

Redes de computadoras

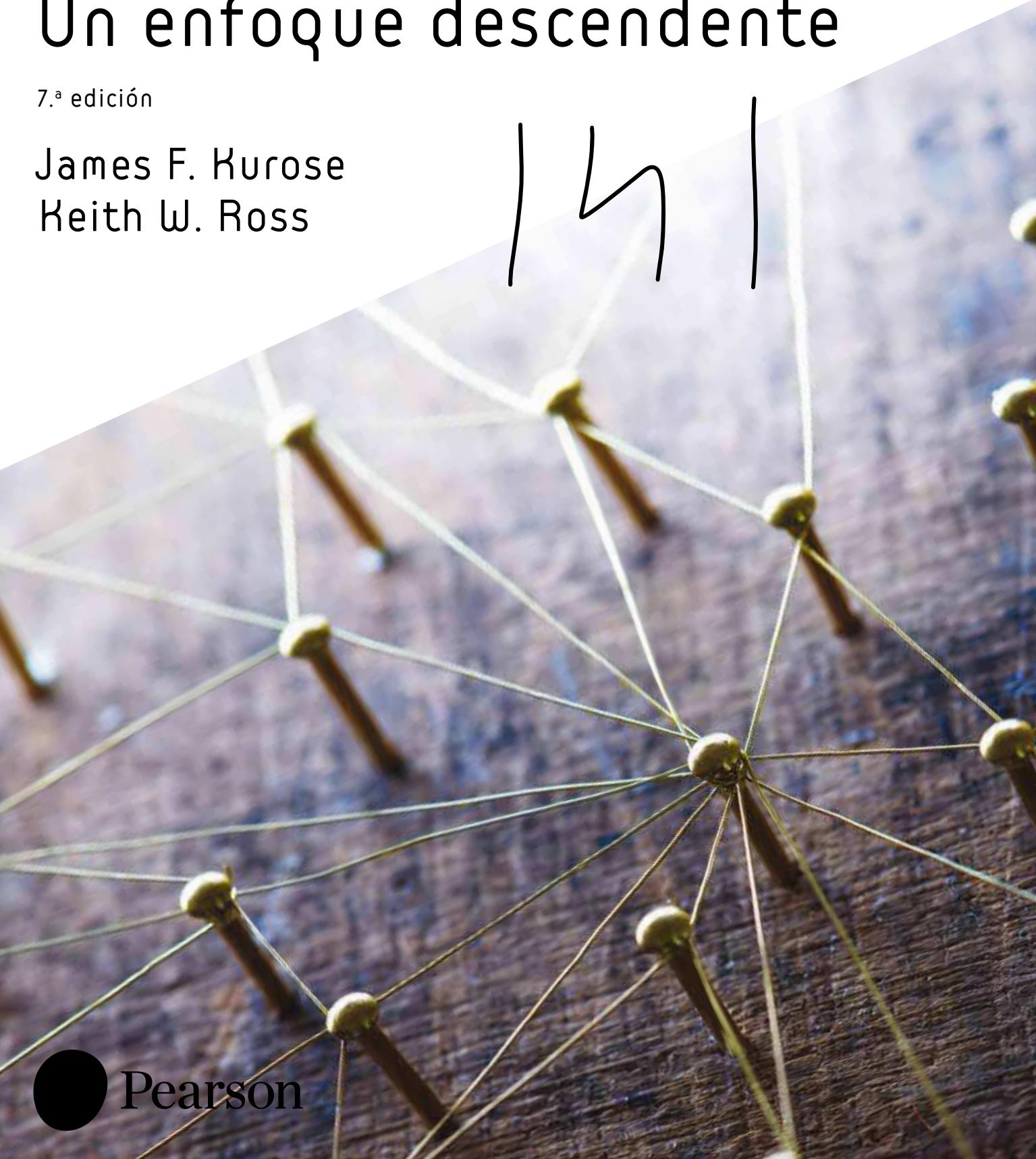
Un enfoque descendente

7.^a edición

James F. Kurose
Keith W. Ross



Pearson



REDES DE COMPUTADORAS

Un enfoque descendente

SÉPTIMA EDICIÓN

Redes de computadoras. Un enfoque descendente, 7 Ed

James F .Kurose; Keith W. Ross

PEARSON EDUCACIÓN, S. A., Madrid, 2017
ISBN: 978-84-9035-528-2
ISBN e-Book: 978-84-9035-529-9

Materia: Informática, 004

Formato: 215 x 270mm. Páginas: 704

Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra sólo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y sgts. Código penal). d e s c a r g a d o e n : e y b o o k s . c o m

Diríjase a CEDRO (Centro Español de Derechos Reprográficos -www.cedro.org), si necesita fotocopiar o escanear algún fragmento de esta obra.

Todos los derechos reservados.

© PEARSON, S.A., 2017

C/ Ribera del Loira, 16-18
28042 Madrid (España)
www.pearson.es

ISBN: 978-84-9035-528-2

ISBN e-Book: 978-84-9035-529-9

Depósito Legal: M-7106-2017

Equipo de edición:

Editor: Miguel Martín-Romo

Equipo de diseño:

Diseñadora Senior: Elena Jaramillo

Equipo de producción:

Directora de producción: Marta Illescas

Coordinadora de producción: Tini Cardoso

Diseño de cubierta: Pearson Educación S. A.

Traducción y Composición: Vuelapluma, S.L.

Impreso por:

Nota sobre enlaces a páginas web ajenas: Este libro incluye enlaces a sitios web cuya gestión, mantenimiento y control son responsabilidad única y exclusiva de terceros ajenos a PEARSON EDUCACIÓN, S. A. Los enlaces u otras referencias a sitios web se incluyen con finalidad estrictamente informativa y se proporcionan en el estado en que se encuentran en el momento de publicación sin garantías, expresas o implícitas, sobre la información que se proporcione en ellas. Los enlaces no implican el aval de PEARSON EDUCACIÓN S. A. a tales sitios, páginas web, funcionalidades y sus respectivos contenidos o cualquier asociación con sus administradores. En consecuencia, PEARSON EDUCACIÓN S. A., no asume responsabilidad alguna por los daños que se puedan derivar de hipotéticas infracciones de los derechos de propiedad intelectual y/o industrial que puedan contener dichos sitios web ni por las pérdidas, delitos o los daños y perjuicios derivados, directa o indirectamente, del uso de tales sitios web y de su información. Al acceder a tales enlaces externos de los sitios web, el usuario estará bajo la protección de datos y políticas de privacidad o prácticas y otros contenidos de tales sitios web y no de PEARSON EDUCACIÓN S. A.

Este libro ha sido impreso con papel y tintas ecológicas

REDES DE COMPUTADORAS

Un enfoque descendente

SÉPTIMA EDICIÓN

JAMES F. KUROSE

University of Massachusetts, Amherst

KEITH W. ROSS

NYU and NYU Shanghai

REVISIÓN TÉCNICA

Carolina Mañoso Hierro

Profesora Titular de Universidad

Dpto. de Sistemas de Comunicación y Control

Escuela Técnica Superior de Ingeniería Informática

Universidad Nacional de Educación a Distancia

Ángel Pérez de Madrid y Pablo

Profesor Titular de Universidad

Dpto. de Sistemas de Comunicación y Control

Escuela Técnica Superior de Ingeniería Informática

Universidad Nacional de Educación a Distancia



Pearson

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

A Julie y a nuestras tres preciosidades
—Chris, Charlie y Nina
JFK

Un enorme AGRADECIMIENTO a mis profesores,
colegas y estudiantes de todo el mundo.
KWR

Acerca de los autores

Jim Kurose

Jim Kurose es un distinguido profesor universitario de Ciencias de la Computación en la Universidad de Massachusetts, Amherst. Actualmente disfruta en la universidad de una excedencia y trabaja como director adjunto en la US National Science Foundation, donde dirige la división de Ingeniería y ciencias de la computación y la información.

Kurose ha recibido numerosos reconocimientos por sus actividades en el campo de la educación, entre los que se incluyen el premio a la excelencia en la labor pedagógica de la Universidad Tecnológica Nacional (ocho veces), la Universidad de Massachusetts y la asociación Northeast Association of Graduate Schools. Ha recibido la Medalla Taylor Booth del IEEE a la Educación y es bien conocido por haber liderado la iniciativa Commonwealth Information Technology de Massachusetts. Ha ganado varios premios al mejor artículo y recibido el *IEEE Infocom Achievement Award* y el *ACM Sigcomm Test of Time Award*.



El Dr. Kurose ha sido editor jefe de las revistas *IEEE Transactions on Communications* e *IEEE/ACM Transactions on Networking*. Ha participado varios años en los comités de programa de las conferencias *IEEE Infocom*, *ACM SIGCOMM*, *ACM Internet Measurement Conference* y *ACM SIGMETRICS*. Es miembro del IEEE de la ACM. Entre sus intereses de investigación se incluyen las arquitecturas y protocolos de red, las medidas de red, las redes de sensores, la comunicación multimedia y el modelado y la evaluación de rendimiento. Es doctor en Ciencias de la Computación por la Universidad de Columbia.

Keith Ross

Keith Ross es el decano de Ingeniería y Ciencias de la Computación en NYU Shanghai y ostenta la cátedra Leonard J. Shustek en el departamento de Ingeniería y Ciencias de la Computación de NYU. Anteriormente trabajó en la Universidad de Pennsylvania (13 años), Eurecom Institute (5 años) y en la Universidad Politécnica (10 años). Se graduó en la Universidad Tufts, tiene un máster por la Universidad de Columbia y es doctor en Ingeniería de Control y Computación por la Universidad de Michigan. Keith Ross fue también el fundador principal y primer consejero delegado de Wimba, que desarrolla aplicaciones multimedia en línea para e-learning y fue adquirida por Blackboard en 2010.



Entre los intereses de investigación del profesor Ross se encuentran las redes sociales, las redes P2P, las medidas para Internet, las redes de distribución de contenido y el modelado estocástico. Es miembro de ACM y del IEEE, ha recibido el premio Infocom 2009 al mejor artículo de investigación y fue galardonado también en 2011 y 2008 con los premios al mejor artículo sobre Comunicaciones multimedia por la IEEE Communications Society. Ha sido miembro del comité editorial de numerosas revistas y de los comités de programa de numerosas conferencias, incluyendo *IEEE/ACM Transactions on Networking*, *ACM SIGCOMM*, *ACM CoNext* y *ACM Internet Measurement Conference*. También ha actuado como asesor de la Comisión Federal de Comercio de los EE.UU: para el tema de la compartición de archivos P2P.

Prefacio

Bienvenido a la séptima edición de *Redes de computadoras: un enfoque descendente*. Desde la publicación de la primera edición hace 16 años, nuestro libro ha sido adoptado por centenares de universidades, traducido a 14 idiomas y utilizado por más cien mil estudiantes y profesionales de todo el mundo. Hemos tenido noticias de muchos de estos lectores y estamos abrumados por su positiva respuesta.

Novedades en la séptima edición

Creemos que una razón importante de este éxito ha sido que nuestro libro continúa ofreciendo un enfoque novedoso y oportuno para la formación en el campo de las redes de computadoras. Hemos realizado cambios en esta séptima edición, pero también hemos conservado todo aquello que creemos que constituyen (y en lo que coinciden) los estudiantes y los profesores que han utilizado nuestro libro) los aspectos más importantes del libro: su enfoque descendente, el hecho de que está centrado en Internet y en un tratamiento moderno de las redes de computadoras, su atención tanto a los principios como a la práctica y su estilo y enfoque pedagógico accesibles en lo que respecta al aprendizaje de las redes de computadoras. No obstante, hemos revisado y actualizado esta séptima edición de forma sustancial.

Los lectores familiarizados con las ediciones anteriores del libro observarán que, por primera vez desde que publicamos este texto, hemos cambiado la organización de los propios capítulos. La capa de red, que antes se cubría en un único capítulo, ahora se trata en el Capítulo 4 (que se centra en el denominado “plano de datos” de la capa de red) y en el Capítulo 5 (centrado en el “plano de control” de la capa de red). Este tratamiento ampliado de la capa de red refleja el rápido crecimiento de la importancia de las redes definidas por software (SDN), tal vez el más importante y fascinante avance de las últimas décadas en el campo de las redes. Aunque se trata de una innovación relativamente reciente, la tecnología SDN ha sido rápidamente adoptada en la práctica: tanto, que ya resulta difícil imaginar una introducción a las modernas redes de computadoras que no cubra SDN. El tema de la gestión de red, que antes se trataba en el Capítulo 9, ha sido ahora integrado en el nuevo Capítulo 5. Como siempre, hemos actualizado también muchas otras secciones del texto, para reflejar los recientes cambios que el dinámico campo de las redes ha experimentado desde la sexta edición. También como siempre, el material que ha sido retirado del texto impreso puede consultarse en el sitio web de acompañamiento del libro. Las actualizaciones más importantes son las siguientes:

- Hemos actualizado el Capítulo 1 para reflejar el alcance y utilización crecientes de Internet.
- El Capítulo 2, que trata la capa de aplicación, ha sido significativamente actualizado. Hemos eliminado el material sobre el protocolo FTP y las tablas hash distribuidas, para hacer sitio para una nueva sección sobre flujos de vídeo de nivel de aplicación y redes de distribución de contenido, junto con los casos de estudio de Netflix y YouTube. Las secciones de programación de sockets se han actualizado de Python 2 a Python 3.
- El Capítulo 3, dedicado a la capa de transporte, solo se ha actualizado ligeramente. El material sobre las redes de modo de transporte asíncrono (ATM) ha sido sustituido por material más moderno sobre la notificación explícita de congestión (ECN) de Internet, que enseña los mismos principios.
- El Capítulo 4 cubre el “plano de datos” de la capa de red: la función de reenvío *en cada router* que determina cómo se reenvía hacia uno de los enlaces de salida de un router un paquete que llegue a uno de los enlaces de entrada de ese router. Hemos actualizado el material sobre el reenvío tradicional de Internet incluido en todas las ediciones anteriores, y hemos añadido nuevo material

sobre la planificación de paquetes. También hemos añadido una nueva sección sobre reenvío generalizado, como el utilizado en SDN. También hay numerosas otras actualizaciones por todo el capítulo. El material sobre comunicaciones de difusión y multidifusión se ha eliminado, para hacer sitio al nuevo material.

- En el Capítulo 5, cubrimos las funciones del plano de control de la capa de red: la lógica *global de la red* que controla el modo en que se enruta un datagrama a lo largo de un camino extremo a extremo de routers, desde el host de origen hasta el host de destino. Como en las ediciones anteriores, cubrimos tanto los algoritmos como los protocolos de enrutamiento (con un tratamiento actualizado de BGP) que se usan en la Internet actual. Hemos añadido una importante sección nueva sobre el plano de control en SDN, en el que el enrutamiento y otras funciones se implementan en los denominados controladores SDN.
- El Capítulo 6, que ahora cubre la capa de enlace, incluye un tratamiento actualizado de Ethernet y de las redes de centros de datos.
- El Capítulo 7, dedicado a las redes inalámbricas y móviles, contiene material actualizado sobre las redes 802.11 (denominadas “redes WiFi”) y las redes celulares, incluyendo 4G y LTE.
- El Capítulo 8, que trata de la seguridad de red y fue actualizado en profundidad en la sexta edición, solo tiene pequeñas actualizaciones en esta séptima edición.
- El Capítulo 9, sobre redes multimedia, es ahora ligeramente más corto que en la sexta edición, ya que el material sobre flujos de vídeo y redes de distribución de contenidos se ha pasado al Capítulo 2 y el material sobre planificación de paquetes se ha incorporado al Capítulo 4.
- Se ha añadido bastante material nuevo en relación con los problemas del final de cada capítulo. Como en todas las ediciones anteriores, se han revisado, añadido y eliminado los problemas asignados a los lectores.

Como siempre, nuestro objetivo al desarrollar esta nueva edición de nuestro libro es continuar proporcionando un tratamiento centrado y moderno de las redes de computadoras, que pone el énfasis tanto en los principios como en la práctica.

Audiencia

Este libro de texto es para un primer curso sobre redes de computadoras. Se puede utilizar tanto en departamentos de informática como de ingeniería eléctrica. En relación con los lenguajes de programación, se supone que el estudiante solo tiene experiencia con C, C++, Java o Python (e incluso esa suposición solo se hace en algunos pocos lugares). Aunque este libro es más preciso y analítico que muchos otros textos introductorios a las redes de computadoras, rara vez utiliza conceptos matemáticos que no se hayan aprendido en el bachillerato. Hemos hecho un esfuerzo deliberado por evitar el uso de cualquier concepto de cálculo avanzado, probabilidad o procesos estocásticos (aunque hemos incluido algunos problemas para los estudiantes que dominen estos conceptos). El libro es apropiado por tanto para cursos de primer ciclo y para el primer año de los cursos de segundo ciclo. También puede ser adecuado para los trabajadores del sector de las telecomunicaciones.

