los problemas clave y algunas soluciones que se están investigando.

Handley and Crowcroft, *The World Wide Web—Beneath the Surf*

Aunque el 99% de los libros sobre la WWW simplemente dicen cómo usar un visualizador o listar algunos URL interesantes, éste explica el funcionamiento interno de la Web. El lado del cliente, el del servidor y el HTML se explican sin indigestar al lector.

Kaufman *et al.*, *Network Security*

Este informado y frecuentemente ingenioso libro es el primer lugar para buscar más información sobre seguridad en las redes. Se explican con detalle algoritmos y protocolos de clave secreta y

pública, parcialización de mensajes, validación de identificación, Kerberos y correo electrónico. Las mejores partes son los debates entre los autores (e inclusive con ellos mismos), etiquetados con subíndices, como: "Yo₂ no pudo hacer que yo₁ fuera más específico..."

Kumar, *MBone: Interactive Multimedia on the Internet*

La cubierta de este libro dice: "Descubra la manera de difundir, anunciar y presentar sus productos en Internet". Afortunadamente, el tema no se menciona en ningún otro lugar del libro. Lo que sí se cubre es la arquitectura e implementación del MBone, incluyendo mucho material sobre su funcionamiento y su uso.

Memeth et al., *UNIX System Administration Handbook*

El capítulo 16 es una larga introducción al DNS. Aborda todos los detalles escabrosos, ilustrando los diferentes archivos y registros de recursos con numerosos ejemplos. También se cubren con algún detalle los programas y otras herramientas usados para administrar un servidor DNS.

Rose, *The Internet Message*

Si le gusta su correo electrónico servido con una pizca de iconoclastia, este libro es una buena apuesta. El autor no duda en levantar la voz de vez en cuando para anunciar lo que está mal en el mundo. Cuando uno logra entrar en el tema, su gusto no resulta malo.

Schneier, *Applied Cryptography*, 2nd ed.

Este compendio monumental es la peor pesadilla de la NSA: un solo libro que describe todos los algoritmos criptográficos conocidos. Para empeorar (o mejorar, según su punto de vista) las cosas, el libro contiene la mayoría de los algoritmos como programas ejecutables (en C). Además, se proporcionan más de 1600 referencias a la literatura criptográfica. Si *en realidad* quiere mantener secretos sus archivos, lea este libro.

Steinmetz and Nahrstedt, *Multimedia: Computing, Communications and Applications*

Aunque algo caótico, este libro cubre mucho terreno de los multimedia. Los temas tratados con detalle incluyen audio, imágenes fijas, imágenes en movimiento, compresión, almacenamiento óptico, sistemas operativos multimedia, redes, hipertexto, sincronización de corrientes y aplicaciones multimedia.

Van der Linden, *Just Java*

Cuando el capítulo 1 de un libro se titula "Entra en mi estancia, le dijo la araña a la mosca", muy probablemente se trata de un cuento de hadas o un libro sobre la WWW. Éste es sobre la Web, específicamente sobre el lenguaje Java y su entorno. Para la gente que quiere jugar con Java, el libro incluye el sistema Java completo en CD-ROM.

8.2. BIBLIOGRAFÍA ALFABÉTICA

- ABEYSUNDARA, B.W., and KAMAL, A.E.: "High-Speed Local Area Networks and Their Performance" *Computing Surveys*, vol. 23, pp. 221-264, June 1991.
- ABRAMSON, N.: "Development of the ALOHANET," *IEEE Trans. on Information Theory*, vol. IT-31, pp. 119-123, March 1985.
- ADAM, J.A.: "Privacy and Computers," *IEEE Spectrum*, vol. 32, pp. 46-52, Dec. 1995.
- ADAMS, N., GOLD, R., SCHILIT, B.N., TSO, M.M., and WANT, R.: "An Infrared Network for Mobile Computers," *Proc. USENIX Mobile and Location-Independent Computing Symposium*, USENIX, pp. 41-51, 1993.
- ANDERSON, R.J.: "Why Cryptosystems Fail," *Commun. of the ACM*, vol. 37, pp. 32-40, Nov. 1994.
- ARMBRUSTER, H.: "The Flexibility of ATM: Supporting Future Multimedia and Mobile Communications," *IEEE Commun. Magazine*, vol. 33, pp. 76-84, Feb. 1995.
- ARMITAGE, G.J., and ADAMS, K.M.: "How Efficient is IP over ATM Anyway?" *IEEE Network Magazine*, vol. 9, pp. 18-26, Jan./Feb. 1995.
- ARNOLD, K., and GOSLING, J.: *The Java Programming Language*; Reading, MA: Addison-Wesley, 1996.
- AT&T and BELLCORE: "Observations of Error Characteristics of Fiber Optic Transmission Systems," CCITT SG XVIII, San Diego, Jan. 1989.
- AWDEH, R.Y., and MOUFTAH, H.T.: "Survey of ATM Switch Architectures," *Computer Networks and ISDN Systems*, vol. 27, pp. 1567-1613, Nov. 1995.
- BAKNE, A., and BADRINATH, B.R.: "I-TCP: Indirect TCP for Mobile Hosts," *Proc. Fifteenth Int'l. Conf. on Distr. Computer Systems*, IEEE, pp. 136-143, 1995.
- BALAKRISHNAN, H., SESHAN, S., and KATZ, R.H.: "Improving Reliable Transport and Hand-off Performance in Cellular Wireless Networks," *Proc. ACM Mobile Computing and Networking Conf.*, ACM, pp. 2-11, 1995.
- BALLARDIE, T., FRANCIS, P., and CROWCROFT, J.: "Core Based Trees (CBT)," *Proc. SIGCOMM '93 Conf.*, ACM, pp. 85-95, 1993.
- BANTZ, D.F., and BAUCHOT, F.J.: "Wireless LAN Design Alternatives," *IEEE Network Magazine*, vol. 8, pp. 43-53, March/April, 1994.
- BARANSEL, C., DOBOSIEWICZ, W., and GBURZYNSKI, P.: "Routing in Multihop Packet Switching Networks: Gb/s Challenge," *IEEE Network Magazine*, vol. 9, pp. 38-61, May/June, 1995.
- BARLOW, J.P.: "Property and Speech: Who Owns What You Say in Cyberspace," *Commun. of the ACM*, vol. 38, pp. 19-22, Dec. 1995.
- BATCHER, K.E.: "Sorting Networks and Their Applications," *Proc. AFIPS Spring Joint Computer Conf.*, vol. 32, pp. 307-315, 1968.

- BATES, R.J.: *Wireless Networked Communications*, New York: McGraw-Hill, 1994.
- BERGHEL, H.L.: "The Client Side of the Web," *Commun. of the ACM*, vol. 39, pp. 33-40, Jan. 1996.
- BELLAMY, J.: *Digital Telephony*, New York: John Wiley, 1991.
- BELLMAN, R.E.: *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.
- BELSNES, D.: "Flow Control in the Packet Switching Networks," *Communications Networks*, Uxbridge, England: Online, pp. 349-361, 1975.
- BERNERS-LEE, T., CAJILAU, A., LOUTONEN, A., NIELSEN, H.F., and SECRET, A.: "The World Wide Web," *Commun. of the ACM*, vol. 37, pp. 76-82, Aug. 1994.
- BERTSEKAS, D., and GALLAGER, R.: *Data Networks*, 2nd ed., Englewood Cliffs, NJ: Prentice Hall, 1992.
- BHARGHAVAN, V., DEMERS, A., SHENKER, S., and ZHANG, L.: "MACAW: A Media Access Protocol for Wireless LANs," *Proc. SIGCOMM '94 Conf.*, ACM, pp. 212-225, 1994.
- BIHAM, E., and SHAMIR, A.: *Differential Cryptanalysis of the Data Encryption Standard*, New York: Springer-Verlag, 1993.
- BINDER, R.: "A Dynamic Packet Switching System for Satellite Broadcast Channels," *Proc. Int'l. Conf. on Commun.*, pp. 41-1 to 41-5a, 1975.
- BLACK, U.D.: *TCP/IP and Related Protocols*, New York: McGraw-Hill, 1995.
- BLACK, U.D.: *Emerging Communications Technologies*, Englewood Cliffs, NJ: Prentice Hall, 1994.
- BLACK, U.D.: *Data Link Protocols*, Englewood Cliffs, NJ: Prentice Hall, 1993.
- BLAZE, M.: "Protocol Failure in the Escrowed Encryption Standard," *Proc. Second ACM Conf. on Computer and Commun. Security*, ACM, pp. 59-67, 1994.
- BOGINENI, K., and SIVALINGAM, K.M.: "Low-Complexity Multiple Access Protocols for Wavelength-Division Multiplexed Photonic Networks," *IEEE Journal on Selected Areas in Commun.*, vol. 11, pp. 590-604, May 1993.
- BONOMI, F., and FENDICK, K.W.: "The Rate-Based Flow Control Framework for the Available Bit-rate ATM Service," *IEEE Network Magazine*, vol. 9, pp. 25-39, March/April 1995.
- BOWMAN, C.M., DANZIG, P.B., HARDY, D.R., MANBER, U., and SCHWARTZ, M.F.: "The Harvest Information Discovery and Access System," *Computer Networks and ISDN Systems*, vol. 28, pp. 119-125, Dec. 1995.
- BOWMAN, C.M., DANZIG, P.B., MANBER, U., and SCHWARTZ, M.F.: "Scalable Internet Resource Discovery: Research Problems and Approaches," *Commun. of the ACM*, vol. 37, pp. 98-107, Aug. 1994.
- BRAKMO, L.S., O'MALLEY, S.W., and PETERSON, L.L.: "TCP Vegas: New Techn. for Congestion Detection and Avoidance," *Proc. SIGCOMM '94 Conf.*, ACM, pp. 24-35, 1994.

- BROADHEAD, M.A. and OWEN, C.B.: "Direct Manipulation of MPEG Compressed Digital Audio," *Proc. of ACM Multimedia '95*, ACM, pp. 499-507, 1995.
- BROWN, L., KWAN, M., PIEPRZYK, J., and SEBERRY, J.: "Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI," *ASIACRYPT '91 Abstracts*, pp. 25-30, 1991.
- BUFORD, J.F.K. (Ed.): *Multimedia Systems*, Reading, MA: Addison-Wesley, 1994.
- BURROWS, M., ABADI, M., and NEEDHAM, R.M.: "A Logic of Authentication," DEC System Research Center Report, Feb. 1990.
- CAMPBELL, A., COULSON, G., and HUTCHISON, D.: "A Quality of Service Architecture," *Computer Commun. Rev.*, vol. 24, pp. 6-27, April 1994.
- CAMPIONE, M., and WALRATH, K.: *The Java Language Tutorial: Object-Oriented Programming for the Internet*, Reading, MA: Addison-Wesley, 1996.
- CAPETANAKIS, J.I.: "Tree Algorithms for Packet Broadcast Channels," *IEEE Trans. on Information Theory*, vol. IT-25, pp. 505-515, Sept. 1979.
- CARL-MITCHELL, S., and QUARTERMAN, J.S.: *Practical Internetworking with TCP/IP and UNIX*, Reading, MA: Addison-Wesley, 1993.
- CATLETT, C.E.: "In Search of Gigabit Applications," *IEEE Commun. Magazine*, vol. 30, pp. 42-51, April 1992.
- CERF, V., and KAHN, R.: "A Protocol for Packet Network Interconnection," *IEEE Trans. on Commun.*, vol. COM-22, pp. 637-648, May 1974.
- CHANDRANMENON, G.P., and VARGHESE, G.: "Trading Packet Headers for Packet Processing," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 162-173, 1995.
- CHANG, Y.-H., COGGINS, D., PITI, D., SKELLERN, D., THAPAR, M., and VENKATRAMAN, C.: "An Open-System Approach to Video-on-Demand," *IEEE Commun. Magazine*, vol. 32, pp. 68-80, May 1994.
- CHAO, J.J., GHOSAL, D., SAHA, D., and TRIPATHI, S.K.: "IP on ATM Local Area Networks," *IEEE Commun. Magazine*, vol. 32, pp. 52-59, Aug. 1994.
- CHAPMAN, D.E., and ZWICKY, E.D.: *Building Internet Firewalls*, Sebastopol, CA: O'Reilly, 1995.
- CHEN, K.-C.: "Medium Access Control of Wireless LANs for Mobile Computing," *IEEE Network Magazine*, vol. 8, pp. 50-63, Sept./Oct. 1994.
- CHEN, M., and YUM, T.-S.: "A Conflict-Free Protocol for Optical WDMA Networks," *Proc. Globecom '91*, pp. 1276-1281, 1991.
- CHEN, W.Y., and WARING, D.L.: "Applicability of ADSL to Support Video Dial Tone in the Copper Loop," *IEEE Commun. Magazine*, vol. 32, pp. 102-106, May 1994.
- CHERITON, D., and WILLIAMSON, C.: "VMTP as the Transport Layer for High-Performance Distributed Systems," *IEEE Commun. Magazine*, vol. 27, pp. 37-44, June 1989.

- CHERVENAK, A.L.: *Tertiary Storage: An Evaluation of New Applications*, Ph.D. thesis, CSD, Univ. of California at Berkeley, 1994.
- CHERVENAK, A.L., PATTERSON, D.A., and KATZ, R.H.: "Choosing the Best Storage System for Video Service," *Proc. of ACM Multimedia '95*, ACM, pp. 109-119, 1995.
- CHESSON, G.L.: "XTP/PE Design Considerations," *IFIP Workshop on Protocols for High-Speed Networks*, IFIP, pp. 27-33, 1989.
- CHESWICK, W.R. and BELLOVIN, S.M.: *Firewalls and Interwalls—Repelling the Wily Hacker*, Reading, MA: Addison-Wesley, 1994.
- CHOUDBURY, A.K., MAXEMCHUK, N.F., PALL, S., and SCHULZRINNE, H.G.: "Copyright Protection for Electronic Publishing on Computer Networks," *IEEE Network Magazine*, vol. 9, pp. 12-20, May/June, 1995.
- CLARK, D.D.: "The Design Philosophy of the DARPA Internet Protocols," *Proc. SIGCOMM '88 Conf.*, ACM, pp. 106-114, 1988.
- CLARK, D.D.: "NETBLT: A Bulk Data Transfer Protocol," RFC 998, 1987.
- CLARK, D.D.: "Window and Acknowledgement Strategy in TCP," RFC 813, July 1982.
- CLARK, D.D., DAVIE, B.S., FARBER, D.J., GOPAL, I.S., KADABA, B.K., SINCOSKIE, W.D., SMITH, J.M., and TENNENHOUSE, D.L.: "The Aurora Gigabit Testbed," *Computer Networks and ISDN Systems*, vol. 25, pp. 599-621, Jan. 1993.
- CLARK, D.D., JACOBSON, V., ROMKEY, J., and SALWEN, H.: "An Analysis of TCP Processing Overhead," *IEEE Commun. Magazine*, vol. 27, pp. 23-29, June 1989.
- CLARK, D.D., LAMBERT, M., and ZHANG, L.: "NETBLT: A High Throughput Transport Protocol," *Proc. SIGCOMM '87 Conf.*, ACM, pp. 353-359, 1987.
- CLOS, C.: "A Study of Non-Blocking Switching Networks," *Bell System Tech. J.*, vol. 32, pp. 406-424, March 1953.
- COMER, D.E.: *The Internet Book*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- COMER, D.E.: *Internetworking with TCP/IP*, vol. 1, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1995.
- COOK, A., and STERN, J.: "Optical Fiber Access—Perspectives Toward the 21st Century," *IEEE Commun. Magazine*, vol. 32, pp. 78-86, Feb. 1994.
- COOPER, E.: *Broadband Network Technology*, Englewood Cliffs, NJ: Prentice Hall, 1986.
- COULOURIS, G.E., DOLLMORE, J., and KINDBERG, T.: *Distributed Systems Concepts and Design*, 2nd ed. Reading, MA: Addison-Wesley, 1994.
- CRESPO, P.M., HONIG, M.L., and SALEHI, J.A.: "Spread-Time Code-Division Multiple Access," *IEEE Trans. on Commun.*, vol. 43, pp. 2139-2148, June 1995.
- CRONIN, W.J., HUTCHINSON, J.D., RAMAKRISHNAN, K.K., and YANG, H.: "A Comparison of High Speed LANs," *Proc. Nineteenth Conf. on Local Computer Networks*, IEEE, pp. 40-49, 1994.