¿Qué hace especial a este libro de texto?

El tema de las redes de computadoras es enormemente complejo, implicando muchos conceptos, protocolos y tecnologías que están entrelazados de una manera intrincada. Para abarcar este alcance y complejidad, muchos textos de redes se han organizado a menudo sobre las “capas” de una arquitectura de red. Con una organización en capas, los estudiantes pueden ver a través de la complejidad de las redes de computadoras y aprender los distintos conceptos y protocolos de una parte de la arquitectura a la vez que ven el gran esquema de cómo todas las partes se ajustan entre sí. Desde una perspectiva pedagógica, nuestra experiencia personal ha sido que dicho enfoque en capas es efecti-

vamente muy deseable. Sin embargo, hemos comprobado que el enfoque tradicional de enseñanza ascendente, es decir, desde la capa física a la capa de aplicación, no es el mejor enfoque para un curso moderno sobre redes de computadoras.

Un enfoque descendente

Nuestro libro rompió moldes hace 16 años al abordar el tema de las redes de arriba hacia abajo; es decir, comenzando con la capa de aplicación y descendiendo desde allí hacia la capa física. Los comentarios recibidos tanto de profesores como de estudiantes nos confirman que el enfoque descendente tiene muchas ventajas y resulta pedagógicamente adecuado. En primer lugar, hace énfasis en la capa de aplicación (un “área de elevado crecimiento” en las redes). De hecho, muchas revoluciones recientes en las redes de computadoras, incluyendo la Web, la compartición de archivos P2P y los flujos multimedia, han tenido lugar en la capa de aplicación. Un énfasis inicial en la capa de aplicación difiere de los métodos considerados en la mayor parte de otros textos, que incluyen solo una pequeña cantidad de material sobre las aplicaciones de red, sus requisitos, paradigmas de la capa de aplicación (por ejemplo, cliente-servidor y P2P) e interfaces de programación de aplicaciones. En segundo lugar, nuestra experiencia como profesores (y de muchos profesores que han utilizado este texto) ha sido que el enseñar las aplicaciones de redes al principio de curso es una potente herramienta de motivación. Los estudiantes se emocionan al aprender cómo funcionan las aplicaciones de red, aplicaciones tales como el correo electrónico o la Web, que la mayoría de los estudiantes utilizan diariamente. Una vez que los estudiantes comprenden las aplicaciones, pueden entonces entender los servicios de red necesarios para proporcionarles ese soporte. El estudiante puede entonces, a su vez, examinar las distintas formas en que tales servicios pueden ser suministrados por, e implementados en, las capas inferiores. Por tanto, tratar las aplicaciones inicialmente proporciona motivación para abordar el resto del texto.

En tercer lugar, un enfoque descendente permite a los profesores introducir el desarrollo de aplicaciones de red en una etapa temprana. Los estudiantes no solo ven cómo funciona una serie de populares aplicaciones y protocolos, sino que también aprenden lo fácil que es crear sus propias aplicaciones de red y protocolos del nivel de aplicación. Con el enfoque descendente, los estudiantes descubren pronto las nociones acerca de la programación de sockets, los modelos de servicio y los protocolos (conceptos importantes que vuelven a serlo en todas las capas subsiguientes). Al proporcionar ejemplos de programación de sockets en Python, destacamos las ideas centrales sin confundir a los estudiantes con fragmentos complejos de código. Los estudiantes de primer ciclo de Ingeniería Eléctrica y Ciencias de la Computación no deben tener dificultades para comprender el código Python.

Un enfoque Internet

Aunque en la cuarta edición del libro eliminamos del título la frase “Caracterización de Internet”, ¡no quiere decir que hayamos eliminado el enfoque de Internet! ¡Nada más lejos de la realidad! En lugar de ello, y dado que Internet se ha vuelto tan ubicua, pensamos que cualquier libro de texto sobre redes debe centrarse significativamente en Internet, por lo que la frase del título era hasta cierto punto innecesaria. En el texto, continuamos utilizando la arquitectura y los protocolos de Internet como vehículo principal para estudiar los conceptos fundamentales de las redes de computadoras. Por supuesto, también incluimos conceptos y protocolos de las arquitecturas de red, pero el foco está centrado claramente en Internet, un hecho que se ha reflejado en la organización del libro, que está centrada alrededor de la arquitectura de cinco capas de Internet: las capas de aplicación, transporte, red, enlace y física.

Otro de los beneficios de centrar la atención en Internet es que la mayoría de los estudiantes de Ciencias de la Computación y de Ingeniería Eléctrica están deseosos de aprender cosas sobre Internet y sus protocolos. Saben que Internet ha sido una tecnología revolucionaria y conflictiva que está cambiando profundamente nuestro mundo. Dada la enorme relevancia de Internet, los

estudiantes sienten una natural curiosidad por ver lo que hay “entre bastidores”. Por tanto, es fácil para un profesor conseguir que los estudiantes se interesen por los principios básicos cuando se utiliza Internet como foco guía.

Cómo enseñar los principios de las redes

Dos de las características únicas del libro (su enfoque descendente y su foco puesto en Internet) se han utilizado en el título en sucesivas ediciones. Si hubiéramos podido introducir una *tercera* frase en el subtítulo, esta habría contenido la palabra *principios*. El campo de las redes está ahora lo suficientemente maduro como para que se puedan identificar una serie de temas fundamentalmente importantes. Por ejemplo, en la capa de transporte, los temas fundamentales son la comunicación fiable sobre una capa de red no fiable, el establecimiento y el cierre de la conexión y el proceso de acuerdo, el control de congestión y flujo, y la multiplexación. Tres temas enormemente importantes de la capa de red son la determinación de “buenas” rutas entre dos routers y la interconexión de un número grande de redes heterogéneas y la gestión de la complejidad de una red moderna. En la capa de enlace de datos, un problema fundamental es la compartición de un canal de acceso múltiple. En el campo de la seguridad de red, las técnicas que permiten proporcionar confidencialidad, autenticación e integridad de los mensajes están basadas en los principios de la criptografía. Este texto identifica los temas fundamentales acerca de las redes y estudia los métodos para abordarlos. El estudiante que aprenda estos principios adquirirá una serie de conocimientos con una larga “vida útil”: mucho después de que los estándares y protocolos de red actuales hayan quedado obsoletos, los principios en que se basan continuarán teniendo importancia y siendo relevantes. Pensamos que la combinación del uso de Internet para atisbar las posibilidades y el énfasis posterior en los temas fundamentales y las soluciones permitirán al estudiante comprender rápidamente cualquier tecnología de redes.

El sitio web

Al adquirir este libro de texto, el lector podrá acceder durante doce meses al sitio web de acompañamiento en la dirección <http://www.pearsonhighered.com/cs-resources/>. Este sitio incluye:

- *Material de aprendizaje interactivo.* El sitio web de acompañamiento del libro contiene Notas de vídeo —presentaciones de vídeo de temas importantes de todo el libro hechas por los autores, así como resoluciones paso a paso de problemas similares a los del final de los capítulos—. Inicialmente, hemos incluido en el sitio web VideoNotes y problemas en línea para los Capítulos 1 a 5, y continuaremos añadiendo y actualizando este material a lo largo del tiempo. Como en ediciones anteriores, el sitio web contiene applets Java interactivas con animaciones de muchos conceptos clave de las redes. El sitio contiene también exámenes interactivos que permiten a los estudiantes comprobar su comprensión básica de los temas tratados. Los profesores pueden integrar estas características interactivas en sus clases y usarlas como miniprácticas de laboratorio.
- *Material técnico adicional.* A medida que hemos ido añadiendo nuevo material en cada edición del libro, hemos tenido que eliminar parte del tratamiento de alguno de los temas existentes, con el fin de que el libro tuviera una longitud aceptable. Por ejemplo, para hacer sitio para el nuevo material incluido en esta edición, hemos eliminado diversos materiales sobre FTP, tablas hash distribuidas y multidifusión. Pero el material incluido en las ediciones anteriores del libro sigue siendo de interés y puede encontrarse en el sitio web de acompañamiento.
- *Tareas de programación.* El sitio web también proporciona una serie de tareas de programación detalladas, entre las que se incluyen la construcción de un servidor web multihebra, la construcción de un cliente de correo electrónico con una interfaz gráfica de usuario (GUI), la programación de los lados emisor y receptor de un protocolo de transporte de datos fiable, la programación de un algoritmo de enrutamiento distribuido y otros.

- *Prácticas de laboratorio con Wireshark.* La comprensión de los protocolos de red puede verse ampliada significativamente viendo esos protocolos en acción. El sitio web proporciona numerosas prácticas con Wireshark que permiten a los estudiantes observar la secuencia de mensajes intercambiados entre dos entidades de protocolo. El sitio web incluye prácticas de laboratorio con Wireshark independientes sobre HTTP, DNS, TCP, UDP, IP, ICMP, Ethernet, ARP, WiFi, SSL, y sobre cómo realizar una traza de todos los protocolos implicados en satisfacer una solicitud de extracción de una página web. Continuaremos añadiendo nuevas prácticas de laboratorio en el futuro.

Además del sitio web de acompañamiento, los autores mantienen un sitio web público, http://gaias.cs.umass.edu/kurose_ross/interactive, que contiene ejercicios interactivos que generan problemas similares a algunos de los incluidos al final de los capítulos (junto con sus correspondientes soluciones). Dado que los estudiantes pueden generar un número ilimitado de problemas similares (junto con sus soluciones), pueden trabajar hasta dominar completamente el material.

Características pedagógicas

Cada uno de nosotros lleva enseñando redes de computadoras durante más de 30 años. Proporcionamos a este texto la experiencia combinada de enseñar durante 60 años a muchos miles de estudiantes. Hemos sido también investigadores activos en redes de computadoras durante este tiempo. (De hecho, Jim y Keith se conocieron cuando eran estudiantes de máster en un curso sobre redes de computadoras impartido por Mischa Schwartz en 1979 en la Universidad de Columbia.) Pensamos que todo esto nos proporciona una buena perspectiva de cuál ha sido la evolución de las redes y hacia dónde es probable que vayan en el futuro. Sin embargo, hemos resistido a la tentación de dirigir el material

de este libro hacia las preferencias de nuestros proyectos de investigación. Consideramos que el lector puede visitar nuestros sitios web personales si está interesado en nuestras investigaciones. Por tanto, este libro se ocupa de las redes de computadoras modernas (de los protocolos y tecnologías contemporáneos, así como de los principios subyacentes a dichos protocolos y tecnologías). También creemos que aprender (y enseñar) redes puede ser divertido. Esperamos que el sentido del humor, el uso de analogías y los ejemplos del mundo real contenidos en el libro hagan el material más divertido.

Suplementos para los profesores

Proporcionamos un paquete de suplementos completo para ayudar a los profesores a la hora de impartir este curso. Se puede acceder a este material en el Centro de recursos de formación de Pearson (<http://www.pearsonhighered.com/irc>). Visite este centro para obtener información sobre cómo acceder a los suplementos para los profesores.

- *Diapositivas PowerPoint®.* Proporcionamos diapositivas para los nueve capítulos. Las diapositivas para esta séptima edición se han actualizado de forma significativa. Estas presentaciones en diapositivas cubren cada capítulo en detalle. Se utilizan gráficos y animaciones (en lugar de emplear únicamente monótonas listas de viñetas) que hacen que las diapositivas sean interesantes y visualmente atractivas. Proporcionamos las diapositivas originales de PowerPoint de modo que pueda personalizarlas con el fin de adaptarlas a sus necesidades a la hora de impartir el curso. Algunas de estas diapositivas han sido aportadas por otros profesores que han enseñado con nuestro libro.
- *Soluciones de los problemas.* Proporcionamos un manual de soluciones para los problemas incluidos en el texto, las tareas de programación y las prácticas de laboratorio con Wireshark. Como hemos dicho anteriormente, hemos añadido muchos problemas nuevos en los primeros seis capítulos del libro.

Dependencias de los capítulos

El primer capítulo de este texto presenta una panorámica general autocontenido de las redes de computadoras, introduciéndose muchos conceptos clave y terminología; este capítulo define el escenario para el resto del libro. Todos los capítulos restantes dependen de este primer capítulo. Recomendamos a los profesores que, después de completar el Capítulo 1, aborden los Capítulos 2 a 6 en orden, siguiendo nuestra filosofía del enfoque descendente. Cada uno de estos cinco capítulos se apoya en el material de los capítulos anteriores. Una vez completados los seis primeros capítulos, el profesor tiene bastante flexibilidad. No existen interdependencias entre los tres últimos capítulos, por lo que se pueden abordar en cualquier orden. Sin embargo, cada uno de los tres últimos capítulos depende del material de los seis primeros. Muchos profesores explican los primeros seis capítulos y luego uno de los tres últimos como “postre”.

Una última nota: agradecemos cualquier comentario

Animamos a los estudiantes y profesores a enviarnos por correo electrónico cualquier comentario que tengan sobre nuestro libro. Ha sido maravilloso para nosotros escuchar opiniones de profesores y estudiantes de todo el mundo sobre las cinco primeras ediciones. Muchas de esas sugerencias se han incorporado en ediciones posteriores del libro. También animamos a los profesores a enviarnos nuevos problemas (y sus soluciones) que complementen los problemas actualmente incluidos, los cuales añadiremos únicamente en la parte de acceso exclusivo para profesores del sitio web. Animamos también a los profesores y estudiantes a crear nuevos applets Java que ilustren los conceptos y protocolos de este libro. Si tiene un applet que piensa que podría ser adecuado para este texto, por favor, envíenoslo. Si el applet (incluyendo la notación y terminología) es apropiado, estaremos encantados de incluirlo en el sitio web del libro, junto con la apropiada referencia a los autores del mismo.

De modo que, como dice la famosa frase, “¡Que no pare la musical!” En serio, *continúen* enviándonos direcciones URL interesantes, señalándonos los errores tipográficos, mostrándose en desacuerdo con cualquiera de nuestras afirmaciones y diciéndonos lo que creen que funciona y lo que no. Díganos qué es lo que piensa que debería o no debería ser incluido en la siguiente edición. Envíenos su correo electrónico a kurose@cs.umass.edu y keithwross@nyu.edu.