- CROWCROFT, J., WANG, Z., SMITH, A., and ADAMS, J.: "A Rough Comparison of the IETF and ATM Service Models," *IEEE Network Magazine*, vol. 9, pp. 12-16, Nov./Dec. 1995.
- CROWTHER, W., RETTBERG, R., WALDEN, D., ORNSTEIN, S., and HEART, F.: "A System for Broadcast Communication: Reservation-Aloha," *Proc. Sixth Hawaii Int. Conf. System Sci.*, pp. 371-374, 1973.
- CUSICK, T.W., and WOOD, M.C.: "The REDOC-II Cryptosystem," *Advances in Cryptology—CRYPTO '90 Proceedings*, NY: Springer-Verlag, pp. 545-563, 1991.
- DAGDEVIREN, N., NEWELL, J.A., SPINDEL, L.A., and STEFANICK, M.J.: "Global Networking with ISDN," *IEEE Commun. Magazine*, vol. 32, pp. 26-32, June 1994.
- DANSKIN, J.M., DAVIS, G.M., and SONG, X.: "Fast Lossy Internet Image Transmission," *Proc. of ACM Multimedia '95*, ACM, pp. 321-332, 1995.
- DANTHINE, A.A.S.: "Protocol Representation with Finite-State Models," *IEEE Trans. on Commun.*, vol. COM-28, pp. 632-643, April 1980.
- DAVIS, P.T., and MCGUFFIN, C.R.: *Wireless Local Area Networks*, New York: McGraw-Hill, 1995.
- DAY, J.D.: "The (Un)Revised OSI Reference Model," *Computer Commun. Rev.*, vol. 25, pp. 39-55, Oct. 1995.
- DAY, J.D., and ZIMMERMANN, H.: "The OSI Reference Model," *Proc. of the IEEE*, vol. 71, pp. 1334-1340, Dec. 1983.
- DE JONGE, W., and CHAUM, D.: "Some Variations on RSA Signatures and Their Security," in *Advances in Cryptology—CRYPTO '86 Proceedings*, Odlyzko, A.M. (Ed.), New York: Springer Verlag, 1987.
- DE PRYCKER, M.: *Asynchronous Transfer Mode*, 2nd. ed., New York: Ellis Horwood, 1993.
- DEAN, D., and WALLACH, D.S.: "Security Flaws in the HotJava Web Browser," Technical Report 502, Dept. of Computer Science, Princeton Univ., 1995.
- DEERING, S.E.: "SIP: Simple Internet Protocol," *IEEE Network Magazine*, vol. 7, pp. 16-28, May/June 1993.
- DEERING, S.E., and CHERITON, D.R.: "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. on Computer Systems*, vol. 8, pp. 85-110, May 1990.
- DEERING, S.E., ESTRIN, D., FARINACCI, D., JACOBSON, V., LIU, C.-G., and WEI, L.: "An Architecture for Wide-Area Multicast Routing," *Proc. SIGCOMM '94 Conf.*, ACM, pp. 126-135, 1994.
- DELODDERE, D., VERBIEST, W., and VERHILLE, H.: "Interactive Video on Demand," *IEEE Commun. Magazine*, vol. 32, pp. 82-88, May 1994.
- DEMERS, A., KESHAV, S., and SHENKER, S.: "Analysis and Simulation of a Fair Queueing Algorithm," *Internet: Research and Experience*, vol. 1, pp. 3-26, Sept. 1990.

- DENNING, D.E., and SACCO, G.M.: "Timestamps in Key Distribution Protocols," *Commun. of the ACM*, vol. 24, pp. 533-536, Aug. 1981.
- DIFFIE, W., and HELLMAN, M.E.: "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," *IEEE Computer Magazine*, vol. 10, pp. 74-84, June 1977.
- DIFFIE, W., and HELLMAN, M.E.: "New Directions in Cryptography," *IEEE Trans. on Information Theory*, vol. IT-22, pp. 644-654, Nov. 1976.
- DIJKSTRA, E.W.: "A Note on Two Problems in Connexion with Graphs," *Numer. Math.*, vol. 1, pp. 269-271, Oct. 1959.
- DIRVIN, R.A., and MILLER, A.R.: "The MC68824 Token Bus Controller: VLSI for the Factory LAN," *IEEE Micro Magazine*, vol. 6, pp. 15-25, June 1986.
- DIXIT, S., and SKELLY, P.: "MPEG-2 over ATM for Video Dial Tone Network," *IEEE Network Magazine*, vol. 9, pp. 30-40, Sept./Oct. 1995.
- DIXON, R.C.: "Lore of the Token Ring," *IEEE Network Magazine*, vol. 1, pp. 11-18, Jan./Feb. 1987.
- DOERRINGER, W.A., DYKEMAN, D., KAISERSWERTH, M., MEISTER, B.W., RUDIN, H., and WILLIAMSON, R.: "A Survey of Light-Weight Transport Protocols for High-Speed Networks," *IEEE Trans. on Commun.*, vol. 38, pp. 2025-2039, Nov. 1990.
- DORFMAN, R.: "Detection of Defective Members of a Large Population," *Annals Math. Statistics*, vol. 14, pp. 436-440, 1943.
- ECKBERG, A.E.: "B-ISDN/ATM Traffic and Congestion Control," *IEEE Network Magazine*, vol. 6, pp. 28-37, Sept./Oct. 1992.
- ECKBERG, A.E., DOSHI, B.T., and ZOCCOLILLO, R.: "Controlling Congestion in B-ISDN/ATM: Issues and Strategies," *IEEE Commun. Magazine*, vol. 29, pp. 64-70, Sept. 1991.
- EDWARDS, A., and MUIR, S.: "Experience Implementing a High-Performance TCP in User-Space," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 197-205, 1995.
- EL GAMAL, T.: "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Trans. on Information Theory*, vol. IT-31, pp. 469-472, July 1985.
- ERIKSSON, H.: "MBone: The Multicast Backbone," *Commun. of the ACM*, vol. 37, pp. 54-60, Aug. 1994.
- ESTRIN, D., REKHTER, Y., and HOTZ, S.: "Scalable Inter-Domain Routing Architecture," *Proc. SIGCOMM '92 Conf.*, ACM, pp. 40-52, 1992.
- FEIG, E., and WINOGRAD, S.: "Fast Algorithms for Discrete Cosine Transformations," *IEEE Trans. on Signal Processing*, vol. 40, Sept. 1992.
- FEIT, S.: *SNMP—A Guide to Network Management*, New York: McGraw-Hill, 1995.
- FIORINI, D., CHIANI, M., TRALLI, V., and SALATI, C.: "Problems with HDLC," *Computer Commun. Rev.*, vol. 25, pp. 61-80, Oct. 1995.

- FISCHER, W., WALLMEIER, E., WORSTER, T., DAVIS, S.P., HAYTER, A.: "Data Communications Using ATM: Architectures, Protocols, and Resource Management," *IEEE Commun. Magazine*, vol. 32, pp. 24-33, Aug. 1994.
- FLOYD, S., and JACOBSON, V.: "Random Early Detection for Congestion Avoidance," *IEEE/ACM Trans. on Networking*, vol. 1, pp. 397-413, Aug. 1993.
- FLUCKIGER, F.: *Understanding Networked Multimedia*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- FORD, L.R., Jr., and FULKERSON, D.R.: *Flows in Networks*, Princeton, NJ: Princeton University Press, 1962.
- FORD, P.S., REKHTER, Y., and BRAUN, H.-W.: "Improving the Routing and Addressing of IP," *IEEE Network Magazine*, vol. 7, pp. 10-15, May/June 1993.
- FORMAN, G.H., and ZAHORIAN, J.: "The Challenges of Mobile Computing," *IEEE Computer Magazine*, vol. 27, pp. 38-47, April 1994.
- FRANCIS, P.: "A Near-Term Architecture for Deploying Pip," *IEEE Network Magazine*, vol. 7, pp. 30-37, May/June 1993.
- FRASER, A.G.: "Early Experiments with Asynchronous Time Division Networks," *IEEE Network Magazine*, vol. 7, pp. 12-27, Jan./Feb. 1993.
- FRASER, A.G.: "Towards a Universal Data Transport System," in *Advances in Local Area Networks*, Kummerle, K., Tobagi, F., and Limb, J.O. (Eds.), New York: IEEE Press, 1987.
- FURHT, B., KALRA, D., KITSON, F.L., RODRIGUEZ, and WALL, W.E.: "Design Issues for Interactive Televisions Systems," *IEEE Computer Magazine*, vol. 28, pp. 25-39, May 1995.
- GARCIA-HARO, J., and JAJSZCZYK, A.: "ATM Shared-Memory Switching Architectures," *IEEE Network Magazine*, vol. 8, pp. 18-26, July/Aug. 1994.
- GARG, V., and WILKES, J.E.: *Wireless and Personal Communication Systems*, Englewood Cliffs, NJ: Prentice Hall, 1996.
- GASMAN, L.: *Broadband Networking*, New York: Van Nostrand Reinhold, 1994.
- GIACOPELLI, J.N., HICKEY, J.J., MARCUS, W.S., SINOSKIE, W.D., and LITTLEWOOD, M.: "Sunshine: A High-Performance Self-Routing Broadband Packet Switch Architecture," *IEEE Journal on Selected Areas in Commun.*, vol. 9, pp. 1289-1298, Oct. 1991.
- GOODMAN, D.J.: "Trends in Cellular and Cordless Communications," *IEEE Commun. Magazine*, vol. 29, pp. 31-40, June 1991.
- GORALSKI, W.J.: *Introduction to ATM Networking*, New York: McGraw-Hill, 1995.
- GOSLING, J., JOY, B., and STEELE, G.: *The Java Language Specification*, Reading, MA: Addison-Wesley, 1996.
- GREEN, P.E., Jr.: *Fiber Optic Networks*, Englewood Cliffs, NJ: Prentice Hall, 1993.

- HAC, ANNA: "Wireless and Cellular Architecture and Services," *IEEE Commun. Magazine*, vol. 33, pp. 98-104, Nov. 1995.
- HAFNER, K., and MARKOFF, J.: *Cyberpunk*, New York: Simon and Schuster, 1991.
- HAMMING, R.W.: "Error Detecting and Error Correcting Codes," *Bell System Tech. J.*, vol. 29, pp. 147-160, April 1950.
- HANDEL, R., HUBER, M.N., and SCHRODER, S.: *ATM Concepts, Protocols, and Applications*, 2nd ed., Reading, MA: Addison-Wesley, 1994.
- HANDLEY, M., and CROWCROFT, J.: *The World Wide Web—Beneath the Surf*, London: UCL Press, 1994.
- HAWLEY, G.T.: "Historical Perspectives on the U.S. Telephone System," *IEEE Commun. Magazine*, vol. 29, pp. 24-28, March 1991.
- HEIN, M., and GRIFFITHS, D.: *SNMP*, London: Thompson, 1995.
- HELD, G.: *The Complete Modem Reference*, 2nd ed., New York: John Wiley, 1994.
- HELLMAN, M.E.: "A Cryptanalytic Time-Memory Tradeoff," *IEEE Trans. on Information Theory*, vol. IT-26, pp. 401-406, July 1980.
- HENDERSON, T.R.: "Design Principles and Performance Analysis of SSCOP: A New ATM Adaptation Layer Protocol," *Computer Commun. Review*, vol. 25, pp. 47-59, April 1995.
- HOARE, C.A.R.: "Monitors, An Operating System Structuring Concept," *Commun. of the ACM*, vol. 17, pp. 549-557, Oct. 1974; Erratum in *Commun. of the ACM*, vol. 18, p. 95, Feb. 1975.
- HODGE, W.W.: *Interactive Television*, New York: McGraw-Hill, 1995.
- HODGE, W.W., MARTIN, S., POWERS, J.T., Jr.: "Video on Demand: Architectures, Systems, and Applications," *Society of Motion Picture and Television Engineers Journal*, vol. 102, pp. 791-803, Sept. 1993.
- HOFFMAN, L.J. (ed.): *Building in Big Brother: The Cryptographic Policy Debate*, New York: Springer-Verlag, 1995.
- HOLFELDER, W.: "MBone VCR—Video Conference Recording on the MBone," *Proc. of ACM Multimedia '95*, ACM, pp. 237-238, 1995.
- HOLZMÄNN, G.J.: *Design and Validation of Computer Protocols*, Englewood Cliffs, NJ: Prentice Hall, 1991.
- HONG, D., and SUDA, T.: "Congestion Control and Prevention in ATM Networks," *IEEE Network Magazine*, vol. 5, pp. 10-16, July/Aug. 1991.
- HUANG, A., and KNAUER, S.: "Starlite: A Wideband Digital Switch," *Proc. Globecom '84*, pp. 121-125, 1984.
- HUGHES, J.P., and FRANTA, W.R.: "Geographic Extension of HIPPI Channels," *IEEE Network Magazine*, vol. 8, pp. 42-53, May/June 1994.

- HUI, J.: "A Broadband Packet Switch for Multi-rate Services," *Proc. Int'l. Conf. on Communications*, IEEE, pp. 782-788, 1987.
- HUIITEMA, C.: *IPv6: The New Internet Protocol*, Englewood Cliffs, NJ: Prentice Hall, 1996.
- HUIITEMA, C.: *Routing in the Internet*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- HUMBLET, P.A., RAMASWAMI, R., and SIVARAJAN, K.N.: "An Efficient Communication Protocol for High-Speed Packet-Switched Multichannel Networks," *Proc. SIGCOMM '92 Conf.*, ACM, pp. 2-13, 1992.
- IEEE: *Communications Magazine*, vol. 33, Jan. 1995.
- IEEE: *802.3: Carrier Sense Multiple Access with Collision Detection*, New York: IEEE, 1985a.
- IEEE: *802.4: Token-Passing Bus Access Method*, New York: IEEE, 1985b.
- IEEE: *802.5: Token Ring Access Method*, New York: IEEE, 1985c.
- IOANNIDIS, J., and MAQUIRE, G.Q., Jr.: "The Design and Implementation of a Mobile Internetworking Architecture," *Proc. Winter USENIX Conf.*, USENIX, pp. 491-502, Jan. 1993.
- IRMER, T.: "Shaping Future Telecommunications: The Challenge of Global Standardization," *IEEE Commun. Magazine*, vol. 32, pp. 20-28, Jan. 1994.
- IVANCIC, W.D., SHALKHAUSER, M.J., and QUINTANA, J.A.: "A Network Architecture for a Geostationary Communication Satellite," *IEEE Commun. Magazine*, vol. 32, pp. 72-84, July 1994.
- JABBARI, B., COLOMBO, G., NAKAJIMA, A., and KULKARNI, J.: "Network Issues for Wireless Communications," *IEEE Commun. Magazine*, vol. 33, pp. 88-98, Jan. 1995.
- JACOBSON, V.: "Congestion Avoidance and Control," *Proc. SIGCOMM '88 Conf.*, ACM, pp. 314-329, 1988.
- JAIN, R.: "Congestion Control and Traffic Management in ATM Networks: Recent Advances and a Survey," *Computer Networks and ISDN Systems*, vol. 27, Nov. 1995.
- JAIN, R.: *FDDI Handbook—High-Speed Networking Using Fiber and other Media*, Reading, MA: Addison-Wesley, 1994.
- JAIN, R.: *The Art of Computer Systems Performance Analysis*, New York: John Wiley, 1991.
- JAIN, R.: "Congestion Control in Computer Networks: Issues and Trends," *IEEE Network Magazine*, vol. 4, pp. 24-30, May/June 1990.
- JIA, F., and MUKHERJEE, B.: "The Receiver Collision Avoidance (RCA) Protocol for a Single-Hop WDM Lightwave Network," *Journal of Lightwave Technology*, vol. 11, pp. 1053-1065, May/June 1993.
- JOHNSON, D.B.: "Scalable Support for Transparent Mobile Host Internetworking," *Wireless Networks*, vol. 1, pp. 311-321, Oct. 1995.

- JOHNSON, H.W.: *Fast Ethernet—Dawn of a New Network*, Englewood Cliffs, NJ: Prentice Hall, 1996.
- KAHN, D.: "Cryptology Goes Public," *IEEE Commun. Magazine*, vol. 18, pp. 19-28, March 1980.
- KAHN, D.: *The Codebreakers*, New York: Macmillan, 1967.
- KALISKI, B.S., and ROBISHAW, M.J.B.: "Fast Block Cipher Proposal," *Proc. Cambridge Security Workshop*, Springer-Verlag, pp. 26-39, 1994.
- KAMOUN, F., and KLEINROCK, L.: "Stochastic Performance Evaluation of Hierarchical Routing for Large Networks," *Computer Networks*, vol. 3, pp. 337-353, Nov. 1979.
- KARN, P.: "MACA—A New Channel Access Protocol for Packet Radio," *ARRL/CRRL Amateur Radio Ninth Computer Networking Conf.*, pp. 134-140, 1990.
- KAROL, M.J., HLUCHYJ, M.G., and MORGAN, S.P.: "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Trans. on Commun.*, vol. 35, pp. 1347-1356, Dec. 1987.
- KARSHIMER, A.J., and THOMAS, J.N.: "Computer Networking on Cable TV Plants," *IEEE Commun. Magazine*, vol. 30, pp. 32-40, Nov. 1992.
- KATZ, D., and FORD, P.S.: "TUBA: Replacing IP with CLNP," *IEEE Network Magazine*, vol. 7, pp. 38-47, May/June 1993.
- KATZ, E.D., BUTLER, M., and MCGRATH, R.: "A Scalable HTTP Server: The NCSA Prototype," *Computer Networks and ISDN Systems*, vol. 27, pp. 155-164, Nov. 1994.
- KAUFMAN, C., PERLMAN, R., and SPECINER, M.: *Network Security*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- KAVAK, N.: "Data Communication in ATM Networks," *IEEE Network Magazine*, vol. 9, pp. 28-37, May/June 1995.
- KENT, C.A., and MOGUL, J.C.: "Fragmentation Considered Harmful," *Proc. SIGCOMM '87 Conf.*, ACM, pp. 390-401, 1987.
- KENT, S.T.: "Internet Privacy Enhanced Mail," *Commun. of the ACM*, vol. 36, pp. 48-60, Aug. 1993.
- KESSLER, G.C.: *ISDN*, 2nd ed., New York: McGraw-Hill, 1993.
- KESSLER, G.C., and TRAIN, D.: *Metropolitan Area Networks: Concepts, Standards, and Services*, New York: McGraw-Hill, 1992.
- KIM, J.B., SUDA, T., and YOSHIMURA, M.: "International Standardization of B-ISDN," *Computer Networks and ISDN Systems*, vol. 27, pp. 5-27, Oct. 1994.
- KLEINROCK, L., and TOBAGI, F.: "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels," *Proc. Nat. Computer Conf.*, pp. 187-201, 1975.
- KOHNO, R., MEIDAN, R., and MILSTEIN, L.B.: "Spread Spectrum Access Methods for Wireless Communication," *IEEE Commun. Magazine*, vol. 33, pp. 58-67, Jan. 1995.

- KUMAR, V.: *MBone: Interactive Multimedia on the Internet*, Indianapolis, IN: New Riders, 1996.
- KUNG, H.T., and MORRIS, R.: "Credit-Based Flow Control for ATM Networks," *IEEE Network Magazine*, vol. 9, pp. 40-48, March/April 1995.
- KWAN, T.T., MCGRATH, R.E., and REED, D.A.: "NCSA's WWW Server: Design and Performance," *IEEE Computer Magazine*, vol. 28, pp. 68-74, Nov. 1995.
- KWOK, T.: "A Vision for Residential Broadband Service: ATM to the Home," *IEEE Network Magazine*, vol. 9, pp. 14-28, Sept./Oct. 1995.
- KYAS, O.: *ATM Networks*, London: International Thomson Publishing, 1995.
- LAI, X.: *On the Design and Security of Block Ciphers*, Konstanz, Germany: Hartung-Gorre, 1992.
- LAI, X., and MASSEY, J.: "A Proposal for a New Block Encryption Standard," *Advances in Cryptology—Eurocrypt '90 Proceedings*, New York: Springer-Verlag, pp. 389-404, 1990.
- LAMPSON, B.W.: "A Note on the Confinement Problem," *Commun. of the ACM*, vol. 10, pp. 613-615, Oct. 1973.
- LANDAU, S.: "Zero-Knowledge and the Department of Defense," *Notices of the American Mathematical Society*, vol. 35, pp. 5-12, Jan. 1988.
- LANGSFORD, A.: "The Open System User's Programming Interfaces," *Computer Networks*, vol. 8, pp. 3-12, 1984.
- LA PORTA, T.F., VEERARAGHAVAN, M., AYANOGLU, E., KAROL, M., and GITLIN, R.D.: "B-ISDN: A Technological Discontinuity," *IEEE Commun. Magazine*, vol. 32, pp. 84-97, Oct. 1994.
- LATIF, A., ROWLANCE, F.J., and ADAMS, R.H.: "The IBM 8209 LAN Bridge," *IEEE Network Magazine*, vol. 6, pp. 28-37, May/June 1992.
- LAUDON, K.C.: "Ethical Concepts and Information Technology," *Commun. of the ACM*, vol. 38, pp. 33-39, Dec. 1995.
- LE BOUDEC, J.-Y.: "The Asynchronous Transfer Mode: A Tutorial," *Computer Networks and ISDN Systems*, vol. 24, pp. 279-309, May 1992.
- LEINER, B.M., COLE, R., POSTEL, J., and MILLS, D.: "The DARPA Internet Protocol Suite," *IEEE Commun. Magazine*, vol. 23, pp. 29-34, March 1985.
- LEVINE, D.A., and AKYILDIZ, I.A.: "PROTON: A Media Access Control Protocol for Optical Networks with Star Topology," *IEEE/ACM Trans. on Networking*, vol. 3, pp. 158-168, April 1995.
- LEVY, S.: "Crypto Rebels," *Wired*, pp. 54-61, May/June 1993.
- LIN, F., CHU, P., and LIU, M.: "Protocol Verification Using Reachability Analysis: The State Space Explosion Problem and Relief Strategies," *Proc. SIGCOMM '87 Conf.*, ACM, pp. 126-135, 1987.

- LIPPER, E.H., and RUMSEWICZ, M.P.: "Teletraffic Considerations for Widespread Deployment of PCS," *IEEE Network Magazine*, vol. 8, pp. 40-49, Sept./Oct. 1994.
- LITTLE, T.D.C., and VENKATESH, D.: "Prospects for Interactive Video-on-Demand," *IEEE Multimedia Magazine*, vol. 1, pp. 14-24, Fall 1994.
- LIU, C.L., and LAYLAND, J.W.: "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, vol. 20, pp. 46-61, Jan. 1973.
- LUOTONEN, A., and ALTIS, K.: "World Wide Web Proxies," *Computer Networks and ISDN Systems*, vol. 27, pp. 147-154, Nov. 1994.
- MACARIO, R.C.V.: *Cellular Radio—Principles and Design*, New York: McGraw-Hill, 1993.
- MACEDONIA, M.R., and BRUTZMAN, D.P.: "MBone Provides Audio and Video Across the Internet," *IEEE Computer Magazine*, vol. 27, pp. 30-36, April 1994.
- MASSEY, J.L.: "SAFER K-64: A Byte-Oriented Block Ciphering Algorithm," *Proc. Cambridge Security Workshop*, Springer-Verlag, pp. 1-17, 1994.
- MATSUI, M.: "Linear Cryptanalysis Method for DES Cipher," *Advances in Cryptology—Eurocrypt '93 Proceedings*, New York: Springer-Verlag, pp. 386-397, 1994.
- McBRYAN, O.: "GENVL and WWW: Tools for Taming the Web," *Proc. First Int'l. WWW Conference*, pp. 79-90, 1994.
- McDYSAN, D.E., and SPOHN, D.L.: *ATM—Theory and Application*, NY: McGraw-Hill, 1995.
- McKENNEY, P.E., and DOVE, K.F.: "Efficient Demultiplexing of Incoming TCP Packets," *Proc. SIGCOMM '92 Conf.*, ACM, pp. 269-279, 1992.
- MENEZES, A.J., and VANSTONE, S.A.: "Elliptic Curve Cryptosystems and Their Implementation," *Journal of Cryptology*, vol. 6, pp. 209-224, 1993.
- MERKLE, R.C.: "Fast Software Encryption Functions," *Advances in Cryptology—CRYPTO '90 Proceedings*, New York: Springer-Verlag, pp. 476-501, 1991.
- MERKLE, R.C., and HELLMAN, M.: "On the Security of Multiple Encryption," *Commun. of the ACM*, vol. 24, pp. 465-467, July 1981.
- MERKLE, R.C., and HELLMAN, M.: "Hiding and Signatures in Trapdoor Knapsacks," *IEEE Trans. on Information Theory*, vol. IT-24, pp. 525-530, Sept. 1978.
- METCALFE, R.M.: "On Mobile Computing," *Byte*, vol. 20, p. 110, Sept. 1995.
- METCALFE, R.M.: "Computer/Network Interface Design: Lessons from Arpanet and Ethernet," *IEEE Journal on Selected Areas in Commun.*, vol. 11, pp. 173-179, Feb. 1993.
- METCALFE, R.M., and BOGGS, D.R.: "Ethernet: Distributed Packet Switching for Local Computer Networks," *Commun. of the ACM*, vol. 19, pp. 395-404, July 1976.
- MIKI, T.: "The Potential of Photonic Networks," *IEEE Commun. Magazine*, vol. 32, pp. 23-27, Dec. 1994a.
- MIKI, T.: "Toward the Service-Rich Era," *IEEE Commun. Magazine*, vol. 32, pp. 34-39, Feb. 1994b.

- MINOLI, D.: *Video Dialtone Technology*, New York: McGraw-Hill, 1995.
- MINOLI, D., and VITELLA, M.: *ATM & Cell Relay for Corporate Environments*, New York: McGraw-Hill, 1994.
- MIRCHANDANI, S., and KHANNA, R. (eds): *FDDI Technologies and Applications*, New York: John Wiley, 1993.
- MISHRA, P.P. and KANAKIA, H.: "A Hop by Hop Rate-Based Congestion Control Scheme," *Proc. SIGCOMM '92 Conf.*, ACM, pp. 112-123, 1992.
- MOCHIDA, Y.: "Technologies for Local-Access Fibering," *IEEE Commun. Magazine*, vol. 32, pp. 64-73, Feb. 1994.
- MOGUL, J.C.: "The Case for Persistent-Connection HTTP," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 299-314, 1995.
- MOGUL, J.C.: "IP Network Performance," in *Internet System Handbook*, Lynch, D.C. and Rose, M.T. (eds.), Reading, MA: Addison-Wesley, pp. 575-675, 1993.
- MOK, A.K., and WARD, S.A.: "Distributed Broadcast Channel Access," *Computer Networks*, vol. 3, pp. 327-335, Nov. 1979.
- MORALES, J., PATKA, A., CHO, P., and KUI, J.: "Video Dial Tone Sessions," *IEEE Network Magazine*, vol. 9, pp. 42-47, Sept./Oct. 1995.
- MOY, J.: "Multicast Routing Extensions," *Commun. of the ACM*, vol. 37, pp. 61-66, Aug. 1994.
- MULLENDER, S.J. (ed.): *Distributed Systems*, 2nd ed., New York: ACM Press, 1993.
- MYLES, A., and SKELLERN, D.: "Comparison of Mobile Host Protocols for IP," *Computer Networks and ISDN Systems*, vol. 26, pp. 349-355, Dec. 1993.
- NAGLE, J.: "On Packet Switches with Infinite Storage," *IEEE Trans. on Commun.*, vol. COM-35, pp. 435-438, April 1987.
- NAGLE, J.: "Congestion Control in TCP/IP Internetworks," *Computer Commun. Rev.*, vol. 14, pp. 11-17, Oct. 1984.
- NEEDHAM, R.M., and SCHROEDER, M.D.: "Authentication Revisited," *Operating Systems Rev.*, vol. 21, p. 7, Jan. 1987.
- NEEDHAM, R.M., and SCHROEDER, M.D.: "Using Encryption for Authentication in Large Networks of Computers," *Commun. of the ACM*, vol. 21, pp. 993-999, Dec. 1978.
- NELSON, M.N., and LINTON, M.: "A Highly Available, Scalable ITV System," *Proc. Fifteenth Symp. on Operating Systems Print.*, ACM, pp. 54-67, 1995.
- NEMETH, E., SNYDER, G., SEEBASS, S., and HEIN, T.R.: *UNIX System Administration Handbook*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- NEMZOW, M.: *Implementing Wireless Networks*, New York: McGraw-Hill, 1995.
- NEUMAN, B.C., and TS'OI, T.: "Kerberos: An Authentication Service for Computer Networks," *IEEE Commun. Magazine*, vol. 32, pp. 33-38, Sept. 1994.