Agradecimientos

Desde que comenzamos a escribir este libro en 1996, muchas personas nos han proporcionado una ayuda inestimable y nos ha influido dando forma a nuestras ideas sobre cómo organizar e impartir un curso de redes. MUCHAS GRACIAS a todos los que nos han ayudado desde el primer borrador del libro hasta la séptima edición. Estamos también *muy* agradecidos a los muchos cientos de lectores de todo el mundo (estudiantes, profesores, usuarios) que nos han enviado ideas y comentarios sobre las ediciones anteriores del libro y sugerencias sobre las futuras ediciones del mismo. Gracias especiales a:

Al Aho (Universidad de Columbia)
Hisham Al-Mubaid (Universidad de Houston-Clear Lake)
Pratima Akkunoor (Universidad del Estado de Arizona)
Paul Amer (Universidad de Delaware)
Shamiul Azom (Universidad del Estado de Arizona)
Lichun Bao (Universidad de California en Irvine)
Paul Barford (Universidad de Wisconsin)
Bobby Bhattacharjee (Universidad de Maryland)
Steven Bellovin (Universidad de Columbia)
Pravin Bhagwat (Wibhu)

Supratik Bhattacharyya (anteriormente en Sprint)
Ernst Biersack (Eurécom Institute)
Shahid Bokhari (University of Engineering & Technology, Lahore)
Jean Bolot (Technicolor Research)
Daniel Brushteyn (former University of Pennsylvania student)
Ken Calvert (Universidad de Kentucky)
Evandro Cantu (Universidad Federal de Santa Catarina)
Jeff Case (SNMP Research International)
Jeff Chaltas (Sprint)
Vinton Cerf (Google)
Byung Kyu Choi (Universidad Tecnológica de Michigan)
Bram Cohen (BitTorrent, Inc.)
Constantine Coutras (Pace University)
John Daigle (Universidad de Mississippi)
Edmundo A. de Souza e Silva (Universidad Federal de Río de Janeiro)
Philippe Decuetos (Eurécom Institute)
Christophe Diot (Technicolor Research)
Prithula Dhunghel (Akamai)
Deborah Estrin (Universidad de California, Los Angeles)
Michalis Faloutsos (Universidad de California en Riverside)
Wu-chi Feng (Oregon Graduate Institute)
Sally Floyd (ICIR, Universidad de California en Berkeley)
Paul Francis (Instituto Max Planck)
David Fullager (Netflix)
Lixin Gao (Universidad de Massachusetts)
JJ Garcia-Luna-Aceves (Universidad de California en Santa Cruz)
Mario Gerla (Universidad de California en Los Angeles)
David Goodman (NYU-Poly)
Yang Guo (Alcatel/Lucent Bell Labs)
Tim Griffin (Universidad de Cambridge)
Max Hailperin (Gustavus Adolphus College)
Bruce Harvey (Florida A&M University, Florida State University)
Carl Hauser (Universidad del Estado de Washington)
Rachelle Heller (Universidad George Washington)
Phillipp Hoschka (INRIA/W3C)
Wen Hsin (Park University)
Albert Huang (antiguo estudiante de la Universidad de Pensilvania)
Cheng Huang (Microsoft Research)
Esther A. Hughes (Virginia Commonwealth University)
Van Jacobson (Xerox PARC)
Pinak Jain (former NYU-Poly student)
Jobin James (Universidad de California en Riverside)
Sugih Jamin (Universidad de Michigan)
Shivkumar Kalyanaraman (IBM Research, India)
Jussi Kangasharju (Universidad de Helsinki)
Sneha Kasera (Universidad de Utah)
Parviz Kermani (formerly of IBM Research)
Hyojin Kim (antiguo estudiante de la Universidad de Pensilvania)
Leonard Kleinrock (Universidad de California en Los Ángeles)
David Kotz (Dartmouth College)
Beshan Kulapala (Universidad del Estado de Arizona)
Rakesh Kumar (Bloomberg)

Miguel A. Labrador (University of South Florida)
Simon Lam (Universidad de Texas)
Steve Lai (Universidad del Estado de Ohio)
Tom LaPorta (Penn State University)
Tim-Berners Lee (World Wide Web Consortium)
Arnaud Legout (INRIA)
Lee Leitner (Universidad de Drexel)
Brian Levine (Universidad de Massachusetts)
Chunchun Li (former NYU-Poly student)
Yong Liu (NYU-Poly)
William Liang (antiguo estudiante de la Universidad de Pensilvania)
Willis Marti (Texas A&M University)
Nick McKeown (Universidad de Stanford)
Josh McKinzie (Park University)
Deep Medhi (Universidad de Missouri, Kansas City)
Bob Metcalfe (International Data Group)
Sue Moon (KAIST)
Jenni Moyer (Comcast)
Erich Nahum (IBM Research)
Christos Papadopoulos (Instituto del Estado de Colorado)
Craig Partridge (BBN Technologies)
Radia Perlman (Intel)
Jitendra Padhye (Microsoft Research)
Vern Paxson (University of California at Berkeley)
Kevin Phillips (Sprint)
George Polyzos (Athens University of Economics and Business)
Sriram Rajagopalan (Universidad del Estado de Arizona)
Ramachandran Ramjee (Microsoft Research)
Ken Reek (Instituto de Tecnología Rochester)
Martin Reisslein (Universidad del Estado de Arizona)
Jennifer Rexford (Universidad de Princeton)
Leon Reznik (Instituto de Tecnología Rochester)
Pablo Rodríguez (Telefónica)
Sumit Roy (Universidad de Washington)
Dan Rubenstein (Universidad de Columbia)
Avi Rubin (Universidad Johns Hopkins)
Douglas Salane (John Jay College)
Despina Saporilla (Cisco Systems)
John Schanz (Comcast)
Henning Schulzrinne (Universidad de Columbia)
Mischa Schwartz (Universidad de Columbia)
Ardash Sethi (Universidad de Delaware)
Harish Sethu (Universidad de Drexel)
K. Sam Shanmugan (Universidad de Kansas)
Prashant Shenoy (Universidad de Massachusetts)
Clay Shields (Universidad de Georgetown)
Subin Shrestra (Universidad de Pensilvania)
Bojie Shu (former NYU-Poly student)
Mihail L. Sichitiu (NC State University)
Peter Steenkiste (Universidad de Carnegie Mellon)
Tatsuya Suda (Universidad de California en Irvine)
Kin Sun Tam (Universidad del Estado de Nueva York en Albany)

Don Towsley (Universidad de Massachusetts)
David Turner (Universidad del Estado de California, San Bernardino)
Nitin Vaidya (Universidad de Illinois)
Michele Weigle (Universidad de Clemson)
David Wetherall (Universidad de Washington)
Ira Winston (Universidad de Pensilvania)
Di Wu (Sun Yat-sen University)
Shirley Wynn (NYU-Poly)
Raj Yavatkar (Intel)
Yechiam Yemini (Universidad de Columbia)
Dian Yu (NYU Shanghai)
Ming Yu (Universidad del Estado de Nueva York en Binghamton)
Ellen Zegura (Instituto de Tecnología de Georgia)
Honggang Zhang (Universidad de Suffolk)
Hui Zhang (Universidad de Carnegie Mellon)
Lixia Zhang (Universidad de California en Los Angeles)
Meng Zhang (antiguo estudiante de NYU-Poly)
Shuchun Zhang (antiguo estudiante de la Universidad de Pensilvania)
Xiaodong Zhang (Universidad del Estado de Ohio)
ZhiLi Zhang (Universidad de Minnesota)
Phil Zimmermann (consultor independiente)
Mike Zink (Universidad de Massachusetts)
Cliff C. Zou (Universidad de Central Florida)

También queremos dar las gracias a todo el equipo de Pearson —en particular, a Matt Goldstein y Joanne Manning—que han hecho un trabajo absolutamente excelente en esta séptima edición (y que han sabido llevar a dos autores que parecen congénitamente incapaces de cumplir con los plazos). Gracias también a nuestros artistas, Janet Theurer y Patrice Rossi Calkin, por su trabajo en las bonitas figuras de esta y de las anteriores ediciones del libro, y a Katie Ostler y su equipo de Cenveo por el maravilloso trabajo de producción en esta edición. Por último, una gratitud muy especial a nuestros dos antiguos editores en Addison-Wesley—Michael Hirsch y Susan Hartman. Este libro no sería lo que es (e incluso puede que nunca hubiera llegado a ser) sin su apropiada gestión del proyecto, su apoyo constante, su paciencia casi infinita, su buen humor y su perseverancia.

Contenido

Capítulo 1 Redes de computadoras e Internet.....	1
1.1 ¿Qué es Internet?	2
1.1.1 Descripción de los componentes esenciales.....	2
1.1.2 Descripción de los servicios.....	5
1.1.3 ¿Qué es un protocolo?	6
1.2 La frontera de la red	8
1.2.1 Redes de acceso.....	8
1.2.2 Medios físicos	16
1.3 El núcleo de la red	18
1.3.1 Comutación de paquetes.....	18
1.3.2 Comutación de circuitos.....	23
1.3.3 Una red de redes	27
1.4 Retardos, pérdidas y tasa de transferencia en las redes de comutación de paquetes.....	29
1.4.1 El retardo en las redes de comutación de paquetes	30
1.4.2 Retardo de cola y pérdida de paquetes	33
1.4.3 Retardo extremo a extremo	35
1.4.4 Tasa de transferencia en las redes de computadoras	37
1.5 Capas de protocolos y sus modelos de servicio.....	39
1.5.1 Arquitectura en capas	40
1.5.2 Encapsulación.....	45
1.6 Ataques a las redes	46
1.7 Historia de Internet y de las redes de computadoras	50
1.7.1 El desarrollo de la comutación de paquetes: 1961–1972	50
1.7.2 Redes propietarias e interredes: 1972–1980.....	51
1.7.3 Proliferación de las redes: 1980–1990	52
1.7.4 La explosión de Internet: década de 1990.....	53
1.7.5 El nuevo milenio	54
1.8 Resumen	55
Problemas y cuestiones de repaso	56
Prácticas de laboratorio con Wireshark	65
Entrevista: Leonard Kleinrock	67
Capítulo 2 La capa de aplicación	69
2.1 Principios de las aplicaciones de red	70
2.1.1 Arquitecturas de las aplicaciones de red	70
2.1.2 Comunicación entre procesos.....	73
2.1.3 Servicios de transporte disponibles para las aplicaciones	75
2.1.4 Servicios de transporte proporcionados por Internet.....	77
2.1.5 Protocolos de la capa de aplicación.....	79
2.1.6 Aplicaciones de red analizadas en este libro	81
2.2 La Web y HTTP	81
2.2.1 Introducción a HTTP.....	82

2.2.2	Conexiones persistentes y no persistentes.....	83
2.2.3	Formato de los mensajes HTTP	86
2.2.4	Interacción usuario-servidor: cookies	89
2.2.5	Almacenamiento en caché web	91
2.3	Correo electrónico en Internet	96
2.3.1	SMTP	97
2.3.2	Comparación con HTTP.....	100
2.3.3	Formatos de los mensajes de correo.....	100
2.3.4	Protocolos de acceso para correo electrónico.....	101
2.4	DNS: el servicio de directorio de Internet.....	104
2.4.1	Servicios proporcionados por DNS	105
2.4.2	Cómo funciona DNS	106
2.4.3	Registros y mensajes DNS	112
2.5	Distribución de archivos P2P	116
2.6	Flujos de vídeo y redes de distribución de contenido.....	121
2.6.1	Vídeo por Internet	121
2.6.2	Flujos de vídeo HTTP y tecnología DASH.....	122
2.6.3	Redes de distribución de contenido.....	123
2.6.4	Casos de estudio: Netflix, YouTube y Kankan	127
2.7	Programación de sockets: creación de aplicaciones de red	130
2.7.1	Programación de sockets con UDP	131
2.7.2	Programación de sockets con TCP	135
2.8	Resumen	140
	Problemas y cuestiones de repaso	140
	Tareas sobre programación de sockets	148
	Prácticas de laboratorio con Wireshark: HTTP	149
	Prácticas de laboratorio con Wireshark: DNS	150
	Entrevista: Marc Andreessen.....	151

Capítulo 3 La capa de transporte.....153

3.1	La capa de transporte y sus servicios	154
3.1.1	Relaciones entre las capas de transporte y de red	154
3.1.2	La capa de transporte en Internet	156
3.2	Multiplexación y demultiplexación	158
3.3	Transporte sin conexión: UDP	164
3.3.1	Estructura de los segmentos UDP	167
3.3.2	Suma de comprobación de UDP	167
3.4	Principios de un servicio de transferencia de datos fiable.....	169
3.4.1	Construcción de un protocolo de transferencia de datos fiable.....	170
3.4.2	Protocolo de transferencia de datos fiable con procesamiento en cadena.....	178
3.4.3	Retroceder N (GBN)	181
3.4.4	Repetición selectiva (SR)	186
3.5	Transporte orientado a la conexión: TCP	191
3.5.1	La conexión TCP.....	191
3.5.2	Estructura del segmento TCP	193
3.5.3	Estimación del tiempo de ida y vuelta y fin de temporización	198
3.5.4	Transferencia de datos fiable.....	200
3.5.5	Control de flujo	207
3.5.6	Gestión de la conexión TCP	209
3.6	Principios del control de congestión.....	214
3.6.1	Las causas y los costes de la congestión	214

3.6.2	Métodos para controlar la congestión	220
3.7	Mecanismo de control de congestión de TCP	221
3.7.1	Equidad.....	230
3.7.2	Notificación explícita de congestión (ECN): control de congestión asistido por la red	232
3.8	Resumen	233
	Problemas y cuestiones de repaso	236
	Tareas de programación	248
	Prácticas de laboratorio con Wireshark: exploración de TCP, UDP.....	249
	Entrevista: Van Jacobson	250
Capítulo 4	La capa de red: el plano de datos	253
4.1	Introducción a la capa de red.....	254
4.1.1	Reenvío y enrutamiento: los planos de datos y de control.....	254
4.1.2	Modelo de servicio de red	258
4.2	El interior de un router	259
4.2.1	Procesamiento en el puerto de entrada y reenvío basado en el destino.....	262
4.2.2	Comunicación	264
4.2.3	Procesamiento en el puerto de salida	266
4.2.4	¿Dónde se crean colas?.....	266
4.2.5	Planificación de paquetes	270
4.3	Protocolo de Internet (IP): IPv4, direccionamiento, IPv6 y más.....	274
4.3.1	Formato de los datagramas IPv4	274
4.3.2	Fragmentación del datagrama IPv4.....	276
4.3.3	Direccionamiento IPv4.....	277
4.3.4	Traducción de direcciones de red (NAT).....	286
4.3.5	IPv6	289
4.4	Reenvío generalizado y SDN	294
4.4.1	Correspondencia.....	296
4.4.2	Acción	297
4.4.3	Ejemplos de correspondencia-acción en OpenFlow	297
4.5	Resumen	299
	Problemas y cuestiones de repaso	300
	Práctica de laboratorio con Wireshark	306
	Entrevista: Vinton G. Cerf.....	307
Capítulo 5	La capa de red: el plano de control	309
5.1	Introducción.....	309
5.2	Algoritmos de enrutamiento	311
5.2.1	Algoritmo de enrutamiento de estado de enlaces (LS).....	314
5.2.2	Algoritmo de enrutamiento por vector de distancias (DV)	317
5.3	Enrutamiento dentro de un sistema autónomo en Internet: OSPF	323
5.4	Enrutamiento entre los ISP: BGP	326
5.4.1	El papel de BGP	327
5.4.2	Anuncio de la información de rutas BGP.....	327
5.4.3	Determinación de las mejores rutas.....	329
5.4.4	IP-Anycast.....	332
5.4.5	Política de enrutamiento	333
5.4.6	Cómo encajan las piezas: obtención de presencia en Internet.....	335
5.5	El plano de control SDN.....	336

5.5.1	El plano de control SDN: controlador SDN y aplicaciones SDN de control de red	338
5.5.2	Protocolo OpenFlow	340
5.5.3	Interacción entre los planos de datos y de control: ejemplo.....	342
5.5.4	SDN: pasado y futuro	343
5.6	Protocolo de mensajes de control de Internet (ICMP)	346
5.7	Gestión de red y SNMP	348
5.7.1	El marco conceptual de la gestión de red.....	348
5.7.2	El protocolo SNMP	350
5.8	Resumen	352
	Problemas y cuestiones de repaso	353
	Tarea de programación con socket	358
	Tarea de programación	358
	Prácticas de laboratorio con Wireshark	359
	Entrevista: Jennifer Rexford.....	360
Capítulo 6	La capa de enlace y las redes de área local.....	363
6.1	Introducción a la capa de enlace.....	364
6.1.1	Servicios proporcionados por la capa de enlace.....	364
6.1.2	¿Dónde se implementa la capa de enlace?	366
6.2	Técnicas de detección y corrección de errores	367
6.2.1	Comprobaciones de paridad	369
6.2.2	Métodos basados en la suma de comprobación.....	370
6.2.3	Comprobación de redundancia cíclica (CRC).....	371
6.3	Protocolos y enlaces de acceso múltiple	373
6.3.1	Protocolos de particionamiento del canal.....	375
6.3.2	Protocolos de acceso aleatorio	377
6.3.3	Protocolos de toma de turnos	384
6.3.4	DOCSIS: el protocolo de la capa de enlace para acceso a Internet por cable.....	385
6.4	Redes de área local commutadas	386
6.4.1	Direccionamiento de la capa de enlace y ARP.....	387
6.4.2	Ethernet	393
6.4.3	Switches de la capa de enlace.....	398
6.4.4	Redes de área local virtuales (VLAN)	403
6.5	Virtualización de enlaces: la red como una capa de enlace.....	406
6.5.1	Commutación de etiquetas multiprotocolo (MPLS)	407
6.6	Redes para centros de datos.....	409
6.7	Retrospectiva: un día en la vida de una solicitud de página web	413
6.7.1	Inicio: DHCP, UDP, IP y Ethernet.....	414
6.7.2	Seguimos con el inicio: DNS y ARP	415
6.7.3	Seguimos con el inicio: enrutamiento dentro del dominio al servidor DNS.....	416
6.7.4	Interacción web cliente-servidor: TCP y HTTP	417
6.8	Resumen	418
	Problemas y cuestiones de repaso	419
	Prácticas de laboratorio con Wireshark	426
	Entrevista: Simon S. Lam.....	427
Capítulo 7	Redes inalámbricas y móviles	429
7.1	Introducción.....	430
7.2	Características de las redes y enlaces inalámbricos	434
7.2.1	CDMA	437