- NEWMAN, P.: "Traffic Management for ATM Local Area Networks," *IEEE Commun. Magazine*, vol. 32, pp. 44-50, Aug. 1994.
- NEWMAN, P.: "ATM Local Area Networks," *IEEE Commun. Magazine*, vol. 32, pp. 86-98, March 1994.
- NIST: "Secure Hash Algorithm," U.S. Government Federal Information Processing Standard 180, 1993.
- OMIDYAR, C.G., and ALDRIDGE, A.: "Introduction to SDH/SONET," *IEEE Commun. Magazine*, vol. 31, pp. 30-33, Sept. 1993.
- OTWAY, D., and REES, O.: "Efficient and Timely Mutual Authentication," *Operating Systems Rev.*, pp. 8-10, Jan. 1987.
- PADGETT, J.E., GUNTHER, C.G., and HATTORI, T.: "Overview of Wireless Personal Communications," *IEEE Commun. Magazine*, vol. 33, pp. 28-41, Jan. 1995.
- PAFF, A.: "Hybrid Fiber/Coax in the Public Telecommunications Infrastructure," *IEEE Commun. Magazine*, vol. 33, pp. 40-45, April 1995.
- PAHLAVAN, K., PROBERT, T.H., and CHASE, M.E.: "Trends in Local Wireless Networks," *IEEE Commun. Magazine*, vol. 33, pp. 88-95, March 1995.
- PALAIS, J.C.: *Fiber Optic Commun.*, 3rd ed., Englewood Cliffs, NJ: Prentice Hall, 1992.
- PALMER, L.C., and WHITE, L.W.: "Demand Assignment in the ACTS LBR System," *IEEE Trans. on Commun.*, vol. 38, pp. 684-692, May 1990.
- PAN, D.: "A Tutorial on MPEG/Audio Compression," *IEEE Multimedia Magazine*, vol. 2, pp. 60-74, Summer 1995.
- PANCHAL, P., and EL ZARKI, M.: "MPEG Coding for Variable Bit Rate Video Transmission," *IEEE Commun. Magazine*, vol. 32, pp. 54-66, May 1994.
- PANDYA, R.: "Emerging Mobile and Personal Communication Systems," *IEEE Commun. Magazine*, vol. 33, pp. 44-52, June 1995.
- PARTRIDGE, C.: *Gigabit Networking*, Reading, MA: Addison-Wesley, 1994.
- PARTRIDGE, C.: "A Proposed Flow Specification," Internet RFC 1363, Sept. 1992.
- PARTRIDGE, C., HUGHES, J., and STONE, J.: "Performance of Checksums and CRCs over Real Data," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 68-76, 1995.
- PARULKAR, G., SCHMIDT, D.C., and TURNER, J.S.: "AITPM: A Strategy for Integrating IP with ATM," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 49-58, 1995.
- PAXSON, V.: "Growth Trends in Wide-Area TCP Connections," *IEEE Network Magazine*, vol. 8, pp. 8-17, July/Aug. 1994.
- PAXSON, V., and FLOYD, S.: "Wide-Area Traffic: The Failure of Poisson Modeling," *Proc. SIGCOMM '94 Conf.*, ACM, pp. 257-268, 1995.
- PERKINS, C.: "Providing Continuous Network Access to Mobile Hosts Using TCP/IP," *Computer Networks and ISDN Systems*, vol. 26, pp. 357-370, Nov. 1993.

- PERLMAN, R.: *Interconnections: Bridges and Routers*, Reading, MA: Addison-Wesley, 1992.
- PERLMAN, R.: *Network Layer Protocols with Byzantine Robustness*, Ph.D. thesis, M.I.T., 1988.
- PERRY, T.S., and ADAM, J.A.: "E-Mail: Pervasive and Persuasive," *IEEE Spectrum*, vol. 29, pp. 22-28, Oct. 1992.
- PETERSON, W.W., and BROWN, D.T.: "Cyclic Codes for Error Detection," *Proc. IRE*, vol. 49, pp. 228-235, Jan. 1961.
- PICKHOLTZ, R.L., SCHILLING, D.L., and MILSTEIN, L.B.: "Theory of Spread Spectrum Communication—A Tutorial," *IEEE Trans. on Commun.*, vol. COM-30, pp. 855-884, May 1982.
- PIERCE, J.: "How Far Can Data Loops Go?" *IEEE Trans. on Commun.*, vol. COM-20, pp. 527-530, June 1972.
- PINKERTON, B.: "Finding What People Want: Experiences with the WebCrawler," *Proc. First Int'l. WorldWide Web Conference*, 1994.
- PISCITELLO, D.M., and CHAPIN, A.L.: *Open Systems Networking: TCP/IP and OSI*, Reading, MA: Addison-Wesley, 1993.
- PITT, D.A.: "Bridging—The Double Standard," *IEEE Network Magazine*, vol. 2, pp. 94-95, Jan. 1988.
- QUICK, R. F., Jr., and BALACHANDRAN, K.: "An Overview of the Cellular Digital Packet Data (CDPD) System," *Fourth Int'l. Symp. on Personal, Indoor, and Mobile Radio Commun.*, pp. 338-343, 1993.
- QUISQUATER, J.-J., and GIRAUT, M.: "Chinese Lotto as an Exhaustive Code-Breaking Machine," *IEEE Computer Magazine*, vol. 24, pp. 14-22, Nov. 1991.
- RABIN, M.O.: "Digital Signatures and Public-Key Functions as Intractable as Factorization," Technical Report LCS-TR-212, M.I.T., Jan 1979.
- RAHNEMA, M.: "Overview of the GSM System and Protocol Architecture," *IEEE Commun. Magazine*, vol. 31, pp. 92-100, April 1993.
- RAJAGOPALAN, B.: "Reliability and Scaling Issues in Multicast Communication," *Proc. SIGCOMM '92 Conf.*, ACM, pp. 188-198, 1992.
- RANSOM, M.N.: "The VISTAnet Gigabit Network Testbed," *Journal of High Speed Networks*, vol. 1, pp. 49-60, 1992.
- RAO, S.K., and HATAMIAN, M.: "The ATM Physical Layer," *Computer Commun. Rev.*, vol. 25, pp. 73-81, April 1995.
- RIVEST, R.L.: "The MD5 Message-Digest Algorithm," RFC 1320, April 1992.
- RIVEST, R.L., and SHAMIR, A.: "How to Expose an Eavesdropper," *Commun. of the ACM*, vol. 27, pp. 393-395, April 1984.

- RIVEST, R.L., SHAMIR, A., and ADLEMAN, L.: "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Commun. of the ACM*, vol. 21, pp. 120-126, Feb. 1978.
- ROBERTS, L.: "Dynamic Allocation of Satellite Capacity through Packet Reservation," *Proc. NCC, AFIPS*, pp. 711-716, 1973.
- ROBERTS, L.: "Extensions of Packet Communication Technology to a Hand Held Personal Terminal," *Proc. Spring Joint Computer Conference, AFIPS*, pp. 295-298, 1972.
- ROMANOW, A., and FLOYD, S.: "Dynamics of TCP Traffic over ATM Networks," *Proc. SIGCOMM '84 Conf.*, ACM, pp. 79-88, 1994.
- ROSE, M.T.: *The Simple Book*, Englewood Cliffs, NJ: Prentice Hall, 1994.
- ROSE, M.T.: *The Internet Message*, Englewood Cliffs, NJ: Prentice Hall, 1993.
- ROSE, M.T., and McCLOGHRIE, K.: *How to Manage Your Network Using SNMP*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- ROSS, F.E., and HAMSTRA, J.R.: "Forging FDDI," *IEEE Journal on Selected Areas in Commun.*, vol. 11, pp. 181-190, Feb. 1993.
- SADIQU, M.N.O., and ARVIND, A.S.: "Annotated Bibliography on Distributed Queue Dual Bus (DQDB)," *Computer Commun. Rev.*, vol. 24, pp. 21-36, Jan. 1994.
- SALTZER, J.H., POGRAN, K.T., and CLARK, D.D.: "Why a Ring?" *Computer Networks*, vol. 7, pp. 223-230, Aug. 1983.
- SALTZER, J.H., REED, D.P., and CLARK, D.D.: "End-to-End Arguments in System Design," *ACM Trans. on Computer Systems*, vol. 2, pp. 277-288, Nov. 1984.
- SANDERSON, D.W., and DOUGHERTY, D.: *Smileys*, Sebastopol, CA: O'Reilly, 1993.
- SANTIFALLER, M.: "TCP/IP and ONC/NFS," Reading, MA: Addison-Wesley, 1994.
- SCHNEIER, B.: *Applied Cryptography*, 2nd ed., New York: John Wiley, 1996.
- SCHNEIER, B.: *E-Mail Security*, New York: John Wiley, 1995.
- SCHNEIER, B.: "Description of a New Variable-Length Key, 64-Bit Block Cipher [Blowfish]," *Proc. of the Cambridge Security Workshop*, Springer-Verlag, pp. 191-204, 1994.
- SCHNORR, C.P.: "Efficient Signature Generation for Smart Cards," *Journal of Cryptology*, vol. 4, pp. 161-174, 1991.
- SCHOLTZ, R.A.: "The Origins of Spread-Spectrum Communications," *IEEE Trans. on Commun.*, vol. COM-30, pp. 822-854, May 1982.
- SCOTT, R.: "Wide Open Encryption Design Offers Flexible Implementations," *Cryptography*, vol. 9, pp. 75-90, Jan. 1985.
- SELFRIFFE, O.G., and SCHWARTZ, R.T.: "Telephone Technology and Privacy," *Technology Rev.*, vol. 82, pp. 56-65, May 1980.

- SEYBOLD, A.M.: *Using Wireless Communications in Business*, New York: Van Nostrand Reinhold, 1994.
- SHACHAM, N., and MCKENNEY, P.: "Packet Recovery in High-Speed Networks Using Coding and Buffer Management," *Proc. INFOCOM '90*, IEEE, pp. 124-130, 1990.
- SHAH, A., and RAMAKRISHNAN, G.: *FDDI—A High Speed Network*, Englewood Cliffs, NJ: Prentice Hall, 1994.
- SHANNON, C.: "A Mathematical Theory of Communication," *Bell System Tech. J.*, vol. 27, pp. 379-423, July 1948; and pp. 623-656, Oct. 1948.
- SHEN, B., and SETHI, I.K.: "Inner-Block Operations on Compressed Images," *Proc. of ACM Multimedia '95*, ACM, pp. 489-498, 1995.
- SHIMIZU, A., and MIYAGUCHI, S.: "Fast Data Encipherment Algorithm FEAL," *Advances in Cryptology—Eurocrypt '87 Proceedings*, NY: Springer-Verlag, pp. 267-278, 1988.
- SHREEDHAR, M., and VARGHESE, G.: "Efficient Fair Queueing Using Deficit Round Robin," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 231-243, 1995.
- SINGLETON, A.: "Wired on the Web," *Byte*, vol. 21, pp. 77-80, Jan. 1996.
- SIPIOR, J.C., and WARD, B.T.: "The Ethical and Legal Quandary of Email Privacy," *Commun. of the ACM*, vol. 38, pp. 48-54, Dec. 1995.
- SIU, K.-Y., and JAIN, R.: "A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management," *Computer Commun. Rev.*, vol. 25, pp. 6-20, April 1995.
- SMITH, P.: *Frame Relay*, Reading, MA: Addison-Wesley, 1993.
- SOHA, M., and PERLMAN, R.: "Comparison of Two LAN Bridge Approaches," *IEEE Network Magazine*, vol. 2, pp. 37-43, Jan./Feb. 1988.
- SPAFFORD, E.H.: "The Internet Worm: Crisis and Aftermath," *Commun. of the ACM*, vol. 32, pp. 678-687, June 1989.
- SPRAGINS, J.D., with HAMMOND, J.L., and PAWLICKOWSKI, K.: *Telecommunications Protocols and Design*, Reading, MA: Addison-Wesley, 1991.
- STALLINGS, W.: *ISDN and Broadband ISDN with Frame Relay and ATM*, Englewood Cliffs, NJ: Prentice Hall, 1995a.
- STALLINGS, W.: *Network and Internetwork Security*, Englewood Cliffs, NJ: Prentice Hall, 1995b.
- STALLINGS, W.: *Protect Your Privacy: The PGP User's Guide*, Englewood Cliffs, NJ: Prentice Hall, 1995c.
- STALLINGS, W.: *Data and Computer Communications*, 4th ed., New York: Macmillan, 1994.
- STALLINGS, W.: *SNMP, SNMPv2, and CMIP*, Reading, MA: Addison-Wesley, 1993a.
- STALLINGS, W.: *Local and Metropolitan Area Networks*, 4th ed., New York: Macmillan, 1993b.

- STEELE, R., WHITEHEAD, J., and WONG, W.C.: "System Aspects of Cellular Radio," *IEEE Commun. Magazine*, vol. 33, pp. 80-86, Jan. 1995a.
- STEELE, R., WILLIAMS, J., CHANDLER, D., DEHGHAN, S., and COLLARD, A.: "Teletraffic Performance of GSM900/DCS1800 in Street Microcells," *IEEE Commun. Magazine*, vol. 33, pp. 102-108, March 1995b.
- STEINER, J.G., NEUMAN, B.C., and SCHILLER, J.I.: "Kerberos: An Authentication Service for Open Network Systems," *Proc. Winter USENIX Conf.*, USENIX, pp. 191-201, 1988.
- STEINMETZ, R., and NAHRSTEDT, K.: *Multimedia: Computing, Communications and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- STEPHENS, W.E., and BANWELL, T.C.: "155.52 Mb/s Data Transmission on Category 5 Cable Plant," *IEEE Commun. Magazine*, vol. 33, pp. 62-69, April 1995.
- STERBENZ, J.P.G., SCHULZRINNE, H.G., and TOUCH, J.D.: "Report and Discussion of the IEEE ComSoc TCGN Gigabit Networking Workshop 1995," *IEEE Network Magazine*, vol. 9, pp. 9-29, July/Aug. 1995.
- STEVENS, W.R.: *TCP/IP Illustrated*, Vol. 1, Reading, MA: Addison-Wesley, 1994.
- STILLER, B.: "A Survey of UNI Signaling Systems and Protocols," *Computer Commun. Rev.*, vol. 25, pp. 21-33, April 1995.
- STINSON, D.R.: *Cryptography Theory and Practice*, Boca Raton, FL: CRC Press, 1995.
- SUNSHINE, C.A., and DALAL, Y.K.: "Connection Management in Transport Protocols," *Computer Networks*, vol. 2, pp. 454-473, 1978.
- SUZUKI, T.: "ATM Adaptation Layer Protocol," *IEEE Commun. Magazine*, vol. 32, pp. 80-83, April 1994.
- TANENBAUM, A.S.: *Distributed Operating Systems*, Englewood Cliffs, NJ: Prentice Hall, 1995.
- TANENBAUM, A.S.: *Modern Operating Systems*, Englewood Cliffs, NJ: Prentice Hall, 1992.
- TERAOKA, F., YOKTE, Y., and TOKORO, M.: "Host Migration Transparency in IP Networks," *Computer Commun. Rev.*, vol. 23, pp. 45-65, Jan. 1993.
- THYAGARAJAN, A.S., and DEERING, S.E.: "Hierarchical Distance-Vector Multicast Routing for the MBone," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 60-66, 1995.
- TOKORO, M., and TAMARU, K.: "Acknowledging Ethernet," *Componet*, IEEE, pp. 320-325, Fall 1977.
- TOLMIE, D.E.: "Gigabit LAN Issues—HIPPI, Fibre Channel, and ATM," in *Proc. High-Performance Computing and Networking*, Hertzberger, B., and Scrazi, G. (Eds.), Berlin: Springer Verlag, pp. 45-53, 1995.
- TOLMIE, D.E.: "Gigabit Networking," *IEEE LTS*, vol. 3, pp. 28-36, May 1992.
- TOLMIE, D.E., and RENWICK, J.: "HIPPI: Simplicity Yields Success," *IEEE Network Magazine*, vol. 7, pp. 28-32, Jan./Feb. 1993.

- TOMLINSON, R.S.: "Selecting Sequence Numbers," *Proc. SIGCOMM/SIGOPS Interprocess Commun. Workshop*, ACM, pp. 11-23, 1975.
- TOUCH, J.D.: "Performance Analysis of MDS," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 77-86, 1995.
- TRUONG, H.L., ELLINGTON, W.W. Jr., LE BOUDEC, J.-Y., MEIER, A.X., and PACE, J.W.: "LAN Emulation on an ATM Network," *IEEE Commun. Magazine*, vol. 33, pp. 70-85, May 1995.
- TUCHMAN, W.: "Hellman Presents No Shortcut Solutions to DES," *IEEE Spectrum*, vol. 16, pp. 40-41, July 1979.
- TURNER, J.S.: "New Directions in Communications (or Which Way to the Information Age)," *IEEE Commun. Magazine*, vol. 24, pp. 8-15, Oct. 1986.
- VAN DER LINDEN, P.: *Just Java*, Englewood Cliffs, NJ: Prentice Hall, 1996.
- VAN OORSCHOT, P.C., and WIENER, M.J.: "A Known-Plaintext Attack on Two-Key Triple Encryption," *Advances in Cryptology—CRYPTO '88 Proceedings*, New York: Springer-Verlag, pp. 119-131, 1988.
- VAN RENESSE, R., VAN STAVEREN, H., and TANENBAUM, A.S.: "Performance of the World's Fastest Distributed Operating System," *Operating Systems Rev.*, vol. 22, pp. 25-34, Oct. 1988.
- VARGHESE, G., and LAUCK, T.: "Hashed and Hierarchical Timing Wheels: Data Structures for the Efficient Implementation of a Timer Facility," *Proc. Eleventh Symp. on Operating Systems Prin.*, ACM, pp. 25-38, 1987.
- VENKATRAMANI, C., and CHIUEH, T.: "Design, Implementation, and Evaluation of a Software-Based Real-Time Ethernet Protocol," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 27-37, 1995.
- VETTER, R.J., SPELL, C., and WARD, C.: "Mosaic and the World-Wide Web," *IEEE Computer Magazine*, vol. 27, pp. 49-57, Oct. 1994.
- VILLAMIZAN, C., and SONG, C.: "High Performance TCP in ANSNET," *Computer Commun. Rev.*, vol. 25, pp. 45-60, Oct. 1995.
- VITERBI, A.J.: *CDMA Principles of Spread Spectrum Communication*, Reading, MA: Addison-Wesley, 1995.
- WADA, H., YOZAWA, T., OHNISHI, T., and TANAKA, Y.: "Mobile Computing Environment Based on Internet Packet Forwarding," *Proc. Winter USENIX Conf.*, USENIX, pp. 503-517, Jan. 1993.
- WALRAND, J.: *Communication Networks: A First Course*, Homewood, IL: Irwin, 1991.
- WATSON, R.W.: "Timer-Based Mechanisms in Reliable Transport Protocol Connection Management," *Computer Networks*, vol. 5, pp. 47-56, Feb. 1981.
- WAYNER, P.: "Picking the Crypto Lock," *Byte*, pp. 77,80, Oct. 1995.

- WEISBAND, S.P., and REINIG, B.A.: "Managing User Perceptions of Email Privacy," *Commun. of the ACM*, vol. 38, pp. 40-47, Dec. 1995.
- WIENER, M.J.: "Efficient DES Key Search," Technical Report TR-244, School of Computer Science, Carleton Univ., Ottawa, 1994.
- WILLIAMS, K.A., DAM, T.Q., and DU, D.H.-C.: "A Media Access Protocol for Time and Wavelength-Division Multiplexed Passive Star Networks," *IEEE Journal on Selected Areas in Commun.*, vol. 11, pp. 560-567, May 1993.
- WILLINGER, W., TAQQU, M.S., SHERMAN, R., and WILSON, D.V.: "Self-Similarity through High Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level," *Proc. SIGCOMM '95 Conf.*, ACM, pp. 100-113, 1995.
- WOLTER, M.S.: "Fiber Distributed Data Interface—A Tutorial," *Connexions*, pp. 16-26, Oct. 1990.
- YANG, C.-Q., and REDDY, A.V.S.: "A Taxonomy for Congestion Control Algorithms in Packet Switching Networks," *IEEE Network Magazine*, vol. 9, pp. 34-45, July/Aug. 1995.
- YEH, Y.-S., BLUCHYJ, M.G., and ACAMPORA, A.S.: "The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching," *IEEE Journal on Selected Areas in Commun.*, vol. 5, pp. 1274-1283, Oct. 1987.
- YOUSSEF, A.M., KALMAN, E., BENZONI, L.: "Technico-Economic Methods of Radio Spectrum Assignment," *IEEE Commun. Magazine*, vol. 33, pp. 88-94, June 1995.
- YUVAL, G.: "How to Swindle Rabin," *Cryptologia*, vol. 3, pp. 187-190, July 1979.
- ZHANG, L.: "Comparison of Two Bridge Routing Approaches," *IEEE Network Magazine*, vol. 2, pp. 44-48, Jan./Feb. 1988.
- ZHANG, L.: "RSVP A New Resource ReSerVation Protocol," *IEEE Network Magazine*, vol. 7, pp. 8-18, Sept./Oct. 1993.
- ZIMMERMANN, P.R.: *The Official PGP User's Guide*, Cambridge, MA: M.I.T. Press, 1995a.
- ZIMMERMANN, P.R.: *PGP: Source Code and Internals*, Cambridge, MA: M.I.T. Press, 1995b.
- ZIPF, G.K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Cambridge, MA: Addison-Wesley, 1949.
- ZIV, J., and LEMPEL, Z.: "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. on Information Theory*, vol. IT-23, pp. 337-343, May 1977.

ÍNDICE

A

- AAL (véase capa de adaptación de ATM)
 AAL 1, 547-549, 753
 AAL 2, 549-550, 753
 AAL 3/4, 550-552, 753
 AAL 5, 552-554, 753
 ABR (véase servicio de tasa de bits disponible)
 abrir primero la trayectoria más corta, 424-429
 acceso múltiple con detección digital, 270-271
 acceso múltiple con prevención de colisiones, 264-265
 acceso múltiple por división de código, 271-275, 333
 acceso múltiple por división en longitud de onda, 260-262
 ACR (véase tasa de células real)
 ACTS (véase satélite de tecnología de comunicaciones avanzada)
 acuerdo de tres vías, 496-498
 acuse negativo, 215
 ADC (véase convertidor analógico a digital)
 ADCCP (véase procedimiento avanzado de control de comunicación de datos)
 adenda, 42, 322
 administración de fichas, 32
 administración postal, telegráfica y telefónica, 67
 ADSL (véase línea asimétrica digital de suscriptor)
 agente de base, 368
 agente de transferencia de mensajes, 645
 agente foráneo, 368
 agente SNMP, 631
 agente usuario, 645, 646-648
 alambre de cobre, comparación con fibra, 92-94
 algoritmo de aprendizaje en reversa, 311-312
 algoritmo de arranque lento, 538-539
 algoritmo de arranque lento de Jacobson, 538-539
 algoritmo de control de congestionamientos, 374-395
 ATM, 467-471
 basado en tasa, 469-471
 cubeta con fichas, 381-384
 cubeta con fugas, 380-381
 en redes de circuitos virtuales, 386-387
 manejo de colas justo ponderado, 388-389
 multitransmisión, 393-395
 paquetes de estrangulamiento, 387-391
 principios generales, 376-378
 TCP, 536-539
 algoritmo de cubeta con fichas, 381-384
 algoritmo de cubeta con fugas, 380-381
 algoritmo de enrutamiento, 345-374
 adaptable, 347
 basado en flujo, 353-355
 difusión, 370-372
 estado de enlace, 359-365
 host móvil, 367-370
 interred, 403-406
 inundación, 351
 jerárquico, 365-367
 multitransmisión, 372-374
 no adaptable, 347
 reenvío por trayectoria invertida, 371-372
 trayectoria más corta, 349
 vector de distancia, 355-359
 algoritmo de Karn, 541
 algoritmo de Nagle, 534-535
 algoritmo de planificación virtual, 466
 algoritmo genérico de tasa de células, 463-466
 algoritmo internacional de cifrado de datos, 596-597
 algoritmo RSA, 598-600, 665-666
 algoritmo seguro de parcialización, 618
 alias, correo electrónico, 647-648
 alimentación de noticias, 675

ALOHA, 246-250
 por satélite, 329
 puro, 247-249
 ranurado, 249-250
American National Standards Institute, 70
AMPS (*véase sistema de telefonía móvil avanzada*)
análisis de asequibilidad, 20
análisis de Fourier, 78
anillo de claves privadas, pgp, 666
anillo de claves públicas, 667
anillo en forma de estrella, 295
ANSI (*véase American National Standards Institute*)
ANSNET, 51
anuncios, IP móvil, 433
aplicación ayudante, 684
aplicaciones Internet
 correo electrónico, 643-670
 MBone, 756-760
 noticias de red, 669-680
 World Wide Web, 681-723
apocalipsis de los dos elefantes, 40-41
applet, 707-709
araña, 720
árbol basado en núcleo, 374
árbol de extensión, 371
archivo de eliminación, 672
área local de acceso y transporte, 106
área OSPF, 425
armadura ASCII, 654
ARP (*véase protocolo de resolución de direcciones*)
ARP apoderado, 423
ARP gratuito, 433
ARPANET, 35, 47-50, 71, 355, 569, 622
ARQ (*véase solicitud de repetición automática*)
arquitectura de red, 18
arquitectura de red de sistemas, 41
arrreglo redundante de discos baratos, 748
artículo de noticias, ejemplo, 676
asignación de canales en las LAN, 244-246
ASN.1 (*véase Notación de Sintaxis Abstracta-1*)
aspectos sociales, 6-7
 relacionados a la criptografía, 620-622
ataque de brigada de cubetas, 606
ataque de cumpleaños, 618-620
ataque de encuentro a la mitad, 594
ataque de sólo texto cifrado, 582
ataque de texto normal conocido, 582

B

backbone de multitransmisión, 756-760
backbone de OSPF, 425
banda de frecuencia, 95
banda de muy alta frecuencia, 95, 97-98
banda de muy baja frecuencia, 95, 97
banda industrial/científica/médica, 99
banda ISM (*véase banda industrial/científica/médica*)
banda VHF (*véase banda de muy alta frecuencia*)
banda VLF (*véase banda de muy baja frecuencia*)
base64, 654
base de información de administración, 632, 641-642
band, 79
Bell Operating Company, 106
Bell System, 103
BGP (*véase protocolo de pasarela exterior*)
bit de paridad, 185
BITNET, 53
Blanca, 56
bloqueo de inicio de línea, 149
BOC (*véase Bell Operating Company*)
BOOTP, 424
BUS (*véase servidor de difusión/desconocido*)
bus doble de cofas distribuidas, 11, 301-303
buñón, 645

C

cabeecera, 19
 correo electrónico, 646
 marco, 192
cabecera de noticias, 676-677

cable coaxial, 84-86
 de banda ancha, 85-86
 de banda base, 84-85
cable de banda ancha, 85-86
cable de banda base, 85, 86, 276, 277, 280
cable de fibra, 90-91
cable dividido a la mitad, 86
cable subdividido, 85
caja de control, 754-756
caja P, 587-588
caja S, 587-588
calidad de servicio, 23, 460-463, 481-483
 ATM, 460-463
campo de vídeo, 728
canal, 11
canal de acceso aleatorio, 243
canal de fibra, 326-327
canal disimulado, 719-720
canal multiacceso, 243
canal virtual, ATM, 450
cancelador de eco, 113
capa, 17
capa AAL, 64, 545-555
capa ATM, 449-473, 63
capa de adaptación sencilla y eficiente, 552-554
capa de adaptación de ATM, 545-555
capa de aplicación, 33-34, 37, 577-766
 administración de red, 630-643
 correo electrónico, 643-669
 multimedia, 723-760
 noticias de red, 669-680
 seguridad de la red, 577-622
 sistema de nombres de dominio, 622-630
 World Wide Web, 681-723
capa de enlace de datos, 175-242
 ATM, 235-239
 control de errores, 182-183
 control de flujo, 183
 cuestiones de diseño, 176-183
 enmarcado, 179-182
 LLC, 275, 302-304
 OSI, 30
 protocolo HDLC, 225-228
 protocolos de ejemplo, 225-239
 protocolos de ventana corrediza, 202-219
 protocolos elementales, 190-202
 relleno de bits, 181

capa de enlace de datos (*cont.*)
 relleno de caracteres, 180-181
 servicios proporcionados, 176-179
capa de interred, 35-36
capa de presentación, 33
capa de red, 31, 35-36, 339-478
 algoritmos de enruteamiento, 345-374
 control de congestión, 374-395
 Internet, 412-449
 interredes, 396-412
 organización interna, 342-345
 OSI, 31
 problemas de diseño, 339-345
 redes ATM, 449-473
 servicios proporcionados, 340-342
capa de sesión, OSI, 32-33
capa de transporte, 31-32, 36-37, 479-576
 AAL de ATM, 545-555
 desempeño de red, 555-572
 ejemplo, 510-521
 elementos de protocolo, 488-510
 Internet, 521-545
 OSI, 31
 servicio proporcionado, 479-487
capa física, 29-30, 77-174
 medios de transmisión, 82-94
 OSI, 29-30
 radio celular, 155-163
 satélites de comunicación, 163-170
 sistema telefónico, 102-163
 transmisión inalámbrica, 94-102
capa host a red, 38
carita, 674
Carnegie-Mellon University, 7
CASA, 56
casi video a solicitud, 744
CBR (*véase servicio de tasa de bits constante*)
CCITT, 68, 119, 121, 122, 124, 142, 644
CD de audio, 724-725
CDMA (*véase acceso múltiple por división de código*)
CDPD (*véase paquete celular digital de datos*)
CDV (*véase variación de retardo de células*)
celda de radio celular, 158
Célula
 ATM, 62
 HTML, 701
célula de administración de recursos, 470