7.3	WiFi: redes LAN inalámbricas 802.11.....	440
7.3.1	La arquitectura 802.11.....	440
7.3.2	El protocolo MAC 802.11	444
7.3.3	La trama IEEE 802.11	449
7.3.4	Movilidad dentro de la misma subred IP.....	452
7.3.5	Características avanzadas de 802.11	453
7.3.6	Redes de área personal: Bluetooth y Zigbee	454
7.4	Acceso celular a Internet	456
7.4.1	Panorámica de la arquitectura de las redes celulares	457
7.4.2	Redes de datos celulares 3G: llevando Internet a los abonados celulares.....	459
7.4.3	Hacia la tecnología 4G: LTE.....	461
7.5	Gestión de la movilidad: principios.....	464
7.5.1	Direccionamiento	467
7.5.2	Enrutamiento hacia un nodo móvil	468
7.6	IP móvil	472
7.7	Gestión de la movilidad en redes celulares	476
7.7.1	Enrutamiento de llamadas hacia un usuario móvil.....	477
7.7.2	Transferencia de llamadas en GSM.....	478
7.8	Tecnología inalámbrica y movilidad: impacto sobre los protocolos de las capas superiores ...	481
7.9	Resumen	483
	Problemas y cuestiones de repaso	483
	Prácticas de laboratorio con Wireshark	487
	Entrevista: Deborah Estrin	488

Capítulo 8 Seguridad en las redes de computadoras.....491

8.1	¿Qué es la seguridad de red?	492
8.2	Principios de la criptografía.....	494
8.2.1	Criptografía de clave simétrica.....	495
8.2.2	Cifrado de clave pública.....	500
8.3	Integridad de los mensajes y firmas digitales.....	505
8.3.1	Funciones hash criptográficas	506
8.3.2	Código de autenticación del mensaje	507
8.3.3	Firmas digitales	509
8.4	Autenticación del punto terminal	514
8.4.1	Protocolo de autenticación <i>ap1.0</i>	515
8.4.2	Protocolo de autenticación <i>ap2.0</i>	515
8.4.3	Protocolo de autenticación <i>ap3.0</i>	516
8.4.4	Protocolo de autenticación <i>ap3.1</i>	516
8.4.5	Protocolo de autenticación <i>ap4.0</i>	517
8.5	Asegurando el correo electrónico	518
8.5.1	Correo electrónico seguro	519
8.5.2	PGP	522
8.6	Asegurando las conexiones TCP: SSL	523
8.6.1	Panorámica general	524
8.6.2	Una panorámica más completa.....	526
8.7	Seguridad de la capa de red: IPsec y redes privadas virtuales.....	528
8.7.1	IPsec y redes privadas virtuales (VPN).....	529
8.7.2	Los protocolos AH y ESP	530
8.7.3	Asociaciones de seguridad	530
8.7.4	El datagrama IPsec	531

8.7.5	IKE: gestión de claves en IPsec	534
8.8	Asegurando las redes LAN inalámbricas	535
8.8.1	WEP (Wired Equivalent Privacy)	535
8.8.2	IEEE 802.11i	537
8.9	Seguridad operacional: cortafuegos y sistemas de detección de intrusiones	539
8.9.1	Cortafuegos	539
8.9.2	Sistemas de detección de intrusiones	546
8.10	Resumen	549
	Problemas y cuestiones de repaso	550
	Prácticas de laboratorio con Wireshark	557
	Prácticas de laboratorio con IPsec	557
	Entrevista: Steven M. Bellovin	558
Capítulo 9	Redes multimedia.....	561
9.1	Aplicaciones multimedia en red	562
9.1.1	Propiedades del vídeo.....	562
9.1.2	Propiedades del audio.....	563
9.1.3	Tipos de aplicaciones multimedia en red	564
9.2	Flujos de vídeo almacenado	566
9.2.1	Flujos UDP	567
9.2.2	Flujos HTTP	568
9.3	Voz sobre IP	572
9.3.1	Limitaciones del servicio IP de entrega de mejor esfuerzo.....	572
9.3.2	Eliminación de las fluctuaciones al reproducir audio en el receptor.....	574
9.3.3	Recuperación frente a pérdidas de paquetes.....	577
9.3.4	Caso de estudio: VoIP con Skype	579
9.4	Protocolos para aplicaciones de conversación en tiempo real.....	582
9.4.1	RTP.....	582
9.4.2	SIP	585
9.5	Soporte de red para aplicaciones multimedia	589
9.5.1	Dimensionamiento de las redes con servicio de entrega de mejor esfuerzo	591
9.5.2	Provisión de múltiples clases de servicio	592
9.5.3	Diffserv	598
9.5.4	Garantías de calidad de servicio (QoS) por conexión: reserva de recursos y admisión de llamadas	601
9.6	Resumen	604
	Problemas y cuestiones de repaso	605
	Tarea de programación	611
	Entrevista: Henning Schulzrinne	613
	Referencias	615
	Índice	645

Redes de computadoras e Internet

Hoy día, Internet es casi indiscutiblemente el sistema de ingeniería más grande creado por la mano del hombre, con cientos de millones de computadoras conectadas, enlaces de comunicaciones y switches; con miles de millones de usuarios que se conectan a través de computadoras portátiles, tabletas y teléfonos inteligentes; y con una amplia variedad de nuevas “cosas” conectadas a Internet, incluyendo consolas de juegos, sistemas de vigilancia, relojes, gafas, termostatos, básculas y vehículos. Dado que Internet es una red tan enorme e incluye tantos componentes distintos y tiene tantos usos, ¿es posible tener la esperanza de comprender cómo funciona? ¿Existen unos principios y una estructura básicos que puedan proporcionar los fundamentos para comprender un sistema tan asombrosamente complejo y grande? Y, en caso afirmativo, ¿es posible que pueda resultar interesante y divertido aprender acerca de las redes de computadoras? Afortunadamente, la respuesta a todas estas preguntas es un rotundo ¡SÍ! De hecho, el objetivo de este libro es proporcionar al lector una introducción moderna al dinámico campo de las redes de computadoras, exponiendo los principios y los conocimientos prácticos que necesitará para entender no solo las redes actuales, sino también las del futuro.

En el primer capítulo se hace una amplia introducción al mundo de las redes de computadoras y de Internet. Nuestro objetivo es proporcionar una visión general y establecer el contexto para el resto del libro, con el fin de poder ver el bosque a través de los árboles. En este capítulo de introducción se abordan muchos de los fundamentos, así como muchos de los componentes que forman una red de computadoras, siempre sin perder de vista la panorámica general.

Vamos a estructurar esta introducción a las redes de computadoras de la siguiente forma: después de exponer algunos conceptos y términos fundamentales, examinaremos los componentes hardware y software esenciales que forman una red de computadoras. Comenzaremos por la frontera de la red y echaremos un vistazo a los sistemas terminales y aplicaciones que se ejecutan en la red. A continuación, exploraremos el núcleo de una red de computadoras, examinando los enlaces y los switches que transportan los datos, así como las redes de acceso y los medios físicos que conectan los sistemas terminales con el núcleo de la red. Aprenderemos que Internet es una red de redes y cómo estas redes se conectan entre sí.

Una vez completada la introducción sobre la frontera y el núcleo de una red de computadoras, en la segunda mitad del capítulo adoptaremos un punto de vista más amplio y abstracto. Examinaremos los retardos, las pérdidas y la tasa de transferencia de datos en una red de computadoras y proporcionaremos modelos cuantitativos simples para los retardos y tasas de transferencia de terminal a terminal: modelos que tienen en cuenta los retardos de transmisión, de propagación y de cola. A continuación, presentaremos algunos de los principios básicos sobre las arquitecturas de red: en concreto, las capas de protocolos y los modelos de servicios. También veremos que las redes son vulnerables a muchos tipos distintos de ataques; revisaremos algunos de estos ataques y veremos cómo es posible conseguir que las redes sean más seguras. Por último, concluiremos el capítulo con una breve historia de las redes de comunicaciones.

1.1 ¿Qué es Internet?

En este libro, vamos a emplear la red pública Internet, una red de computadoras específica, como nuestro principal vehículo para explicar las redes de computadoras y sus protocolos. Pero, ¿qué es Internet? Hay dos formas de responder a esta pregunta. La primera de ellas es describiendo las tuercas y tornillos que forman la red, es decir, los componentes hardware y software básicos que forman Internet. La segunda es describiéndola en términos de la infraestructura de red que proporciona servicios a aplicaciones distribuidas. Comenzaremos por la descripción de los componentes esenciales, utilizando la Figura 1.1 para ilustrar la exposición.

1.1.1 Descripción de los componentes esenciales

Internet es una red de computadoras que interconecta miles de millones de dispositivos informáticos a lo largo de todo el mundo. No hace demasiado tiempo, estos dispositivos eran fundamentalmente computadoras PC de escritorio tradicionales, estaciones de trabajo Linux y los llamados servidores, que almacenan y transmiten información tal como páginas web y mensajes de correo electrónico. Sin embargo, cada vez se conectan a Internet más “cosas” (dispositivos) no tradicionales, como computadoras portátiles, teléfonos inteligentes, tabletas, televisiones, consolas de juegos, termóstatos, sistemas domésticos de alarma, electrodomésticos, relojes, gafas, vehículos, sistemas de control de tráfico y otros. De hecho, el término *red de computadoras* está comenzando a sonar algo obsoleto, a causa de la gran cantidad de dispositivos no tradicionales que se conectan a Internet. En la jerga de Internet, todos estos dispositivos se denominan **hosts** o **sistemas terminales**. Según algunas estimaciones, en 2015 había unos 5.000 millones de dispositivos conectados a Internet, y esa cifra alcanzará los 25.000 millones en 2020 [Gartner 2014]. Se estima que en 2015 había más de 3.200 millones de usuarios de Internet en todo el mundo, aproximadamente el 40% de la población mundial [ITU 2015].

Los sistemas terminales se conectan entre sí mediante una red de **enlaces de comunicaciones y comutadores de paquetes**. En la Sección 1.2 veremos que existen muchos tipos de enlaces de comunicaciones, los cuales están compuestos por diferentes tipos de medios físicos, entre los que se incluyen el cable coaxial, el hilo de cobre, la fibra óptica y el espectro de radio. Los distintos enlaces pueden transmitir los datos a distintas velocidades y la **velocidad de transmisión** de un enlace se mide en bits/segundo. Cuando un sistema terminal tiene datos que enviar a otro sistema terminal, el emisor segmenta los datos y añade bytes de cabecera a cada segmento. Los paquetes de información resultantes, conocidos como **paquetes** en la jerga informática, se envían entonces a través de la red hasta el sistema terminal receptor, donde vuelven a ser ensamblados para obtener los datos originales.

Un comutador de paquetes toma un paquete que llega a través de uno de sus enlaces de comunicaciones de entrada y lo reenvía a través de uno de sus enlaces de comunicaciones de salida. Los comutadores de paquetes se suministran en muchas formas y modelos, pero los dos

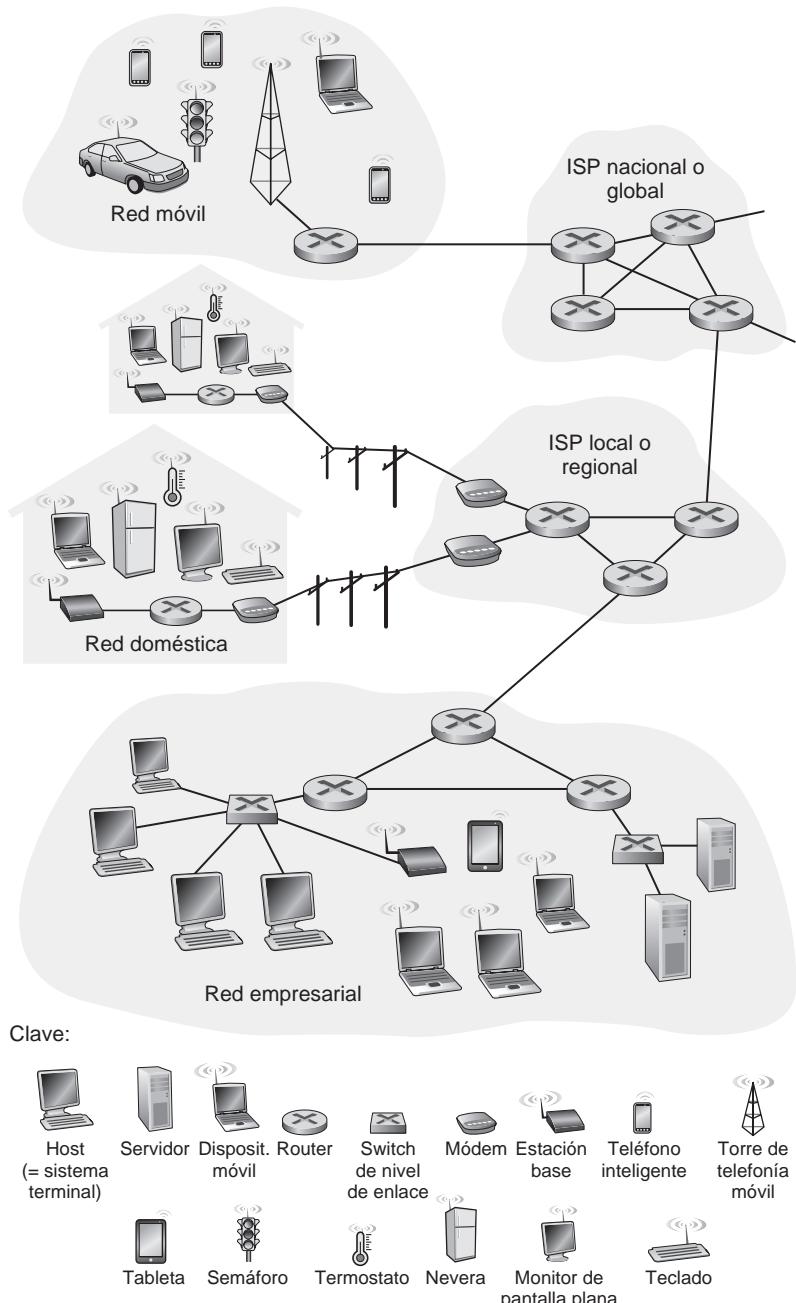


Figura 1.1 ♦ Algunos de los componentes de Internet.

tipos más utilizados actualmente en Internet son los **routers** y los **switches de la capa de enlace**. Ambos tipos de conmutadores reenvían paquetes hacia sus destinos finales. Los switches de la capa de enlace normalmente se emplean en las redes de acceso, mientras que los routers suelen utilizarse en el núcleo de la red. La secuencia de enlaces de comunicaciones y conmutadores de paquetes que atraviesa un paquete desde el sistema terminal emisor hasta el sistema terminal receptor, se conoce con el nombre de **ruta** a través de la red. Según las predicciones de Cisco, el tráfico IP global anual sobrepasará el umbral del zettabyte (10^{21} bytes) a finales de 2016 y alcanzará los 2 zettabytes por año en 2019 [Cisco VNI 2015].

Las redes de conmutación de paquetes (que transportan paquetes) son similares en muchos aspectos a las redes de transporte formadas por autopistas, carreteras e intersecciones (que transportan vehículos). Por ejemplo, imagine que una fábrica necesita trasladar un enorme cargamento a un cierto almacén de destino que se encuentra a miles de kilómetros. En la fábrica, el cargamento se reparte y se carga en una flota de camiones. Cada camión hace el viaje hasta el almacén de destino de forma independiente a través de la red de autopistas, carreteras e intersecciones. En el almacén de destino, la carga de cada camión se descarga y se agrupa con el resto del cargamento correspondiente al mismo envío. Así, en cierto sentido, los paquetes son como los camiones, los enlaces de comunicaciones como las autopistas y carreteras, los dispositivos de conmutación de paquetes como las intersecciones y los sistemas terminales son como los edificios (la fábrica y el almacén). Al igual que un camión sigue una ruta a través de la red de transporte por carretera, un paquete sigue una ruta a través de una red de computadoras.

Los sistemas terminales acceden a Internet a través de los **ISP (Internet Service Provider, Proveedor de servicios de Internet)**, incluyendo los ISP residenciales, como son las compañías telefónicas o de cable locales; los ISP corporativos; los ISP universitarios; los ISP que proporcionan acceso inalámbrico (WiFi) en aeropuertos, hoteles, cafeterías y otros lugares públicos; y los ISP de datos móviles, que proporcionan acceso móvil a nuestros teléfonos inteligentes y otros dispositivos. Cada ISP es en sí mismo una red de conmutadores de paquetes y enlaces de comunicaciones. Los ISP proporcionan una amplia variedad de tipos de acceso a red a los sistemas terminales, entre los que se incluyen el acceso de banda ancha residencial, mediante módem por cable o DSL; el acceso LAN (*Local Area Network*, Red de área local) de alta velocidad y el acceso inalámbrico para dispositivos móviles. Los ISP también proporcionan acceso a Internet a los proveedores de contenido, conectando sitios web y servidores de vídeo directamente a Internet. El objetivo de Internet no es otro que conectar los sistemas terminales entre sí, por lo que los ISP que proporcionan el acceso a los sistemas terminales también tienen que estar interconectados entre ellos. Estos ISP de nivel inferior se interconectan a través de ISP de nivel superior nacionales e internacionales, como Level 3 Communications, AT&T, Sprint y NTT. Un ISP de nivel superior está compuesto por routers de alta velocidad interconectados a través de enlaces de fibra óptica de alta velocidad. La red de cada ISP, sea de nivel inferior o superior, se administra de forma independiente, ejecuta el protocolo IP (véase más adelante) y se ajusta a determinados convenios de denominación y de asignación de direcciones. En la Sección 1.3 examinaremos más detalladamente los ISP y sus interconexiones.