ARCMA
célula de operación y mantenimiento, 236
célula OAM (*véase* célula de operación y mantenimiento)
célula RM (*véase* célula de administración de recursos)
censura
de CompuServe, 676
del CMU, 7
central de conmutación de datos, 12
central de ramal privado, 142
centro de conmutación móvil, 159
centro de distribución de claves, 607-610
centro de información de redes, 417
CER (*véase* tasa de error de células)
CGI (*véase* interfaz de pasarela común)
Chapultín Colorado, 745
chip, 272
CIDR (*véase* enrutamiento interdominios sin clases)
cifrado
de bloque, 585, 590, 595-596
de César, 582-583
por sustitución, 582-583
por transposición, 583-585
cifrado (*véase* criptografía)
cifrado de enlace, 579
cifrado por sustitución monoalfabética, 582-583
circuito, 11
circuito virtual, 342-345
comparación con datagrama, 344-345
circuito virtual conmutado, 60, 145-146
circuito virtual permanente, 60, 145-146
circuitos virtuales concatenados, 401-402
clave criptográfica, 580
clave de sesión, 602
CLR (*véase* tasa de pérdida de células)
CLUT (*véase* tabla de búsqueda de colores)
CMR (*véase* tasa de mala inserción de células)
codec, 121
codificación, 730
entrópica, 731-732
por fuente, 732-734
codificación de entrelazo, 112
codificación diferencial, 732
codificación entrecomillada imprimible, 654
codificación estadística, 731-732
codificación Manchester, 279-280
codificación Manchester diferencial, 279-280
codificación por longitud de serie, 731

codificación por transformación, 732-733
codificación predictiva, 124
código de corrección de errores, 184
código de detección de errores, 184, 186-190
código de redundancia efílica, 187
código polinómico, 187
Comisión Federal de Comunicaciones, 96, 100, 167
comparación de las LAN, 299-301
compendio de mensaje, 617-618
compresión de datos, 730-744
CLUT, 732
codificación de origen, 732-734
codificación diferencial, 732
codificación entrópica, 731-732
codificación estadística, 731-732
codificación por longitud de serie, 731
codificación por transformación, 732
con pérdidas, 732-734
cuantización vectorial, 733
sin pérdidas, 731-732
transformación por coseno discreto, 733
CompuServe, 676
computadora *big endian*, 413
computadora *little endian*, 413
comunicación dúplex integral, 21, 113
comunicación semidúplex, 21, 113
comunicación simplex, 21
concentrador de satélite, 165
Conferencia Mundial Administrativa de Radio, 95
confirmación, 25-27
conformación de tráfico, 379-380, 463-468
comunicación
de circuitos, 130-134
de mensajes, 131-133
de paquetes, 133-134
por almacenamiento y reenvío, 133
telefónica, 130-139
comutador
barra cruzadas, 135-138
por división de espacio, 136-138
por división de tiempo, 138-139
comutador ATM, 147-155
comutador Batcher-baniano, 151-155
comutador de barras cruzadas, 135-138
comutador de punto de cruce, 135-138
comutador por división de espacio, 136-138
comutador por división en el tiempo, 138-139

comutador por knockout, 150-151
Consejo de Arquitectura de Internet, 71
consulta recursiva, 630
conteo binario descendente, 255-256
control de admisión, 386, 468
control de congestionamiento basado en tasa, 469-471
control de enlace de alto nivel, 225-228
control de enlace lógico, 275, 302-304
control de errores de cabecera, ATM, 235-238
control de errores, 182-190
control de fluctuación, 392-393
control de flujo, 183, 502-506
control síncrono de enlace de datos, 226-227
convertidor analógico a digital, 725
corrección de errores, 184-190
correo con confidencialidad mejorada, 667-669
comparación con pgp, 669-670
correo electrónico, 5, 53, 643-670
agente de transferencia de mensaje, 645
agente de usuario, 645, 646-648
arquitectura y servicios, 645-646
cabecera, 646
comandos de usuario, 648-650
confidencialidad, 663-669
cuerpo, 646
entrega final, 662-663
envío, 646-648
filtro, 662
formato de mensaje, 650-658
formato MIME, 653-657
formato RFC 822, 651-653
funciones, 645
lectura, 648-650
pasarela, 659-661
primeros sistemas, 644
sobre, 646
transferencia de mensaje, 657-663
corriente de programa, MPEG, 743
corriente de transporte, 743
corriente elemental empacada, 743
criptoanálisis, 581
criptoanálisis diferencial, 595
criptoanálisis lineal, 595
criptografía, 577-622
de clave pública, 597-601
de clave secreta, 587-597
tradicional, 580-585

criptografía de clave simétrica, 598
criptología, 581
creminalidad, 728-729

CSMA (*véase* protocolo de acceso múltiple con detección de portadora)
CSMA/CD, 252-254
CSNET, 50
CTD (*véase* retardo de transferencia de células)
cuantización de vector, 733
cuenta de shell, 229
cuerpo, correo electrónico, 646
CVDT (*véase* tolerancia de variación de retardo de células)

D

daemon de vacaciones, 663
datagrama, 342
comparación con circuito virtual, 344-345
datos urgentes, 524
DCE (*véase* equipo de terminación de datos)
DCS 1800, 266
decibel, 81, 724
decodificación, 730
DES (*véase* estándar de cifrado de datos)
descriptor de tráfico, 461
desempeño de red, 555-572
IEEE 802.3, 283-285
desempeño, aspectos de, 555-572
desprendimiento de carga, 390-392
desvanecimiento multirayectoria, 99
detección de errores, 183-184
difusión, 8
digrama, 583
dirección, 492
de transporte, 489-492
dirección de difusión, 280
dirección de encargado, 433
dirección de multitransmisión, 280
dirección IP, 416-419
dirección jerárquica, 492
dirección plana, 492
directiva HTML, 696-699
dispersión, en fibra, 89
distancia de Hamming, 184
distorsión por retraso, 109

división, 748
 división de discos, 748
DMSP (véase protocolo de sistema de correo distribuido)
DNS (véase sistema de nombres de dominio)
 dominio, 623
DQDB (véase bus doble de colas distribuidas)
DS1, 121
DSMA (véase acceso múltiple con detección digital)
DSS (véase estándar de firmas digitales)
DTE (véase equipo de terminación de circuito de datos)
DVMRP (véase protocolo de enrutamiento de multitransmisión por vector de distancia)

E

EARN, 53
 elefantes, apocalipsis de los dos, 40
email (véase correo electrónico)
 encadenamiento de bloque cifrado, 590-591
encapsulado, Java, 713
 encolamiento justo ponderado, 388-389
 enmarcado, 179-182
 enrutador de multitransmisión, 757-758
 enrutador designado, 428
 enrutador m (véase enrutamiento de multitransmisión)
 enrutador multiprotocolo, 398
 enrutamiento adaptable, 347
 enrutamiento basado en flujo, 353-355
 enrutamiento Bellman-Ford, 355
 enrutamiento de multitransmisión, 372-374
 enrutamiento de sesión, 346
 enrutamiento desde el origen, 415-416
 enrutamiento entre redes, 405-406
 enrutamiento Ford-Fulkerson, 355
 enrutamiento interdominios sin clases, 434-437
 enrutamiento IS-IS, 365
 enrutamiento jerárquico, 365-367
 enrutamiento multidestino, 370
 enrutamiento no adaptable, 347
 enrutamiento por difusión, 370-372
 enrutamiento por estado de enlace, 359-365
 enrutamiento por trayectoria más corta, 348-352
 enrutamiento por vector de distancia, 355-359
 ensamblador desensamblador de paquetes, 60

entidad, 22
 entidad de transporte, 480
 entidad igual, 22
 entrega celular, 169
 envoltura síncrona de carga útil, 126-127
 equipo de terminación de circuito de datos, 114
 equipo de terminación de datos, 114
ER (véase tasa explícita)
 especificación de flujo, 384-386
 espectro electromagnético, 94-97
 espectro extendido, 96
 secuencia directa, 96
 establecer modo de respuesta normal extendido, 228
 establecer modo extendido balanceado asíncrono, 228
 establecimiento de conexión, 493-498
 TCP, 529-530
 estación administradora, 631
 estación de referencia, 329
 estándar de cifrado de datos, 588-595
 ataque, 592-595
 controversia, 593
 encadenamiento, 589-592
 estándar *de facto*, 67
 estándar de firmas digitales, 616
 estándar *de jure*, 67
 estándar internacional, 70
 estándar JPEG, 734-738
 estándar MPEG, 738-744, 753-754
 corrientes, 743
 Marco B, 742
 Marco I, 740
 Marco P, 740-741
 macrobloque, 740-741
 MPEG-1, 738-742
 MPEG-2, 742-744
 perfiles, 742
 estándares ISO
 ISO 3166, 623
 ISO 8802, 70, 275
 ISO 8859-1, 696
 estandarización
 Internet, 70-72
 ISO, 69-70
 red, 66-72
 telecomunicaciones, 67-69
 video a solicitud, 756-757
 estandarización de redes, 66-72

estandarización de telecomunicaciones, 67-69
 estrella pasiva, 92
 estructura de información de administración, 639-641
 éter luminífero, 276
Ethernet, 10, 276 (véase también IEEE 802.3)
 Ethernet comutado, 285-287
 Ethernet delgado, 277
 Ethernet rápido, 322-324
 etiqueta, HTML, 696-699
 extensiones multipropósito de correo de Internet, 653-657

F

factor de trabajo, 581
FAQ (véase preguntas comunes)
FCC (véase Comisión Federal de Comunicaciones)
FDDI (véase interfaz de datos distribuidos por fibra)
FDM (véase multiplexión por división en frecuencia)
 fibra hasta la acera, 116-118, 120, 751-752
 fibra hasta la casa, 116-118, 752-753
 fibra multimodo, 88
 fibra óptica, 87-94
 atenuación, 88-89
 comparación con el cobre, 92-94
 comparación con satélite, 168-170
 componentes de sistema, 86
 dispersión, 89
FDDI, 319-322
 multimodo, 88
 principios básicos, 87-88
SONET, 125-130
 unimodo, 88
WDM, 119-121
 fibra unimodo, 88
 fibras ópticas (véase también fibra óptica)
 multimodo, 88
 unimodo, 88
 ficha, 287-288, 293
 filtro de paquetes, 411
 firma digital, 613-620
 de clave pública, 615-616
 de clave secreta, 614-615
 fluctuación, 385, 724
 forma, HTML, 701-706
Foro ATM, 65

fragmentación interred, 406-409
 frecuencia, 94
FTP (véase protocolo de transferencia de archivos)
FTTC (véase fibra hasta la acera)
FTTH (véase fibra hasta la casa)
 Fuerza de Trabajo de Ingeniería de Internet, 71
 funcionamiento en cascada, 209
fuzzball, 50

G

gabinete de alambrado, 83
GCRA (véase algoritmo genérico de tasa de células)
Gopher, 693
 granja de discos, 748
 grupo, 119
 grupo de noticias
 creación, 674-675
 ejemplo, 673
 grupo de noticias moderado, 673
 grupo maestro, 119
GSM (véase sistema global de comunicación móvil)
 guerra de llamas, 672
 gusano, 720

H

haz localizador, 165
HDLC (véase control de enlace de datos de alto nivel)
HDTV (véase televisión de alta definición)
head-end, cable de, 85-86
HEC (véase control de errores de cabecera, ATM)
HEPNET, 53
 herencia, Java, 713
 herramientas de ventana abstracta, 717-718
HFC (véase híbrido fibra/coaxial)
 híbrido fibra/coaxial, 752
 hipermédia, 684
 hipertexto, 682
 hipervínculo, 682
HIPPI (véase interfaz paralela de alto desempeño)
 hoja de estilos, 698
 horizonte dividido, 358-359
host, 11
host móvil, algoritmo de enrutamiento, 367-370

- HTML (*véase* lenguaje de marcación de hipertexto)
 HTTP (*véase* protocolo de transferencia de hipertexto)
- ## I
- IAB (*véase* Consejo de Arquitectura de Internet)
 IBM, 41, 226, 307, 588, 593-594
 ICMP (*véase* protocolo de control de mensajes del Internet)
 iconos de emoción, 674
 IDEA (*véase* algoritmo internacional de cifrado de datos)
 identificador universal de recursos, 695
 IDU (*véase* unidad de datos de interfaz)
 IEEE, 70
 IEEE 802, 275-301
 comparación de LAN, 299-301
 IEEE 802.2, 302-304
 IEEE 802.3, 276-279
 cableado, 276-279
 codificación de señales, 279-280
 comutado, 285-287
 desempeño, 283-285
 Ethernet rápido, 322-324
 formato de marco, 281
 protocolo, 280-283
 IEEE 802.3u, 322-324
 IEEE 802.4, 287-292
 mantenimiento de anillo, 290-292
 protocolo, 288-290
 IEEE 802.5, 292-299
 mantenimiento de anillo, 298-299
 protocolo, 296-298
 IEEE 802.6, 301-303
 IETF (*véase* Fuerza de Trabajo de Ingeniería del Internet)
 IGMP (*véase* protocolo de mantenimiento de grupos del Internet)
 igual, 17
 IMAP (*véase* protocolo interactivo de acceso de correo)
 IMP (*véase* procesador de mensajes de interfaz)
 IMTS (*véase* servicio telefónico móvil mejorado)
 incorporación, 202-203

- indicación, 25-27
 ingreso remoto, 53
 interbloqueo de protocolos, 222
 intercambio de claves Diffie-Hellman, 605-606
 interconexión de redes, 396-412
 orientada a conexiones, 401-402
 por qué es necesaria, 399-400
 sin conexiones, 401-402
 interfaz de datos distribuidos por fibra, 319-322
 interfaz de pasarela común, 705-706
 interfaz digital de instrumentos musicales, 726
 interfaz entre capas, 18
 interfaz paralela de alto desempeño, 325-326
 interferómetro
 Fabry-Perot, 91, 261
 Mach-Zehnder, 91, 261
 Internet, 16
 administración de conexiones, 529-533
 capa de enlace de datos, 229-235
 capa de interred, 35-36, 412-449
 CIDR, 434-437
 historia, 52-54
 IP, 36, 412-419
 IP móvil, 432-434
 IPv6, 437-449
 multitransmisión, 431-432
 protocolos de enrutamiento, 424-431
 TCP, 36-37, 521-542
 interred, 16
 interrupción, SNMP, 632
 intruso, 580
 inundación, 351
 inundación selectiva, 351
 IP (*véase* protocolos de Internet)
 IP de línea en serie, 229-230, 685
 IP móvil, 432-434
 IPRA (*véase* Autoridad de Registro de Políticas de Internet)
 IPv4, 413-419
 IPv5, 438
 IPv6, 437-449
 cabecera de extensión, 443-446
 cabecera principal, 439-443
 controversias, 447-449
 direcciones, 441
 jumbograma, 445
 IPX, 46

- ISDN (*véase* red digital de servicios integrados)
 ISDN de banda ancha, 61-65, 144-155
 ISDN de banda angosta, 139-144
 ISO (*véase* Organización Internacional de Estándares)
 ITU (*véase* Unión Internacional de Telecomunicaciones)
 ITU-R, 68
 ITU-T, 68
 IXC (*véase* portadora entre centrales)
- ## J
- Java, 706-720
 API, 716-718
 clase, 713-716
 descripción del lenguaje, 709-718
 herramientas de ventana abstracta, 717-718
 orientación a objetos, 712-716
 polimorfismo, 715
 seguridad, 718-720
 jerarquía de almacenamiento, 746-747
 jerarquía de conmutadores, teléfono, 134-135
 jerarquía de protocolo, 17-20
 jerarquía digital sincrónica, 125-130
 jerarquías USENET, 671
 juicio final modificado, 104
 jumbograma, 445
- ## K
- KDC (*véase* centro de distribución de claves)
 Kerberos, 610-612
 knowbot, 720
- ## L
- LAN (*véase* red de área local)
 LAN de alta velocidad, 318-327
 LAN de ATM, 471-473
 LAN token bus (*véase* IEEE 802.4)
 LAN token ring, 292-299
 LAN voladora, 15
 LAP (*véase* procedimiento de acceso de enlace)
 LATA (*véase* área local de acceso y transporte)