Los sistemas terminales, los conmutadores de paquetes y otros dispositivos de Internet ejecutan **protocolos** que controlan el envío y la recepción de información dentro de Internet. El protocolo **TCP (Transmission Control Protocol, Protocolo de control de transmisión)** y el protocolo **IP (Internet Protocol, Protocolo Internet)** son dos de los protocolos más importantes de Internet. El protocolo IP especifica el formato de los paquetes que se envían y reciben entre los routers y los sistemas terminales. Los principales protocolos de Internet se conocen colectivamente como protocolos **TCP/IP**. En este capítulo de introducción comenzaremos a estudiar los protocolos, pero esto solo es el principio, ya que gran parte del libro se dedica a los protocolos empleados por las redes de computadoras.

Debido a la importancia de los protocolos en Internet, es importante que todo el mundo esté de acuerdo en qué hacen todos y cada uno de ellos, para que la gente pueda crear sistemas y productos capaces de interoperar. Aquí es donde entran en juego los estándares. Los **estándares de Internet** son desarrollados por el IETF (Internet Engineering Task Force, Grupo de trabajo de ingeniería de Internet) [IETF 2016]. Los documentos asociados a estos estándares IETF se conocen como **documentos RFC (Request For Comments, Solicitud de comentarios)**. Los RFC nacieron como solicitudes de comentarios de carácter general (de ahí su nombre) para solucionar los problemas de diseño de la red y de los protocolos a los que se enfrentó el precursor de Internet [Allman 2011]. El contenido de estos documentos suele ser bastante técnico y detallado. Definen protocolos tales como **TCP, IP, HTTP (para la Web) y SMTP (para el correo electrónico)**. Actualmente, existen más de 7.000 documentos RFC. Existen también otros organismos dedicados a especificar estándares para componentes de red, especialmente para los enlaces de red. El comité de estándares IEEE 802 LAN/

MAN [IEEE 802 2016], por ejemplo, especifica los estándares para redes Ethernet y comunicación WiFi inalámbrica.

1.1.2 Descripción de los servicios

Hasta el momento hemos identificado muchos de los componentes que forman Internet, pero también podemos describir Internet desde un punto de vista completamente diferente, en concreto como *una infraestructura que proporciona servicios a las aplicaciones*. Además de aplicaciones tradicionales como el correo electrónico o la navegación web, las aplicaciones Internet abarcan aplicaciones para tabletas y teléfonos móviles inteligentes, incluyendo la mensajería Internet, mapas con información de tráfico en tiempo real, reproducción de música desde la nube, reproducción de películas y programas de televisión a través de Internet, redes sociales en línea, videoconferencia, juegos multipersona y sistemas de recomendación basados en la ubicación. Se dice que estas aplicaciones son **aplicaciones distribuidas**, porque implican a varios sistemas terminales que intercambian datos entre sí. Es importante saber que las aplicaciones de Internet se ejecutan en los sistemas terminales, no en los comutadores de paquetes que forman el núcleo de la red. Aunque los dispositivos de conmutación de paquetes facilitan el intercambio de datos entre sistemas terminales, no se preocupan de la aplicación que esté actuando como origen o destino de los datos.

Vamos a ahondar un poco más en lo que queremos decir al hablar de una infraestructura que proporciona servicios a las aplicaciones. Para ello, supongamos que tenemos una idea nueva y excitante acerca de una aplicación distribuida de Internet, que puede beneficiar enormemente a la humanidad o que simplemente puede hacernos ricos y famosos. ¿Cómo podríamos transformar esa idea en una aplicación real de Internet? Puesto que las aplicaciones se ejecutan en los sistemas terminales, tendremos que escribir programas software que se ejecuten en dichos sistemas. Por ejemplo, podríamos escribir programas en Java, C o Python. Ahora bien, dado que estamos desarrollando una aplicación Internet distribuida, los programas que se ejecuten en los distintos sistemas terminales tendrán que enviarse datos entre sí. Y aquí es cuando llegamos al meollo de la cuestión, que nos lleva a la forma alternativa de describir Internet como una plataforma para aplicaciones. ¿Cómo hace un programa que se ejecuta en un sistema terminal para ordenar a Internet que entregue datos a otro programa que se ejecuta en otro sistema terminal?

Los sistemas terminales conectados a Internet proporcionan una **interfaz de sockets** que especifica cómo un programa software, que se ejecuta en un sistema terminal, pide a la infraestructura de Internet que suministre datos a un programa de destino específico que se está ejecutando en otro sistema terminal. Esta interfaz de sockets de Internet es un conjunto de reglas que el programa que transmite los datos debe cumplir, para que Internet pueda entregar esos datos al programa de destino. En el Capítulo 2 se aborda en detalle la interfaz de sockets de Internet. Por el momento, veamos una sencilla analogía, una que emplearemos con frecuencia a lo largo de este libro. Supongamos que Alicia desea enviar una carta a Benito utilizando el servicio postal. Por supuesto, Alicia no puede escribir la carta (los datos) y lanzarla por la ventana. En lugar de ello, el servicio postal exige que Alicia introduzca la carta en un sobre, escriba el nombre completo de Benito, su dirección y código postal en el sobre, lo cierre y pegue un sello en la esquina superior derecha del sobre. Por último, tendrá que introducir el sobre en un buzón oficial del servicio postal. Por tanto, el servicio postal de correos tiene su propia “interfaz de servicio postal”, es decir, su propio conjunto de reglas que Alicia debe seguir para que el servicio postal entregue su carta a Benito. De forma similar, Internet tiene una interfaz de sockets que el programa que envía los datos debe seguir, para que Internet entregue los datos al programa que debe recibirlas.

Por supuesto, el servicio postal proporciona más de un servicio a sus clientes, como correo urgente, acuse de recibo, correo ordinario y otros muchos. Del mismo modo, Internet proporciona múltiples servicios a sus aplicaciones. Cuando desarrolle una aplicación de Internet, también tendrá que seleccionar para su aplicación uno de los servicios de Internet. En el Capítulo 2 describiremos esos servicios.

Acabamos de proporcionar dos descripciones de Internet: una en términos de sus componentes hardware y software, y otra como infraestructura que proporciona servicios a aplicaciones distribuidas. Pero es posible que el lector no tenga claro todavía qué es Internet. ¿Qué son la conmutación de paquetes y TCP/IP? ¿Qué son los routers? ¿Qué tipos de enlaces de comunicaciones existen en Internet? ¿Qué es una aplicación distribuida? ¿Cómo puede conectarse a Internet un termostato o una báscula? Si se siente un poco abrumado ahora por todas estas preguntas, no se preocupe: el propósito de este libro es presentarle tanto los componentes esenciales de Internet, como los principios que regulan cómo y por qué funciona. En las siguientes secciones y capítulos explicaremos todos estos términos y daremos respuesta a estas cuestiones.

1.1.3 ¿Qué es un protocolo?

Ahora que ya hemos visto por encima qué es Internet, vamos a ocuparnos de otro término importante en el mundo de las redes de computadoras: *protocolo*. ¿Qué es un protocolo? ¿Qué hace un protocolo?

Analogía humana

Probablemente, sea más sencillo comprender el concepto de protocolo de red considerando en primer lugar algunas analogías humanas, ya que las personas utilizamos protocolos casi constantemente. Piense en lo que hace cuando necesita preguntar a alguien qué hora es. En la Figura 1.2 se muestra cómo se lleva a cabo un intercambio de este tipo. El protocolo entre personas (o las buenas maneras, al menos) dicta que para iniciar un proceso de comunicación con alguien lo primero es saludar (el primer “Hola” mostrado en la Figura 1.2). La respuesta típica a este saludo será también “Hola”. Implícitamente, el saludo cordial de respuesta se toma como una indicación de que se puede continuar

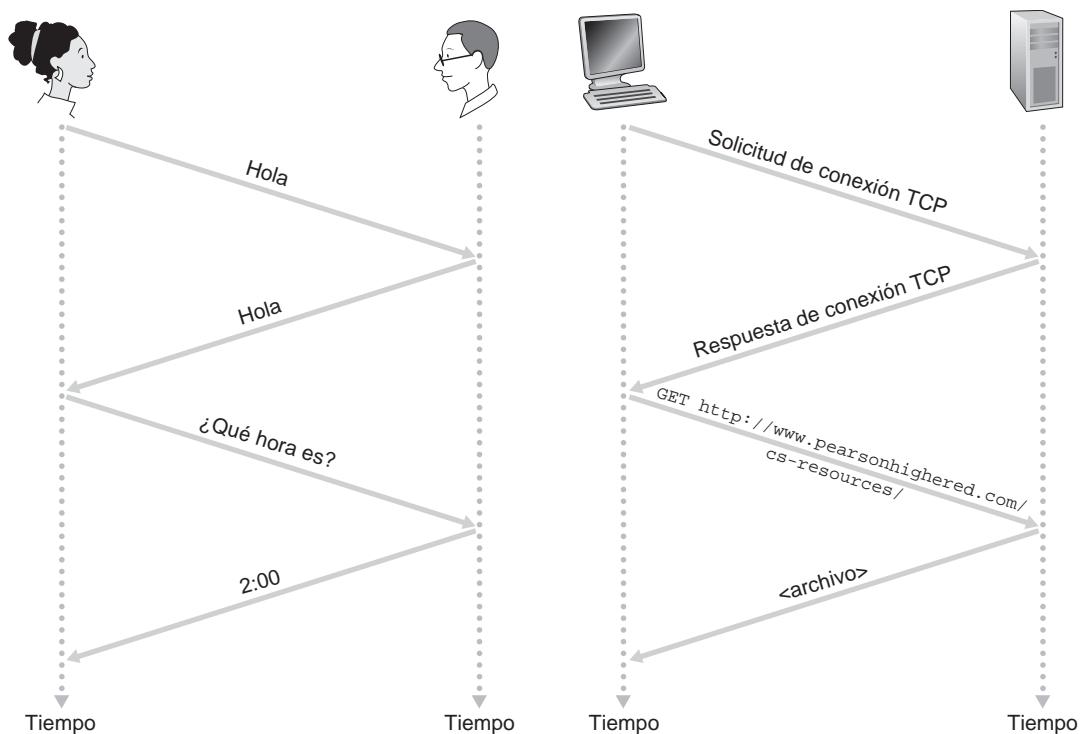


Figura 1.2 ♦ Un protocolo humano y un protocolo de red.

con el proceso de comunicación y preguntar la hora. Una respuesta diferente al “Hola” inicial (como por ejemplo, “¡No me moleste!” o “No hablo su idioma”, o cualquier respuesta impublicable), podría indicar una falta de disposición o una incapacidad para comunicarse. En este caso, el protocolo de las relaciones entre personas establece que no debe preguntarse la hora. En ocasiones, no se obtiene ninguna respuesta, en cuyo caso normalmente renunciamos a preguntar a esa persona la hora que es. Observe que, en el protocolo entre personas, *existen mensajes específicos que enviamos y acciones específicas que tomamos como respuesta a los mensajes de contestación recibidos o a otros sucesos* (como, por ejemplo, no recibir una respuesta en un periodo de tiempo determinado). Evidentemente, los mensajes transmitidos y recibidos, y las acciones tomadas al enviar o recibir estos mensajes o al producirse otros sucesos, desempeñan un papel principal en el protocolo humano. Si las personas adoptan protocolos diferentes (por ejemplo, si una persona guarda las formas pero la otra no lo hace, o si uno comprende el concepto de tiempo y el otro no), los protocolos no interoperarán y no podrá llevarse a cabo ninguna tarea útil. Esta misma idea también es aplicable a las redes: *hacen falta dos (o más) entidades comunicándose y ejecutando el mismo protocolo para poder llevar a cabo una tarea.*

Consideremos ahora una segunda analogía humana. Suponga que está asistiendo a una clase en la universidad (¡por ejemplo, sobre redes!). El profesor está hablando acerca de los protocolos y usted se siente confuso. El profesor detiene su explicación y dice: “¿Alguna pregunta?” (un mensaje que se transmite a todos los estudiantes que no estén dormidos, y que todos ellos reciben). Usted levanta la mano (transmitiendo un mensaje implícito al profesor). El profesor le dirige una sonrisa y le dice “¿Si . . . ?” (un mensaje transmitido que le anima a plantear su pregunta, ya que los profesores *adoran* que les planteen cuestiones) y, a continuación, usted hace la pregunta (es decir, transmite su mensaje al profesor). El profesor escucha la pregunta (recibe su mensaje) y le responde (le transmite una respuesta). De nuevo, vemos que la transmisión y la recepción de mensajes y el conjunto de acciones convencionales tomadas cuando se envían y reciben estos mensajes, constituyen el núcleo de este protocolo de pregunta-respuesta.

Protocolos de red

Un protocolo de red es similar a un protocolo humano, excepto en que las entidades que intercambian mensajes y llevan a cabo las acciones son los componentes hardware o software de cierto dispositivo (por ejemplo, una computadora, un teléfono inteligente, una tableta, un router u otro dispositivo de red). *Cualquier actividad de Internet que implique dos o más entidades remotas que se comunican está gobernada por un protocolo.* Por ejemplo, los protocolos implementados por hardware en las tarjetas de interfaz de red de dos computadoras conectadas físicamente controlan el flujo de bits a través del “cable” conectado entre las dos tarjetas de interfaz de red; los protocolos de control de congestión de los sistemas terminales controlan la velocidad a la que se transmiten los paquetes entre el emisor y el receptor; los protocolos de los routers determinan la ruta que seguirá un paquete desde el origen al destino. Los protocolos se ejecutan por todas partes en Internet y, en consecuencia, gran parte de este libro está dedicada a los protocolos de redes de computadoras.

Como ejemplo de un protocolo de red con el que probablemente estará familiarizado, vamos a ver lo que ocurre cuando se hace una solicitud a un servidor web, es decir, cuando usted escribe el URL de una página web en un navegador. Este escenario se ilustra en la mitad derecha de la Figura 1.2. En primer lugar, su computadora enviará un mensaje de solicitud de conexión al servidor web y esperará una respuesta. El servidor web recibirá su mensaje de solicitud de conexión y le devolverá un mensaje de respuesta de conexión. Sabiendo ahora que es posible solicitar el documento web, su computadora envía el nombre de la página web que desea extraer del servidor web, mediante un mensaje GET. Por último, el servidor web envía la página web (archivo) a su computadora.

Basándonos en los ejemplos anteriores de protocolos humanos y de red, el intercambio de mensajes y las acciones tomadas cuando se envían y reciben estos mensajes constituyen los elementos claves para la definición de un protocolo:

Un protocolo define el formato y el orden de los mensajes intercambiados entre dos o más entidades que se comunican, así como las acciones tomadas al producirse la transmisión y/o recepción de un mensaje u otro suceso.

Internet, y las redes de computadoras en general, hacen un uso extensivo de los protocolos. Se utilizan diferentes protocolos para llevar a cabo las distintas tareas de comunicación. A medida que avance en el libro, verá que algunos protocolos son simples y directos, mientras que otros son complejos e intelectualmente profundos. Dominar el campo de las redes de computadoras es equivalente a entender el qué, el por qué y el cómo de los protocolos de red.

1.2 La frontera de la red

En la sección anterior hemos presentado una introducción de carácter general sobre Internet y los protocolos de red. Ahora vamos a profundizar un poco más en los componentes de una red de computadoras (y de Internet, en concreto). Comenzaremos la sección en la frontera de una red y nos fijaremos en los componentes con los que estamos más familiarizados, es decir, las computadoras, los teléfonos inteligentes y otros dispositivos que utilizamos a diario. En la siguiente sección nos desplazaremos desde la frontera de la red hasta el núcleo de la misma y examinaremos los procesos de conmutación y enrutamiento que tienen lugar en las redes.

Recuerde de la sección anterior que en la jerga de las redes informáticas, las computadoras y el resto de los dispositivos conectados a Internet a menudo se designan como sistemas terminales, porque se sitúan en la frontera de Internet, como se muestra en la Figura 1.3. Entre los sistemas terminales de Internet se incluyen las computadoras de escritorio (por ejemplo, PCs de escritorio, computadoras Mac y equipos Linux), servidores (por ejemplo, servidores web y de correo electrónico) y dispositivos móviles (por ejemplo, computadoras portátiles, teléfonos inteligentes y tabletas). Además, cada vez se conectan más “cosas” no tradicionales a Internet como sistemas terminales (véase el recuadro Historia).

Los sistemas terminales también se conocen como *hosts* (huéspedes), ya que albergan (es decir, ejecutan) programas de aplicación tales como navegadores web, servidores web, programas cliente de correo electrónico o servidores de correo electrónico. A lo largo de este libro utilizaremos indistintamente los términos host y sistema terminal; es decir, *host = sistema terminal*. En ocasiones, los hosts se clasifican en dos categorías: **clientes y servidores**. De un modo informal, podríamos decir que los clientes suelen ser las computadoras de escritorio y portátiles, los teléfonos inteligentes, etc., mientras que los servidores suelen ser equipos más potentes que almacenan y distribuyen páginas web, flujos de vídeo, correo electrónico, etc. Hoy en día, la mayoría de los servidores desde los que recibimos resultados de búsqueda, correo electrónico, páginas web y vídeos, residen en grandes centros de datos. Por ejemplo, Google tiene entre 50 y 100 centros de datos, incluyendo unos 15 grandes centros, cada uno con más de 100.000 servidores.

1.2.1 Redes de acceso

Una vez vistas las aplicaciones y los sistemas terminales existentes en la “frontera de la red”, podemos pasar a ver las **redes de acceso – la red que conecta físicamente un sistema terminal con el primer router** (conocido también como “router de frontera”) de la ruta existente entre el sistema terminal y cualquier otro sistema terminal distante. La Figura 1.4 muestra varios tipos de redes de acceso, resaltadas mediante líneas gruesas y oscuras, junto con el entorno (doméstico, empresarial y acceso móvil inalámbrico de área extensa) en el que se utilizan.

Acceso doméstico: DSL, cable, FTTH, acceso telefónico y satélite

En 2014, más del 78% de las viviendas en los países desarrollados disponían de acceso a Internet, siendo los países líderes Corea, Países Bajos, Finlandia y Suecia, donde disponen de acceso a

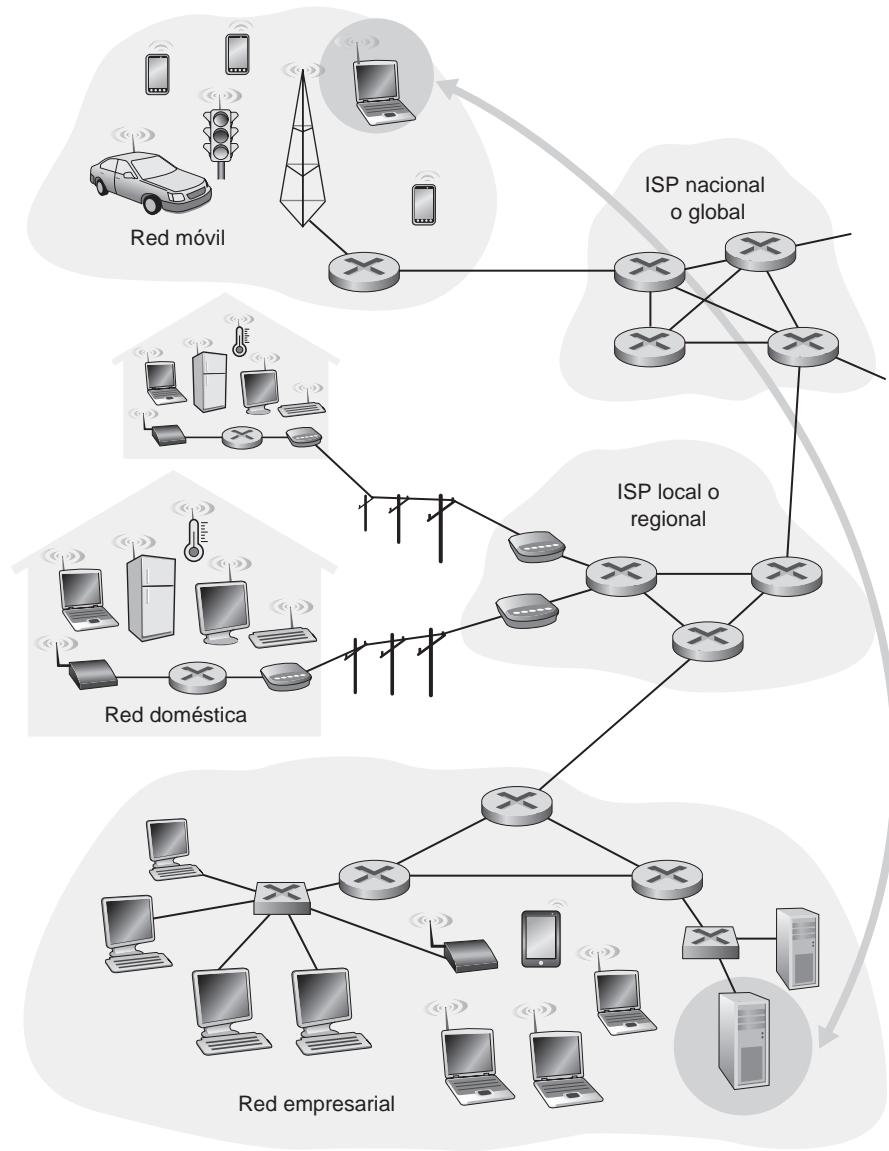


Figura 1.3 ♦ Interacción de los sistemas terminales.

Internet más del 80% de las viviendas, casi todas ellas a través de conexión de banda ancha de alta velocidad [ITU 2015]. Dado este uso generalizado de redes de acceso domésticas, comenzemos nuestro repaso de las redes de acceso analizando cómo se conectan las viviendas a Internet.

Hoy en día, los dos tipos de acceso residencial de banda ancha predominantes son las líneas **DSL** (**Digital Subscriber Line, Línea de abonado digital**) y el cable. Por regla general, los domicilios particulares contratan el servicio DSL de acceso a Internet con la misma empresa telefónica local (telco) que les proporciona el acceso telefónico local fijo. Por tanto, cuando se utiliza el acceso mediante DSL, la compañía telefónica del cliente también actúa como ISP. Como se muestra en la Figura 1.5, el módem DSL de cada cliente utiliza la línea telefónica existente (hilo de cobre de par trenzado, del que hablaremos en la Sección 1.2.2) para intercambiar datos con un multiplexor de acceso DSL (DSLAM), ubicado en la central de la compañía telefónica local. El módem DSL de la vivienda toma los datos digitales y los traduce a tonos de alta frecuencia para su transmisión a través

HISTORIA

LA INTERNET DE LAS COSAS

¿Se imagina un mundo en el que prácticamente todo esté conectado a Internet por vía inalámbrica? ¿Un mundo en el que la mayoría de las personas, vehículos, bicicletas, gafas, relojes, equipos hospitalarios, sensores domóticos, aulas, sistemas de videovigilancia, sensores atmosféricos, productos expuestos en los comercios y mascotas estén conectados? Puede que este mundo de la Internet de las Cosas (IoT) esté ya a la vuelta de la esquina.

Según algunas estimaciones, en 2015 había ya 5.000 millones de cosas conectadas a Internet, y ese número podría alcanzar los 25.000 millones en 2020 [Gartner 2014]. Entre esas cosas se incluyen nuestros teléfonos inteligentes, que ya nos siguen a todas partes en nuestros domicilios, oficinas y vehículos, informando de nuestra geolocalización y nuestros datos de uso a nuestro ISP y a las aplicaciones de Internet. Pero además de nuestros teléfonos inteligentes, ya están disponibles como productos una amplia variedad de “cosas” no tradicionales. Por ejemplo, existen complementos conectados a Internet (*wearables*), incluyendo relojes (de Apple y muchos otros) y gafas. Las gafas conectadas a Internet pueden, por ejemplo, cargar en la nube todo lo que vemos, permitiéndonos compartir nuestras experiencias visuales en tiempo real con personas de todo el mundo. Ya hay disponibles cosas conectadas a Internet para viviendas inteligentes, incluyendo termostatos conectados a Internet que podemos controlar de manera remota desde nuestro teléfono inteligente; o básculas conectadas a Internet que nos permiten revisar gráficamente, desde nuestro teléfono inteligente, el progreso de nuestra dieta. Existen juguetes conectados a Internet, como por ejemplo muñecas que reconocen e interpretan las palabras de un niño y responden apropiadamente.

La Internet de las Cosas ofrece ventajas potencialmente revolucionarias a los usuarios. Pero al mismo tiempo, también existen enormes riesgos de seguridad y confidencialidad. Por ejemplo, los atacantes podrían ser capaces, vía Internet, de acceder a los dispositivos de la Internet de las Cosas, o a los servidores que recopilan datos de esos dispositivos. O por ejemplo, un atacante podría acceder a una muñeca conectada a Internet y hablar directamente con un niño. O podría entrar en una base de datos que almacene información personal sobre nuestra salud y nuestras actividades, recopilada a partir de los dispositivos que vestimos. Estos problemas de seguridad y confidencialidad podrían socavar la confianza del consumidor que se necesita para que las tecnologías desarrollen todo su potencial, y podrían hacer que la adopción de las mismas fuera menos generalizada [FTC 2015].

de los hilos telefónicos hasta la central; las señales analógicas de una gran cantidad de esas viviendas son vueltas a traducir a datos digitales en el DSLAM.

La línea telefónica residencial transporta simultáneamente los datos y las señales telefónicas tradicionales, las cuales se codifican a frecuencias distintas:

- Un canal de descarga de alta velocidad, en la banda de 50 kHz a 1 MHz.
- Un canal de carga de velocidad media, en la banda de 4 kHz a 50 kHz.
- Un canal telefónico ordinario bidireccional, en la banda de 0 kHz a 4 kHz.

Este método hace que el único enlace DSL existente se comporte como tres enlaces separados, de manera que una llamada de teléfono y una conexión a Internet pueden compartir el enlace DSL a un mismo tiempo. (En la Sección 1.3.1 describiremos esta técnica de multiplexación por división de frecuencia.) En el lado del cliente, las señales que llegan al domicilio son separadas en señales de datos y telefónicas mediante un circuito separador (*splitter*), que reenvía la señal de datos al módem DSL. En el lado de la compañía telefónica, en la central, el multiplexor DSLAM separa las señales de datos y de telefonía y envía los datos a Internet. Cientos o incluso miles de viviendas pueden estar conectadas a un mismo DSLAM [Dischinger 2007].

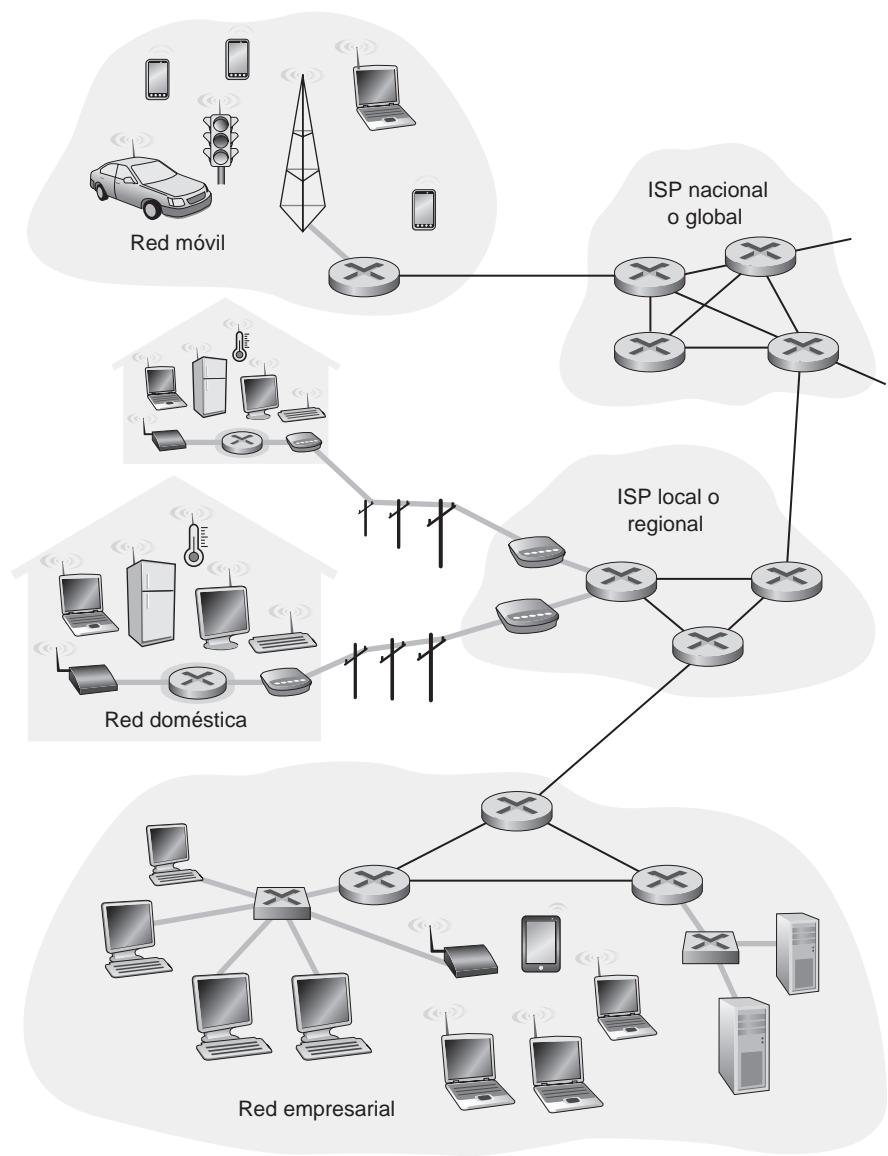


Figura 1.4 ♦ Redes de acceso.

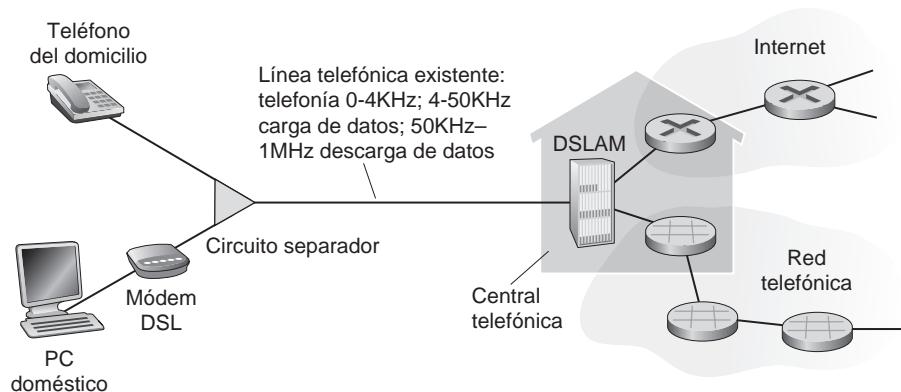


Figura 1.5 ♦ Acceso a Internet mediante DSL.

Los estándares DSL definen múltiples velocidades de transmisión, incluyendo 12 Mbps de bajada (descarga) y 1,8 Mbps de subida (carga) [ITU 1999], y 55 Mbps de bajada y 15 Mbps de subida [ITU 2006]. Puesto que las velocidades de subida y de bajada son diferentes, se dice que este tipo de acceso es asimétrico. Las velocidades de transmisión reales de subida y de bajada que se logren obtener pueden ser inferiores a las anteriormente indicadas, ya que el proveedor del servicio DSL puede limitar deliberadamente la velocidad de una vivienda cuando ofrece un servicio por niveles (diferentes velocidades disponibles a diferentes precios). La velocidad máxima está limitada también por la distancia existente entre la vivienda y la central telefónica, por el calibre de la línea de par trenzado y por el grado de interferencia eléctrica. Los ingenieros han diseñado expresamente los sistemas DSL para distancias cortas entre el domicilio y la central; generalmente, si el domicilio no se encuentra en un radio de entre 8 y 16 kilómetros de la central, el usuario deberá recurrir a una forma alternativa de acceso a Internet.