- lazo local, 104, 108-118
 de fibra, 115-118
 LCP (*véase* protocolo de control de enlace)
 LEC (*véase* portadora de central local)
 lenguaje de marcación, 695
 lenguaje de marcación de hipertexto, 691-706
 formas, 701-706
 versiones, 699-701
 lenguaje de marcación estándar generalizado, 695
 LES (*véase* servidor de simulación de LAN)
 ley de Zipf, 746
 liberación de conexión, 498-502, 530-533
 liberación de una conexión, 498-502
 TCP, 530-533
 límite de Nyquist, 81
 límite de Shannon, 81-82
 línea asimétrica digital de suscriptor, 751
 línea de fases alternantes, 728-729
 línea de grado de voz, 79
 línea, SONET, 126
 LIS (*véase* subred lógica IP)
 lista de correo, 645-646
 LLC (*véase* control de enlace lógico)
 localizador uniforme de recursos, 692-695
 esquemas, 692-693
 longitud de onda, 94
 lotería china, 593
 luminancia, 728-729
- ## M
- MACA (*véase* acceso múltiple con prevención de colisiones)
 MACAW, 265
 macrobloque, 740-741
 mailto, 693
 MAN (*véase* red de área metropolitana)
 mapa activo, 684
 máquina de búsqueda, World Wide Web, 720-723
 máquina de estados finitos, 219-223, 519-521
 máquina de flujo de datos, 8
 marco
 acuse, 30
 de datos, 30
 de vídeo, 727
 Marco B, MPEG, 742
- 803

marco de información, 226-227
 marco de supervisión, 226-228
Marco I, MPEG, 740
 marco no numerado, 226-228
Marco P, MPEG, 740-741
 máscara de subred, 419
MBone (*véase backbone de multitransmisión*)
MCR (*véase tasa de células mínima*)
MD5, 618, 665
 media pasarela, 398
 medio físico, 18
 medios continuos, 724
 medios de transmisión, 82-94
 mensaje de crédito, 519
 método, HTTP, 690
MIB (*véase base de información de administración*)
MIDI (*véase interfaz digital de instrumentos musicales*)
MILNET, 50
MIME (*véase extensiones multipropósito de correo Internet*)
MNP5, 112
 modelo cliente-servidor, 3-4
 modelo de red de Petri, 223-224
 modelo de referencia, 28-44
 B-ISDN, 63-65
 comparación de OSI y TCP/IP, 38-39
 OSI, 28-35
 TCP/IP, 35-38
 modelo de referencia TCP/IP, 35-38, 43-44
 comparación con OSI, 38-39
 módem, 109-113
 módem nulo, 114
 modo de libro de código electrónico, 590
 modo de realimentación de cifrado, 591-592
 modo de realimentación de salida, 593
 modo de referencia OSI, 28-35
 comparación con TCP/IP, 38-39
 crítica, 40-43
 modo de transferencia asíncrona, 61-65
 calidad de servicio, 460-463
 canal virtual, 450
 capa de enlace de datos, 235-239
 categorías de servicio, 458-460
 conformación de tráfico, 463-468
 control de congestionamientos, 467-471
 cubeta con fugas, 466

modo de transferencia asíncrona (*cont.*)
 enrutamiento y conmutación, 455-458
 establecimiento de conexión, 452-455
 formato de célula, 450-452
 NNI, 450-451
 perspectiva, 65
 plano de control, 64
 plano de usuario, 64
 subcapa CS, 65
 subcapa PMD, 64
 subcapa SAR, 65
 subcapa TC, 64-65
 trayectoria virtual, 450
 UNI, 450-451
 modo promiscuo, 306
 modulación, 110-112
 amplitud, 110
 fase, 110
 frecuencia, 110
 modulación de amplitud en cuadratura, 111
 modulación delta, 123-124
 modulación por codificación de pulsos, 121
 Mosaic, 696
MOSPF (*véase OSPF de multitransmisión*)
MOTIS, 644
MSC (*véase centro de conmutación móvil*)
MTSO (*véase oficina de conmutación de telefonía móvil*)
MTU (*véase unidad máxima de transferencia*)
 multicomputadora, 8
 multimedia, 723-760
 audio, 724-726
 compresión de datos, 730-744
 MBone, 756-760
 vídeo, 727-730
 vídeo a solicitud, 744-756
 multiplexión, 118-130, 506-508
 ascendente, 506
 descendente, 507
 multiplexión por división en el tiempo, 118, 121-124, 330-333
 multiplexión por división en frecuencia, 118-121, 330
 multiplexión por división en longitud de onda, 119-121
 multitransmisión, 8, 372, 393-395
 Internet, 431-432
 multitransmisión independiente del protocolo, 760

N

N-ISDN (*véase ISDN de banda angosta*)
NAK (*véase acuse negativo*)
NAP (*véase punto de acceso a la red*)
National Institute of Standards and Technology, 70
National Security Agency, 593
National Television Standards Committee, 728-729
NCP (*véase protocolo de control de red*)
NCP (*véase protocolo de núcleo de red*)
Néctar, 56
 negociación, 26
 negociación de opciones, 483
NETBLT, 572
NetWare, Novell, 45-47
NIC (*véase centro de información de redes*)
NIST (*véase National Institute of Standards and Technology*)
NNTP (*véase protocolo de transferencia de noticias de red*)
 nodo de conmutación de paquetes, 12
 Notación de Sintaxis Abtracta 1, 633-636
 notación decimal con puntos, 417
 noticias, 53, 669-680, 693, 694
 noticias de red (*véase USENET*)
Novell NetWare, 45-47
NREN, 51-52
NSA (*véase National Security Agency*)
NSAP (*véase punto de acceso a servicio de red*)
NSFNET, 50-52
NT1, 140-142
NT2, 141-142
NTSC (*véase National Television Standards Committee*)
 nómico, 608

O

objeto
 Java, 713
 SNMP, 632, 641-642
OC-n (*véase portadora óptica*)
 oficina central, 104
 oficina central local, 104
 oficina de cargo, 104
 oficina de cinta rota, 133

oficina de conmutación de telefonía móvil, 159
 oficina tandem, 104
 oficina terminal, 104
 onda milimétrica, 100
ONU (*véase unidad de red óptica*)
 Organización Internacional de Estándares, 69
 oruga, 720
Oryctolagus cuniculus, 18
OSPF (*véase abrir primero la trayectoria más corta*)
OSPF de multitransmisión, 760

P
PAD (*véase ensamblador/desenamblador de paquetes*)
 página de Web, 682, 683, 697
PAL (*véase línea de fases alternantes*)
 paquete, 7
 paquete celular digital de datos, 15, 269-271
 paquete de estrangulamiento, 387-391
 paquete, Java, 713
 par trenzado, 83-84
 categoría 3, 83
 categoría 5, 84
 par trenzado sin blindaje, 84
 pared cortafuego, 410-412
 pasarela, 16
 pasarela de aplicación, 398, 411
 pasarela de transporte, 398
 patrón de constelación, 111-112
PBX (*véase central de ramal privado*)
PCA (*véase autoridad certificadora de políticas*)
PCM (*véase modulación por codificación de pulsos*)
 PCM diferencial, 123
PCN (*véase red personal de comunicaciones*)
PCR (*véase tasa de células pico*)
PCS (*véase servicios personales de comunicación*)
PDU (*véase unidad de datos de protocolo*)
PEM (*véase correo con confidencialidad mejorada*)
 perfil de usuario, 648
PES (*véase corriente elemental empacada*)
PGP (*véase confidencialidad bastante buena*)
 pila de protocolos, 18
PIM (*véase multitransmisión independiente del protocolo*)
 pixel, 729
 plano de control ATM, 64

plano de usuario, ATM, 64
 polimorfismo, Java, 715
 polinomio generador, 187
 política, 43
 política de "leche", 390
 política de sangrado, 390
 POP (*véase* punto de presencia)
 POP3 (*véase* protocolo de oficina postal-3)
 portadora
 comú, 67, 119
 módem, 110-111, 114
 portadora de central local, 106-107
 portadora de lado A, 160
 portadora de lado B, 160
 portadora de línea alámbrica, 160
 portadora entre centrales, 106-107
 portadora óptica, 128-129
 portadora T1, 121-122
 portadora T2, 124
 portadora T3, 124
 portadora T4, 124
 POTS (*véase* servicio telefónico tradicional)
 PPP (*véase* protocolo punto a punto)
 predicción de cabecera, 567
 preguntas comunes, 674
 prevención de congestionamientos, 378-379
 primitiva de servicio, 25-27
 ejemplo, 510-512
 primitivas de servicio de transporte, 483-486
 principal, 601
 principio de optimalidad, 347-348
 privacidad bastante buena, 664-667
 comparación con PEM, 669-670
 problema de conteo a infinito, 357-358
 problema de estación expuesta, 264
 problema de estación oculta, 264
 problema de los dos ejércitos, 499-500
 procedimiento avanzado de control de comunicación de datos, 226
 procedimiento de acceso a enlace, 226
 procesador de interfaz de terminal, 48
 procesador de mensajes de interfaz, 47
 procesamiento rápido de TPDU, 565-568
 proceso en túnel, 404-405
 producto ancho de banda-retardo, 557
 proporción de bloques de células con errores severos, 463

protocolo, 17
 802.5, 296-298
 AAL, 545-555
 AAL de ATM, 547-554
 acceso múltiple, 246-275
 ADCCP, 226
 ARP, 420-423
 ARQ, 200-202
 BGP, 429-431
 BOOTP, 424
 canal ruidoso, 197-200
 contención limitada, 256-259
 conteo descendente binario, 255-256
 CSMA, 250-254
 de 1 bit, 205-207
 DMSP, 662
 DSMA, 270-271
 DVMRP, 758-759
 elemental de enlace de datos, 190-202
 HDLC, 225-228
 HTTP, 689-691
 ICMP, 419-420
 IEEE 802.3, 280-283
 IEEE 802.4, 288-290
 IEEE 802.5, 296-298
 IGMP, 431-432
 IMAP, 662
 IP, 36, 412-419
 IPX, 46
 LAN inalámbrica, 262-265
 LAP, 226
 LAPB, 226
 LCP, 231
 libre de colisiones, 254-255
 MACA, 264-265
 MACAW, 265
 mapa de bits, 254-255
 NCP (protocolo de control de red), 231
 NCP (protocolo de núcleo de red), 46
 Needham-Schroeder, 608-609
 NNTP, 677-680
 Otway-Rees, 609-610
 PAR, 200-202
 pasarela exterior, 405-406
 pasarela interior, 405-406
 PIM, 760
 POP3, 662

protocolo (*cont.*)
 PPP, 231-235
 Q.2931, 453
 RARP, 423-424
 recorrido de árbol, 258-259
 red de gigabits, 568-572
 regresar n, 207-213
 repetición selectiva, 213-219
 reto-respuesta, 602-604
 RSVP, 394-396
 SDLC, 226-227
 simplex sin restricciones, 195-197
 SLIP, 229-230
 SMTP, 658-660
 SNMP, 642-643
 SSCOP, 555
 TCP, 36-37, 521-542
 UDP, 37, 542-544
 validación de identificación, 601-613
 ventana corrediza, 202-219
 WDMA, 260-262
 protocolo de acceso múltiple con detección de portadora, 250-254
 protocolo de administración de grupos de Internet, 431-432, 759
 protocolo de conexión inicial, 490
 protocolo de control de enlace, 231
 protocolo de control de mensajes de Internet, 419-420
 protocolo de control de red, 231
 protocolo de control de transmisión, 36-37, 521-542, 658, 678, 685
 administración de conexión, 529-533
 administración de temporizadores, 539-542
 algoritmo de Karn, 541
 algoritmo de Nagle, 534-535
 cabecera de segmento, 526-529
 control de congestionamientos, 536-539
 modelo de servicio, 523-524
 política de transmisión, 533-536
 redes inalámbricas, 543-545
 síndrome de ventana boba, 534-535
 protocolo de datagramas de usuario, 37, 542-544
 protocolo de enrutamiento de multitransmisión por vector de distancia, 758-759
 protocolo de mapa de bits, 254-255
 protocolo de núcleo de red, 46
 protocolo de oficina postal-3, 662

protocolo de pasarela exterior, 405-406, 424, 429-431
 protocolo de pasarela interior, 405-406, 424-429
 protocolo de recorrido de árbol, 258-259
 protocolo de regresar n, 207-213
 protocolo de repetición selectiva, 213-219
 protocolo de reserva de recursos, 394-395
 protocolo de resolución de direcciones, 420-423
 ARP gratuito, 433
 protocolo de resolución de direcciones en reversa, 423-424
 protocolo de reto-respuesta, 602-604
 protocolo de sapo de boca ancha, 607
 protocolo de servicio específico orientado a conexiones, 555
 protocolo de sistema de correo distribuido, 662
 protocolo de transferencia de archivos, 693
 protocolo de transferencia de hipertexto, 689-691
 protocolo de transporte, 488
 control de flujo, 502-506
 de Internet, 521-545
 direccionamiento, 489-492
 elementos, 488-510
 multiplexión, 506-508
 protocolo de validación de identificación, 601-613
 clave pública, 612-613
 Kerberos, 610-612
 usando KDC, 607-620
 protocolo de ventana corrediza, 202-219
 de 1 bit, 205-207
 protocolo interactivo de acceso al correo, 662
 protocolo libre de colisiones, 254-256
 protocolo Needham-Schroeder, 608-609
 protocolo Otway-Rees, 609-610
 protocolo punto a punto, 231-235, 685
 protocolo sencillo de administración de redes, 630-643
 protocolo sencillo de transferencia de correo, 658-660
 protocolo simple de Internet mejorado, 438
 protocolo SNMP, 642-643
 protocolos AAL, comparación, 554-555
 protocolos de acceso múltiple, 246-275
 protocolos de Internet
 ARP, 420-423, 433
 BGP, 429-431
 DVMRP, 758-759
 HTTP, 689-691
 ICMP, 419-420
 IGMP, 431-432, 759

protocolos de Internet (*cont.*)

- IP, 36, 412-419
- NNTP, 677-680
- OSPF, 424-429
- PIM, 760
- PPP, 231-235, 685
- RARP, 423-424
- RSVP, 394-395
- SLIP, 229-230, 685
- SMTP, 658-660
- TCP, 36-37, 521-542, 658, 678, 685
- UDP, 37, 542-544

protocolos de transferencia de noticias de red, 677-680

proveedor de servicio, 22

proveedor de servicio de transporte, 481

proveedor de servicio Internet, 229

PSTN (*véase* red telefónica pública comutada)

PTT (*véase* administración postal, telegráfica y telefónica)

publicaciones cruzadas, 672

puente, 304-318, 398

árbol de extensión, 310-313

de enrutamiento desde el origen, 314-316

entre LAN IEEE 802, 307-310

remoto, 317-318

transparente, 310-313

puerto bien conocido, 523

puerto, TCP, 523

bien conocido, 523

punto de acceso a servicio, 22

punto de acceso a servicio de red, 489

punto de acceso al servicio de transporte, 489

punto de acceso de red, 52

punto de cruce, 136

punto de presencia, 107

punto de referencia, ISDN, 142

PVC (*véase* circuito virtual permanente)

R

radio celular, 155-163

digital, 266-275

RAID (*véase* arreglo redundante de discos baratos)

RARP (*véase* protocolo de resolución de direcciones en reversa)