Mientras que la tecnología DSL emplea la infraestructura local existente de la compañía telefónica, el acceso por cable a Internet utiliza la infraestructura de televisión por cable existente. Las viviendas obtienen el acceso por cable a Internet de la misma compañía que les proporciona la televisión por cable. Como se ilustra en la Figura 1.6, se usa fibra óptica para conectar el terminal de cabecera del cable a una serie de nodos de área situados en el vecindario, a partir de los cuales se utiliza cable coaxial tradicional para llegar a todos los domicilios. Cada nodo de área suele dar soporte a entre 500 y 5.000 viviendas. Puesto que en este sistema se emplea tanto cable coaxial como fibra, a menudo se denomina sistema HFC (*Hybrid Fiber Coax*, Híbrido de fibra y coaxial).

El acceso por cable a Internet requiere el uso de un módem especial, que se conoce como módem por cable. Al igual que un módem DSL, normalmente el módem por cable es un dispositivo externo que se conecta a un PC de la vivienda a través de un puerto Ethernet (en el Capítulo 6 veremos más detalles acerca de Ethernet). En el terminal de cabecera del cable, un sistema CMTS (*Cable Modem Termination System*, Sistema de terminación del módem de cable) cumple una función similar al DSLAM de la red DSL —transformar a formato digital la señal analógica enviada por los modems de cable de numerosas viviendas situadas aguas abajo. Los modems por cable dividen la red HFC en dos canales: un canal de descarga y un canal de carga. Al igual que en el caso de DSL, el acceso suele ser asimétrico, teniendo el canal de descarga asignada normalmente una velocidad de transmisión mayor que el canal de carga. El estándar DOCSIS 2.0 define velocidades de descarga de hasta 42,8 Mbps y velocidades de carga de hasta 30,7 Mbps. Como en el caso de las redes DSL, la velocidad máxima alcanzable puede no llegar a obtenerse, debido a las deficiencias del medio o a que las velocidades de datos contratadas sean más bajas.

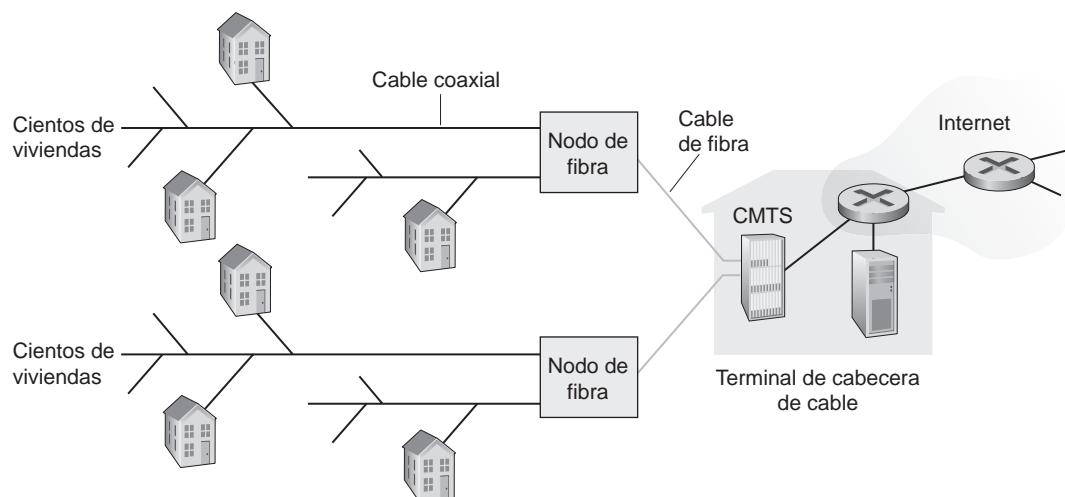


Figura 1.6 ♦ Red de acceso híbrida de fibra óptica y cable coaxial.

Una característica importante del acceso a Internet por cable es que se trata de un medio de difusión compartido. Es decir, cada uno de los paquetes enviados por el extremo de cabecera viaja a través de cada enlace hasta cada vivienda y los paquetes enviados desde las viviendas viajan a través del canal de subida hasta el extremo de cabecera. Así, si varios usuarios descargan simultáneamente un archivo de vídeo a través del canal de descarga, la velocidad real a la que cada usuario recibe su archivo de vídeo será significativamente menor que la velocidad agregada de descarga por cable. Por el contrario, si solo hay unos pocos usuarios activos que están navegando por la Web, cada uno de ellos recibirá las páginas web a la velocidad de descarga máxima del cable, ya que los usuarios rara vez solicitarán sus páginas web exactamente al mismo tiempo. Puesto que el canal de subida también es compartido, se necesita un protocolo distribuido de acceso múltiple para coordinar las transmisiones y evitar las colisiones (veremos este problema de las colisiones con más detalle en el Capítulo 6).

Aunque las redes DSL y de cable representan en la actualidad más del 85% del acceso de banda ancha residencial en Estados Unidos, una tecnología emergente que proporciona velocidades aún más altas es la tecnología **FTTH** (*Fiber-To-The-Home, Fibra hasta el hogar*) [FTTH Council 2016]. Como su nombre sugiere, el concepto en que se basa FTTH es muy simple: proporcionar una ruta de fibra óptica directa hasta la vivienda desde la central telefónica. Muchos países —incluyendo los Emiratos Árabes Unidos, Corea del Sur, Hong Kong, Japón, Singapur, Taiwán, Lituania y Suecia— tienen hoy tasas de penetración en el mercado residencial superiores al 30% [FTTH Council 2016].

Existen varias tecnologías que compiten por la distribución a través de fibra óptica desde las centrales a los hogares. La red de distribución óptica más simple se denomina fibra directa, en la que existe una fibra que sale de la central hasta cada domicilio. Sin embargo, lo más habitual es que cada fibra saliente de la central sea compartida por muchas viviendas y esta no se divida en fibras individuales específicas del cliente hasta llegar a un punto muy próximo a las viviendas. Hay disponibles dos arquitecturas de distribución de fibra óptica que llevan a cabo esta separación: las redes ópticas activas (AON, *Active Optical Network*) y las redes ópticas pasivas (PON, *Passive Optical Network*). Las redes AON son fundamentalmente redes Ethernet commutadas, de las cuales hablaremos en el Capítulo 6.

Aquí vamos a ver brevemente las redes ópticas pasivas, que se utilizan en el servicio FIOS de Verizon. La Figura 1.7 muestra un sistema FTTH que utiliza la arquitectura de distribución PON. Cada vivienda dispone de una terminación de red óptica (ONT, *Optical Network Terminator*), que se conecta a un distribuidor óptico mediante un cable de fibra óptica dedicado. El distribuidor combina una cierta cantidad de viviendas (normalmente menos de 100) en un único cable de fibra óptica compartido, que se conecta a una terminación de línea óptica (OLT, *Optical Line Terminator*) en la central de la compañía telefónica. La OLT, que realiza la conversión de señales ópticas en

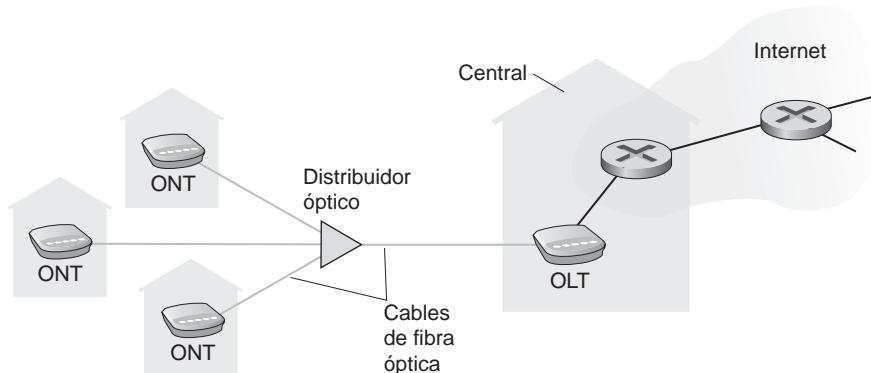


Figura 1.7 ♦ Acceso a Internet mediante FTTH.

eléctricas, se conecta a Internet a través de un router de la compañía telefónica. En los domicilios, los usuarios conectan su router doméstico (normalmente un router inalámbrico) con la ONT y acceden a Internet a través de este router. En la arquitectura PON, todos los paquetes enviados desde la OLT al distribuidor se replican en este distribuidor (de forma similar a un terminal de cabecera de cable).

En teoría, la tecnología FTTH puede proporcionar velocidades de acceso a Internet del orden de los gigabits por segundo. Sin embargo, la mayoría de los ISP de FTTH ofrecen diferentes velocidades, siendo lógicamente las tarifas más caras cuanto mayor es la velocidad. La velocidad media de descarga para los clientes de FTTH en Estados Unidos era de aproximadamente 20 Mbps en 2011 (comparada con los 13 Mbps de las redes de acceso por cable y los menos de 5 Mbps de DSL) [FTTH Council 2011b].

Hay otras dos tecnologías de redes de acceso que también se usan para proporcionar acceso a Internet a las viviendas. En aquellos lugares donde no estén disponibles las tecnologías DSL, FTTH y de cable (por ejemplo, en ciertos entornos rurales), puede utilizarse un enlace vía satélite para conectar a Internet un domicilio a velocidades superiores a 1 Mbps; dos de los proveedores de acceso vía satélite son StarBand y HughesNet. El acceso telefónico a través de líneas telefónicas tradicionales se basa en el mismo modelo que DSL – un módem doméstico se conecta a través de la línea telefónica a un módem situado en las instalaciones del ISP. Comparado con DSL y otras redes de acceso de banda ancha, el acceso telefónico es insoportablemente lento, de solo 56 kbps.

Acceso empresarial (y doméstico): Ethernet y WiFi

En los campus universitarios y corporativos, y cada vez más en entornos domésticos, se utiliza una red de área local (LAN, *Local Area Network*) para conectar un sistema terminal al router de frontera. Aunque existen muchos tipos de tecnologías LAN, Ethernet es con mucho la tecnología de acceso predominante en las redes corporativas, universitarias y domésticas. Como se ilustra en la Figura 1.8, los usuarios de Ethernet utilizan cable de cobre de par trenzado para conectarse a un switch Ethernet, una tecnología que se verá en detalle en el Capítulo 6. A su vez, el switch Ethernet (o una red de tales switches) se conecta a Internet. Con acceso Ethernet, normalmente los usuarios disponen de velocidades de acceso de 100 Mbps o 1 Gbps al switch Ethernet, mientras que los servidores pueden disponer de acceso de 1 Gbps o incluso 10 Gbps.

Sin embargo, cada vez es más habitual que los usuarios accedan a Internet a través de conexiones inalámbricas desde computadoras portátiles, teléfonos inteligentes, tabletas y otras “cosas” (véase el recuadro anterior “La Internet de las cosas”). En un entorno de LAN inalámbrica, los usuarios inalámbricos transmiten/reciben paquetes hacia/desde un punto de acceso que está conectado a la red empresarial (probablemente utilizando Ethernet cableada), que a su vez se conecta a la red Internet cableada. Habitualmente, los usuarios de una LAN inalámbrica deben encontrarse a unas

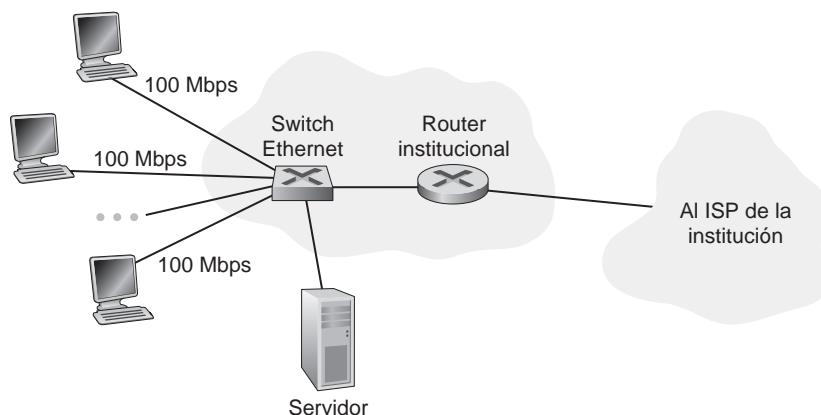


Figura 1.8 ♦ Acceso a Internet utilizando tecnología Ethernet.

pocas decenas de metros del punto de acceso. Actualmente, el acceso mediante LAN inalámbrica basada en la tecnología IEEE 802.11, más coloquialmente conocido como WiFi, podemos encontrarlo por todas partes: universidades, oficinas, cafés, aeropuertos, domicilios e incluso en los aviones. En muchas ciudades, alguien puede estar parado en la esquina de una calle y encontrarse dentro del alcance de diez o veinte estaciones base (para ver un mapa global navegable de estaciones base 802.11 descubiertas y registradas en un sitio web por personas que disfrutan haciendo este tipo de cosas, consulte [wigle.net 2009]). Como se explica detalladamente en el Capítulo 7, hoy en día la tecnología 802.11 proporciona una velocidad de transmisión compartida de hasta 100 Mbps o superior.

Aunque las redes de acceso Ethernet y WiFi se implantaron inicialmente en entornos empresariales (corporativos, universitarios), en los últimos tiempos se han convertido en componentes relativamente comunes de las redes domésticas. Muchas viviendas combinan acceso residencial de banda ancha (es decir, modems por cable o DSL) con estas baratas tecnologías de LAN inalámbrica para crear redes domésticas potentes [Edwards 2011]. La Figura 1.9 muestra el esquema de una red doméstica típica. Esta red doméstica está formada por un portátil con función de itinerancia (*roaming*) y un PC de sobremesa; una estación base (el punto de acceso inalámbrico), que se comunica con el portátil y otros dispositivos inalámbricos de la vivienda; un módem por cable, que proporciona el acceso de banda ancha a Internet; y un router, que interconecta la estación base y el PC de sobremesa con el módem por cable. Esta red permite a los miembros de la familia disponer de acceso de banda ancha a Internet, pudiendo uno de ellos acceder mientras se mueve de la cocina a los dormitorios y al jardín.

Acceso inalámbrico de área extensa: 3G y LTE

Cada vez se utilizan más dispositivos como los iPhone o los basados en Android para enviar mensajes, compartir fotos en redes sociales, ver películas y oír música a través de la red mientras nos desplazamos de un sitio a otro. Estos dispositivos emplean la misma infraestructura inalámbrica de la telefonía móvil para enviar/recibir paquetes a través de una estación base operada por el proveedor de telefonía móvil. A diferencia de WiFi, un usuario solo necesita estar a unas pocas decenas de kilómetros (en vez de unas pocas decenas de metros) de la estación base.

Las empresas de telecomunicaciones han hecho grandes inversiones en lo que se conoce como redes inalámbricas de tercera generación (3G), que proporcionan acceso inalámbrico a Internet mediante una red de área extensa de conmutación de paquetes, a velocidades por encima de 1 Mbps. Pero ya se están implantando tecnologías de acceso de área extensa con velocidades aun más altas: una cuarta generación (4G) de redes inalámbricas de área extensa. LTE (*Long-Term Evolution*, Evolución a largo plazo, un claro candidato al Premio Anual al Peor Acrónimo del Año) tiene sus raíces en la tecnología 3G y puede conseguir velocidades superiores a 10 Mbps. En las redes LTE comerciales se han medido velocidades de descarga de varias decenas de Mbps. En el Capítulo 7 hablaremos de los principios básicos de las redes inalámbricas y la movilidad, así como de las tecnologías WiFi, 3G y LTE (¡y otras cosas!).

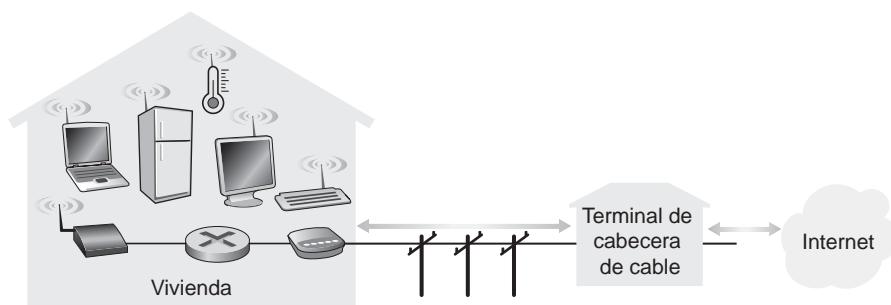


Figura 1.9 ♦ Una red doméstica típica.

1.2.2 Medios físicos

En la subsección anterior hemos proporcionado una panorámica de algunas de las tecnologías de acceso a red más importantes disponibles para Internet. Según hemos ido describiendo estas tecnologías, hemos indicado los medios físicos utilizados. Por ejemplo, hemos dicho que la tecnología HFC emplea una combinación de cable de fibra óptica y de cable coaxial. También hemos señalado que Ethernet y DSL utilizan cable de cobre. Y dijimos que las redes para acceso móvil usan el espectro de radio. En esta subsección vamos a hacer una breve introducción a estos y otros medios de transmisión que se emplean habitualmente en Internet.