Recorte, 760

recuperación de caídas, 508-510

red ATM, 144-155

red de área amplia, 11-13

red de área local, 9-10

ATM, 471-473

de alta velocidad, 318-327

Ethernet, 10, 276-287

Ethernet rápido, 322-324

IEEE 802, 275-304

reparto de canales, 244-246

token bus, 287-292

token ring, 292-299

red de área metropolitana, 10-11

red de computadoras, 2

uso, 3-7

red de difusión, 7-8

red de distribución, 750-754

red de fibra óptica, 91-94

red de gigabit, 54-56, 568-572

red de punta, 430

red de tránsito, 430

red digital de servicios integrados, 61, 139-155

red multiacceso, 425

red personal de comunicaciones, 162-163

red punto a punto, 8

red satelital, 327-333

comparación con fibra, 168-170

comunicación, 163-179

de órbita baja, 167-170

geosíncrona, 164-167

red síncrona óptica, 125-130

red telefónica pública comutada, 102

redes inalámbricas, 13-15

hosts móviles, 367-370, 432-434

LAN inalámbricas, 262-265

ondas electromagnéticas, 94-101

radio analógica, 155-163

radio digital, 266-275

TCP inalámbrico, 543-545

reenviador anónimo, 674

reenvío por trayectoria invertida, 371-372

reflectometría de dominio de tiempo, 277

registro autorizado, 629

registro de recursos, 624-628

reino, Kerberos, 611

relación de señal a ruido, 81

relevador de células, 62

releyo de marco, 60-61

relleno de bits, 181

relleno de caracteres, 180-181

relleno de una sola vez, 585

repetición selectiva, 209

repetidor, 91-94, 279, 398

repetidor activo, 91

repetidor, satélite, 164

reservación de recursos, 468-469

resolvedor, DNS, 622

respuesta, 25-27

retardo de transferencia de células, 462

retroceso exponencial binario, 282-283

RFC (*véase también* solicitud de comentario)

RFC 768, 542

RFC 792, 420

RFC 793, 522

RFC 821, 651, 659, 761, 644

RFC 822, 644, 650, 651-653, 655, 660,

661, 665, 667, 676, 677, 688, 689,

690, 691, 761

RFC 826, 422

RFC 903, 423

RFC 951, 424

RFC 977, 677

RFC 1028, 630

RFC 1034, 622

RFC 1035, 622

RFC 1036, 676

RFC 1048, 424

RFC 1055, 229

RFC 1056, 662

RFC 1064, 622

RFC 1067, 630

RFC 1084, 424

RFC 1106, 528, 529

RFC 1112, 432

RFC 1122, 522

RFC 1144, 230

ÍNDICE

RFC (*cont.*)

RFC 1155, 630

RFC 1157, 630

RFC 1213, 642

RFC 1225, 662

RFC 1247, 424

RFC 1268, 431

RFC 1323, 522, 528

RFC 1421, 667

RFC 1422, 667

RFC 1423, 667

RFC 1424, 667

RFC 1425, 659

RFC 1441, 630

RFC 1442, 630, 639

RFC 1443, 630

RFC 1444, 630

RFC 1445, 630

RFC 1446, 630

RFC 1447, 630

RFC 1448, 630, 643

RFC 1449, 630

RFC 1450, 630

RFC 1451, 630

RFC 1452, 630

RFC 1483, 473, 554

RFC 1519, 435

RFC 1521, 653, 654, 655

RFC 1550, 437

RFC 1577, 342, 473, 554

RFC 1654, 431

RFC 1661, 231, 234

RFC 1662, 231

RFC 1663, 231

RFC 1700, 415, 523

RFC 1715, 443

RFC 1883, 438

RFC 1884, 438

RFC 1885, 438

RFC 1886, 438

RFC 1887, 438

rock and roll, relación de señal a ruido, 739

RS-232, 114-116

RS-422-A, 115

RS-423-A, 115

RS-449, 115-116

Q

Q.2931, 453

QAM (*véase* modulación de amplitud en cuadratura)

QoS (*véase* calidad de servicio)

- RSVP (*véase* protocolo de reserva de recursos)
 rueda de temporización, 567-568
 ruido, 109
 ruido de cuantización, 725
- S**
- SABME (*véase* establecer modo extendido balanceado asíncrono)
 SAP (*véase* punto de acceso a servicio)
 satélite de comunicaciones, 163-170
 satélite de órbita baja, 167-170
 satélite de tecnología de comunicación avanzada, 331
 satélite geosíncrono, 164-167
 SCR (*véase* tasa de células sostenida)
 SDH (*véase* jerarquía digital síncrona)
 SDLC (*véase* control síncrono de enlace de datos)
 SDU (*véase* unidad de datos de servicio)
 SEAL (*véase* capa de adaptación sencilla y eficiente)
 SECAM (*véase* *sequençiel couleur avec memoire*)
 SECBR (*véase* proporción de bloques de células con errores severos)
 sección, SONET, 126
 secuencia de *chips*, 272
 segmento, TCP, 525
 seguridad
 Java, 718-720
 teléfono celular, 161
 seguridad en las redes, 577-622
 señal síncrona de transporte-1, 126
 señalamiento asociado a canal, 122
 señalamiento en banda, 113
 señalización de canal común, 122
sequençiel couleur avec memoire, 728-729
 servicio
 de datagramas, 24-25
 de solicitud-respuesta, 24-25
 orientado a conexiones, 23-25
 sin conexiones, 23-25
 servicio confirmado, 26-27
 servicio de datos conmutado multimegabit, 57-59
 servicio de tasa constante de bits, 458-459
 servicio de tasa de bits variable, 459
 servicio de tasa de datos disponible, 459-460
 servicio de tasa de datos no especificada, 460
- servicio no confirmado, 26-27
 servicio telefónico móvil mejorado, 157
 servicio telefónico tradicional, 142
 servicios personales de comunicación, 162-163
 servidor apoderado, 688
 servidor ATMARP, 473
 servidor de archivos, 3
 servidor de difusión/desconocido, 472
 servidor de directorios, 491
 servidor de nombres, 491, 628-630
 servidor de procesos, 491
 servidor de simulación de LAN, 472
 servidor de vídeo, 745-750
 software, 747-749
 SGML (*véase* lenguaje de marcación estándar generalizado)
 SHA (*véase* algoritmo seguro de parcialización)
 sincronización, 33
 síndrome de ventana boba, 534-535
 sintaxis de transferencia ASN.1, 637-638
 SIPP (*véase* protocolo simple de Internet mejorado)
 sistema autónomo, 406, 412
 sistema de avisos, 155-156
 sistema de contención, 246-247, 252-258
 sistema de nombres de dominio, 421, 622-630
 sistema de telefonía móvil avanzada, 158-161
 sistema distribuido, 2
 sistema global de comunicación móvil, 266-275
 sistema intermedio, 12
 sistema presione para hablar, 157
 sistema telefónico, 102-163
 comunicación, 130-139
 lazo local, 108-118
 política, 106-108
 portadora T1, 121-122
 SONET, 125-130
 troncales y multiplexión, 118-130
 sistema terminal, 11
 SLIP (*véase* IP de línea en serie)
 SMDS (*véase* servicio de datos conmutado multimegabit)
 SMI (*véase* estructura de información de administración)
 SMTP (*véase* protocolo sencillo de transferencia de correo)
 SMTP extendido, 659
 SNA (*véase* arquitectura de red de sistemas)

- SNMP (*véase* protocolo sencillo de administración de redes)
 SNRME (*véase* establecer modo de respuesta normal extendido)
 sobre de correo electrónico, 646
 sobrecarga, Java, 715
 Sociedad Internet, 53, 71
 socket, 486-487
 software de red, 16-28
 solicitud, 25-27
 solicitud automática de repetición, 200-202
 solicitud de comentario, 71
 Soliton, 89
 sondeo, 328
 sondeo dirigido por interrupción, 632
 SONET (*véase* red óptica síncrona)
 SPADE, 330
 SPAN, 53
 SPE (*véase* envoltura síncrona de carga útil)
 SPX, 46
 SSCOP (*véase* protocolo de servicio específico orientado a conexiones)
 STS-1 (*véase* señal síncrona de transporte-1)
 subcapa cs, aim, 65
 subcapa de acceso al medio, 243-335
 subcapa de convergencia AAL, 546
 subcapa de línea, SONET, 129-130
 subcapa de sección, SONET, 129-130
 subcapa de trayectoria, SONET, 129-130
 subcapa fotónica, SONET, 129
 subcapa MAC (*véase* subcapa de control de acceso al medio)
 subcapa PMD, ATM, 64, 147, 235-239
 subcapa SAR, ATM, 65, 546
 subcapa TC, ATM, 64-65, 235-239
 subclase, Java, 713
 subred, 11
 Internet, 417-419
 subred de almacenamiento y reenvío, 12
 subred de comunicaciones, 11
 subred de conmutación de paquetes, 12
 subred IP lógica, 473
 subred punto a punto, 12
 suma de comprobación, 179, 182, 187-191, 235
 superclase, Java, 713
 supergrupo, 119
 supresor de eco, 112-113
- T**
- tabla de búsqueda de colores, 732
 tarifa, 67
 tasa básica ISDN, 142-143
 tasa de células mínima, 461
 tasa de células pico, 461
 tasa de células real, 471
 tasa de células sostenida, 461
 tasa de error de células, 463
 tasa de mala inserción de células, 463
 tasa de pérdida de células, 462
 tasa explícita, 471
 tasa primaria, ISDN, 142-143
 TCP (*véase* protocolo de control de transmisión)
 TCP indirecto, 543-544
 TDM (multiplexión por división en el tiempo)
 teléfono celular, 157-163
 administración de llamadas, 160-161
 AMPS, 158-161
 análogo, 157-161
 digital, 162
 seguridad, 161
 teléfono inalámbrico, 157
 televisión
 analógica, 727-729
 digital, 729-730
 televisión de alta definición, 729
 Telnet, 686-687, 693, 694
 temporización, ficha, 321
 temporizador de mantenimiento de vida, TCP, 542
 temporizador de persistencia, TCP, 542
 temporizador de retransmisiones, TCP, 539-540
 terminal de abertura muy pequeña, 165
 terminal virtual de red, 33
 texto cifrado, 580
 texto normal, 580
 tiempo de retención de ficha, 296
 TIP (*véase* procesador de interfaz de terminal)
 tolerancia de variación de retardo de células, 462
 tormenta de difusión, 557
 tostador en un poste, 168
 TPDU (*véase* unidad de datos de protocolo de transporte)
 trama de conmutación, 148
 transceptor, 277
 transferencia de archivos, 53

transformación por coseno discreto, 733
 transmisión
 de onda de luz, 100-102
 infraroja, 100
 transmisión a cualquiera, 442
 transmisión balanceada, 115
 transmisión de radio, 97-98
 transmisión no balanceada, 115
 transmisión por microondas, 98-99
 trayectoria virtual, ATM, 450
 trayectoria, SONET, 126
 tributario, SONET, 127
 trígrama, 583
 triple X, 60
 troncal, 11, 118-130
 troncal con conexión de cargo, 104
 troncal interoficinas, 104
 troncal intertarifas, 104
 FSAP (*véase* punto de acceso al servicio de transporte)
 tubo de bits, 140
 TV por cable, 85-86, 107, 144, 172

U

UBR (*véase* servicio de tasa de datos no especificada)
 UDP (*véase* protocolo de datagramas de usuario)
 umbral de congestionamiento, 538
 unidad de datos de protocolo, 22-23
 unidad de datos de protocolo de transporte, 484
 unidad de datos de servicio, 22
 unidad de interfaz de datos, 22
 unidad de red óptica, 751-752
 unidad máxima de transferencia, 525
 Unión Internacional de Telecomunicaciones, 68, 453, 471, 545, 634, 648, 734
 URI (*véase* identificador universal de recursos)
 URL (*véase* localizador uniforme de recursos)
 USENET, 669-680, 693
 implementación, 675-680
 relación con la Internet, 669
 vista de usuario, 670-675
 usuario de servicio, 22
 usuario de servicio de transporte, 481
 UTP (*véase* par trenzado sin blindaje)
 UUCP, 669

V

V.24, 114
 V.32, 111
 V.32 bis, 111
 V.34, 111
 V.42 bis, 112
 variación de retardo de células, 462
 VBR (*véase* servicio de tasa de bits variable)
 vector de inicialización, 590
 velocidad de la luz, 94
 ventana de congestionamiento, 537-538
 ventana receptora, 203
 ventana transmisora, 203
 verificación de protocolo, 219-224
 video, 727-730
 analógico, 727-729
 digital, 729-730
 entrelazado, 728
 progresivo, 728
 video a solicitud, 744-756
 caja de control, 754-756
 red de distribución, 750-754
 servidor, 745-750
 video compuesto, 728
 videoconferencia, 5
 vigilancia del tráfico, 379-380
 visor externo, 684
 VISTAnet, 56
 visualizador de la World Wide Web, 682
 VSAT (*véase* terminal de abertura muy pequeña)
 VTMP, 572

W

WAN (*véase* red de área amplia)
 WARC (*véase* Conferencia Mundial Administrativa de Radio)
 WDM (*véase* multiplexión por división en longitud de onda)
 WDMA (*véase* acceso múltiple por división de longitud de onda)
 Web (*véase* World Wide Web)
 World Wide Web, 54, 681-723
 CGI, 705-706

World Wide Web (cont.)
 hipermédia, 684
 hipertexto, 682
 hipervínculo, 682
 Java, 706-720
 lenguaje HTML, 691-706
 máquina de búsqueda, 720-723
 obtención de una página, 685-687
 protocolo HTTP, 689-691
 servidor, 685-689
 URL, 692-695
 visor externo, 684
 visualizador, 682
 WWV, 494
 WWW (*véase* World Wide Web)
 WYSIWYG, 695

X

X.3, 60
 X.21, 59
 X.25, 59-60
 X.28, 60
 X.29, 60
 X.400, 644, 661
 X.509, 668-669
 XTP, 572

Z

zona, DNS, 628

Acerca del autor

Andrew S. Tanenbaum tiene una licenciatura en ciencias del M.I.T. y un doctorado en filosofía por la University of California at Berkeley. Actualmente es profesor de informática en la Vrije Universiteit de Amsterdam, Países Bajos, donde encabeza el Grupo de Sistemas de Computadoras. También es director de la Escuela Avanzada de Computación e Imágenes, una escuela interuniversitaria de posgrado en la que se lleva a cabo investigación sobre sistemas paralelos avanzados, sistemas distribuidos y sistemas de imágenes. No obstante, hace un gran esfuerzo por evitar convertirse en un burócrata.

En el pasado, el autor ha efectuado investigaciones sobre compiladores, sistemas operativos, redes y sistemas distribuidos de área local. Sus investigaciones actuales se enfocan principalmente al diseño de sistemas distribuidos de área amplia en la escala de millones de usuarios. Estos proyectos de investigación han conducido a más de 70 artículos arbitrados en publicaciones periódicas y actas de conferencias. También es autor de cinco libros (véase la página ii).

El profesor Tanenbaum también ha producido un volumen considerable de *software*: fue el arquitecto principal del *Amsterdam Compiler Kit*, un grupo de herramientas de amplio uso para escribir compiladores portátiles, y de MINIX, un pequeño sistema operativo tipo UNIX para cursos de sistemas operativos. Junto con sus estudiantes de doctorado y programadores, ayudó a diseñar Amoeba, un sistema operativo distribuido de alto desempeño basado en micronúcleo. MINIX y Amoeba ahora están disponibles gratuitamente en la Internet para educación e investigación a través de Internet.

Los estudiantes de doctorado del profesor Tanenbaum han seguido su camino para cosechar grandes triunfos tras recibir sus títulos. Él está muy orgulloso de ellos; en este sentido se parece a la mamá gallina.

El profesor Tanenbaum es socio del ACM, miembro senior del IEEE, miembro de la Academia Real Holandesa de Ciencias y Artes, y ganador en 1994 del premio Karl V. Karlstrom del ACM para educadores sobresalientes. También está listado en el *Who's Who in the World*. Su página base de la *World Wide Web* se encuentra en <http://www.cs.vu.nl/~ast/>.