Para definir lo que se entiende por medio físico, reflexionemos sobre la breve vida de un bit. Imagine un bit que viaja desde un sistema terminal atravesando una serie de enlaces y routers hasta otro sistema terminal. ¡Este pobre bit se desplaza de un lado a otro y es transmitido un montón de veces! En primer lugar, el sistema terminal de origen transmite el bit y poco tiempo después el primer router de la serie recibe dicho bit; el primer router transmite entonces el bit y poco después lo recibe el segundo router, y así sucesivamente. Por tanto, nuestro bit, al viajar desde el origen hasta el destino, atraviesa una serie de parejas de transmisores y receptores. En cada par transmisor-receptor, el bit se envía mediante ondas electromagnéticas o pulsos ópticos a lo largo de un **medio físico**. Este medio físico puede tener muchas formas y no tiene por qué ser del mismo tipo para cada par transmisor-receptor existente a lo largo de la ruta. Entre los ejemplos de medios físicos se incluyen el cable de cobre de par trenzado, el cable coaxial, el cable de fibra óptica multimodo, el espectro de radio terrestre y el espectro de radio por satélite. Los medios físicos se pueden clasificar dentro de dos categorías: **medios guiados** y **medios no guiados**. En los medios guiados, las ondas se canalizan a través de un medio sólido, como por ejemplo un cable de fibra óptica, un cable de cobre de par trenzado o un cable coaxial. En los medios no guiados, las ondas se propagan por la atmósfera y el espacio exterior, tal como ocurre en las redes LAN inalámbricas o en un canal de satélite digital.

Pero antes de abordar las características de los distintos tipos de medios, veamos algunos detalles acerca de los costes. El coste real de un enlace físico (cable de cobre, de fibra óptica, etc.) suele ser relativamente pequeño cuando se compara con los restantes costes de la red. En particular, el coste de mano de obra asociado con la instalación del enlace físico puede ser varios órdenes de magnitud mayor que el coste del material. Por ello, muchos constructores instalan cables de par trenzado, de fibra óptica y coaxiales en todas las habitaciones de los edificios. Incluso aunque inicialmente solo se utilice uno de los medios, existen muchas posibilidades de que se emplee algún otro medio físico en un futuro próximo y, por tanto, se ahorre dinero al no tener que tirar cables adicionales.

Cable de cobre de par trenzado

El medio de transmisión guiado más barato y más comúnmente utilizado es el cable de cobre de par trenzado. Se ha utilizado durante más de un siglo en las redes telefónicas. De hecho, más del 99 por ciento de las conexiones cableadas utilizan cable de cobre de par trenzado entre el propio teléfono y el conmutador telefónico local. La mayoría de nosotros hemos visto cable de par trenzado en nuestros hogares (¡o en los de nuestros padres o abuelos!) y entornos de trabajo. Este cable consta de dos hilos de cobre aislados, de un milímetro de espesor cada uno de ellos, que siguen un patrón regular en espiral. Los hilos se trenzan para reducir las interferencias eléctricas procedentes de pares similares próximos. Normalmente, se meten una serie de pares dentro de un mismo cable, envolviendo todos los pares en una pantalla protectora. Cada par de hilos constituye un único enlace de comunicaciones. El **par trenzado no apantallado (UTP, Unshielded Twisted Pair)** se utiliza habitualmente para redes de computadoras confinadas dentro de un edificio, es decir, para **redes LAN**. La velocidad de transmisión de datos de las LAN actuales que emplean cables de par trenzado varía entre 10 Mbps y 10 Gbps. Las velocidades de transmisión de datos que se pueden alcanzar dependen del espesor del cable y de la distancia existente entre el transmisor y el receptor.

Cuando surgió la tecnología de la fibra óptica en la década de 1980, muchas personas despreciaron el cable de par trenzado a causa de sus relativamente bajas velocidades de transmisión. Algunos pensaron incluso que la fibra óptica desplazaría por completo al cable de par trenzado.

Pero el cable de par trenzado no se dio por vencido tan fácilmente. La tecnología moderna de par trenzado, como por ejemplo los cables de categoría 6a, pueden alcanzar velocidades de datos de 10 Gbps para distancias de hasta 100 metros. Al final, los cables de par trenzado se han establecido como la solución dominante para las redes LAN de alta velocidad.

Como hemos mencionado anteriormente, los cables de par trenzado también suelen utilizarse para el acceso a Internet de tipo residencial. Hemos dicho que los modems de acceso telefónico permiten proporcionar acceso a velocidades de hasta 56 kbps utilizando cables de par trenzado. También hemos comentado que la tecnología DSL (Digital Subscriber Line) ha permitido a los usuarios residenciales acceder a Internet a velocidades de decenas de Mbps empleando cables de par trenzado (siempre y cuando los usuarios vivan en las proximidades de la central de comunicaciones del ISP).

Cable coaxial

Al igual que el par trenzado, el cable coaxial consta de dos conductores de cobre, pero dispuestos de forma concéntrica, en lugar de en paralelo. Con esta construcción y un aislamiento y apantallamiento especiales, el cable coaxial puede proporcionar altas velocidades de transmisión de datos. El cable coaxial es bastante común en los sistemas de televisión por cable. Como ya hemos visto, recientemente los sistemas de televisión por cable han comenzado a incorporar modems por cable con el fin de proporcionar a los usuarios residenciales acceso a Internet a velocidades de decenas de Mbps. En la televisión por cable y en el acceso a Internet por cable, el transmisor desplaza la señal digital a una banda de frecuencia específica y la señal analógica resultante se envía desde el transmisor a uno o más receptores. El cable coaxial puede utilizarse como un medio compartido guiado; es decir, una serie de sistemas terminales pueden estar conectados directamente al cable, recibiendo todos ellos lo que envíen los otros sistemas terminales.

*uno adentro
del otro*

Fibra óptica

La fibra óptica es un medio flexible y de muy pequeño espesor que conduce pulsos de luz, representando cada pulso un bit. Un único cable de fibra óptica puede soportar velocidades de bit tremadamente altas, de hasta decenas o incluso centenares de gigabits por segundo. La fibra óptica es inmune a las interferencias electromagnéticas, presenta una atenuación de la señal muy baja para distancias de hasta 100 kilómetros y es muy difícil que alguien pueda llevar a cabo un “pinchazo” en una de estas líneas. Estas características hacen de la fibra óptica el medio de transmisión guiado a larga distancia preferido, especialmente para los enlaces transoceánicos. Muchas de las redes telefónicas para larga distancia de Estados Unidos y otros países utilizan hoy día exclusivamente fibra óptica. La fibra óptica también es el medio predominante en las redes troncales de Internet. Sin embargo, el alto coste de los dispositivos ópticos —como transmisores, receptores y switches— está entorpeciendo su implantación para el transporte a corta distancia, como por ejemplo en el caso de una LAN o en el domicilio en una red de acceso residencial. Las velocidades de enlace estándar para portadora óptica (OC, *Optical Carrier*) van desde 51,8 Mbps a 39,8 Gbps; suele hacerse referencia a estas especificaciones mediante la designación OC-*n*, donde la velocidad del enlace es igual a $n \times 51,8$ Mbps. Entre los estándares en uso actuales se encuentran: OC-1, OC-3, OC-12, OC-24, OC-48, OC-96, OC-192, OC-768. Consulte [Mukherjee 2006, Ramaswami 2010] para ver más información acerca de diversos aspectos de las redes ópticas.

Canales de radio terrestres

Los canales de radio transportan señales dentro del espectro electromagnético. Constituyen un medio atractivo, porque no requieren la instalación de cables físicos, pueden atravesar las paredes, proporcionan conectividad a los usuarios móviles y potencialmente pueden transportar una señal a grandes distancias. Las características de un canal de radio dependen de forma significativa

del entorno de propagación y de la distancia a la que la señal tenga que ser transportada. Las consideraciones ambientales determinan la pérdida del camino y la atenuación por sombra (que disminuyen la intensidad de la señal a medida que recorre una distancia y rodea/atraviesa los objetos que obstruyen su camino), la atenuación multicamino (debido a la reflexión de la señal en los objetos que interfieren) y las interferencias (debidas a otras transmisiones y a señales electromagnéticas en general).

Los canales de radio terrestre pueden clasificarse en tres amplios grupos: aquellos que operan a distancia muy corta (por ejemplo, de uno o dos metros), los que operan en áreas locales, normalmente con un alcance de entre diez y unos cientos de metros y los que operan en un área extensa, con alcances de decenas de kilómetros. Los dispositivos personales, como los cascos inalámbricos, teclados y ciertos dispositivos médicos, operan a distancias cortas; las tecnologías LAN inalámbricas descritas en la Sección 1.2.1 emplean canales de radio de área local y las tecnologías de acceso celulares utilizan canales de radio de área extensa. En el Capítulo 7 se estudian en detalle los canales de radio.

Canales de radio vía satélite

Un satélite de comunicaciones enlaza dos o más transmisores/receptores de microondas situados en la superficie, que se conocen como estaciones terrestres. El satélite recibe las transmisiones en una banda de frecuencia, regenera la señal utilizando un repetidor (véase más adelante) y transmite la señal a otra frecuencia. En este tipo de comunicaciones se emplean dos tipos de satélites: los satélites geoestacionarios y los satélites de la órbita baja terrestre (LEO, *Low-Earth Orbiting*) [Wiki Satellite 2016].

Los satélites geoestacionarios están permanentemente situados sobre el mismo punto de la Tierra. Esta presencia estacionaria se consigue poniendo el satélite en órbita a una distancia de 36.000 kilómetros por encima de la superficie terrestre. Esta enorme distancia de ida y vuelta entre la estación terrestre y el satélite introduce un significativo retardo de propagación de la señal de 280 milisegundos. No obstante, los enlaces vía satélite, que pueden operar a velocidades de cientos de Mbps, a menudo se emplean en áreas en las que no hay disponible acceso a Internet mediante DSL o cable.

Los satélites LEO se colocan mucho más cerca de la Tierra y no se encuentran permanentemente sobre un mismo punto de la superficie, sino que giran alrededor de la Tierra (al igual que lo hace la Luna) y pueden comunicarse entre sí, así como con las estaciones terrestres. Para poder proporcionar una cobertura continua a un área, es preciso poner en órbita muchos satélites. Actualmente se están desarrollando muchos sistemas de comunicaciones de baja altitud. La tecnología de satélites LEO podrá utilizarse, en algún momento del futuro, para acceder a Internet.

1.3 El núcleo de la red

Una vez que hemos examinado la frontera de Internet, vamos a adentrarnos en el núcleo de la red, la malla de comutadores de paquetes y enlaces que interconectan los sistemas terminales de Internet. En la Figura 1.10 se ha resaltado el núcleo de la red con líneas más gruesas.

1.3.1 Comutación de paquetes

En una aplicación de red, los sistemas terminales intercambian mensajes unos con otros. Los mensajes pueden contener cualquier cosa que el diseñador de la aplicación desee. Los mensajes pueden realizar una función de control (por ejemplo, los mensajes de saludo “Hola” del ejemplo anterior sobre establecimiento de la comunicación, en la Figura 1.2) o pueden contener datos, como por ejemplo un mensaje de correo electrónico, una imagen JPEG o un archivo de audio MP3. Para enviar un mensaje desde un sistema terminal de origen hasta un sistema terminal de destino, el

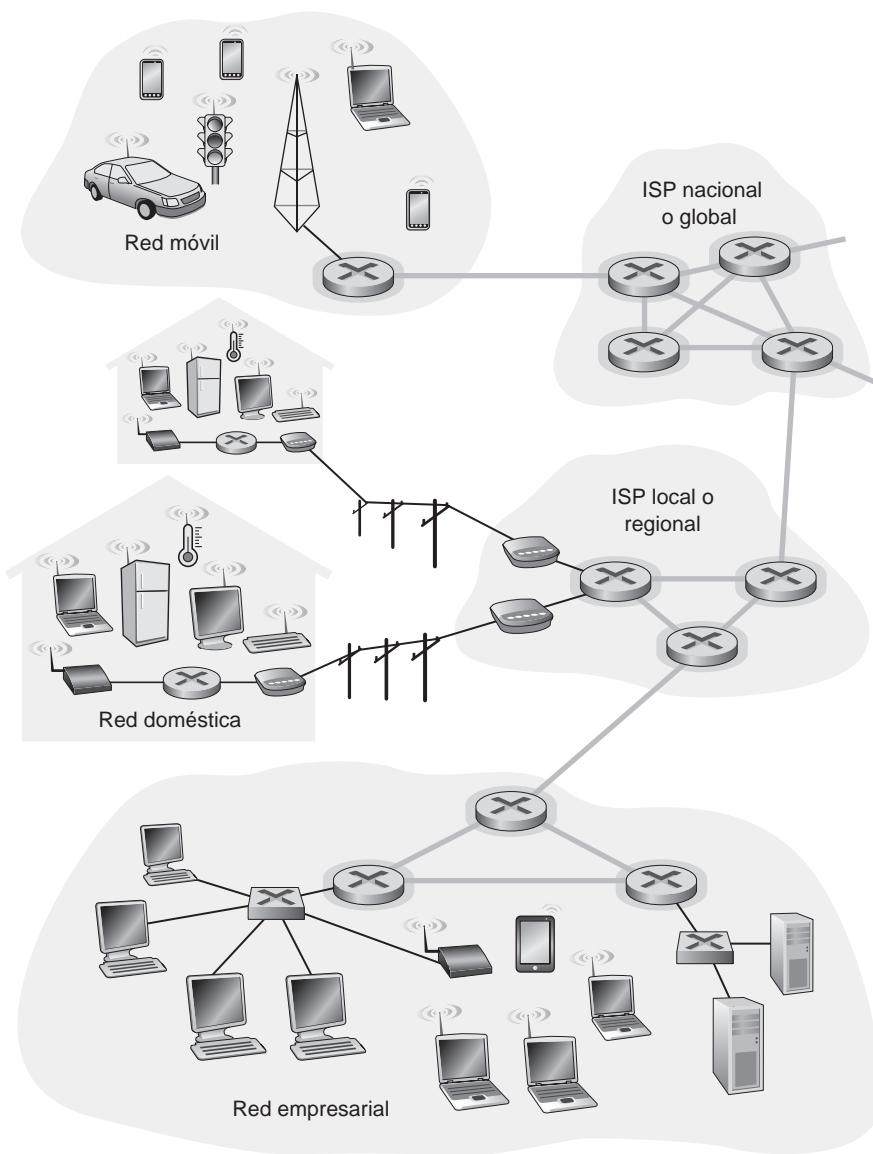


Figura 1.10 ♦ El núcleo de la red.

origen divide los mensajes largos en fragmentos de datos más pequeños que se conocen como **paquetes**. Entre el origen y el destino, cada uno de estos paquetes viaja a través de enlaces de comunicaciones y de **comunicaciones y de conmutadores de paquetes** (de los que existen dos tipos predominantes: los **routers** y los **switches de la capa de enlace**). Los paquetes se transmiten a través de cada enlace de comunicaciones a una velocidad igual a la velocidad de transmisión *máxima* del enlace. Por tanto, si un sistema terminal de origen o un conmutador de paquetes están enviando un paquete de L bits a través de un enlace cuya velocidad de transmisión es R bits/s, entonces el tiempo necesario para transmitir el paquete es igual a L/R segundos.

Transmisión de almacenamiento y reenvío

La mayoría de los conmutadores de paquetes aplican el método de **transmisión de almacenamiento y reenvío** en las entradas de los enlaces. Transmisión de almacenamiento y reenvío

significa que el conmutador de paquetes tiene que recibir el paquete completo antes de poder comenzar a transmitir el primer bit del paquete al enlace de salida. Para analizar con más detalle la transmisión de almacenamiento y reenvío, consideremos una red simple, compuesta de dos sistemas terminales conectados mediante un único router, como se muestra en la Figura 1.11. Normalmente, un router tendrá muchos enlaces entrantes, porque su trabajo consiste en conmutar un paquete entrante hacia un enlace de salida; en este ejemplo simple, el router tiene encomendada la sencilla tarea de transferir paquetes desde un enlace (de entrada) hacia el único otro enlace conectado. En este ejemplo, el origen dispone de tres paquetes para enviar hacia el destino, cada uno de ellos compuesto por L bits. En el instante temporal mostrado en la Figura 1.11, el origen ha transmitido ya parte del paquete 1, y el inicio del paquete 1 ha llegado ya al router. Puesto que el router emplea el mecanismo de almacenamiento y reenvío, en este instante de tiempo el router no puede enviar los bits que ha recibido; en lugar de ello, debe primero guardar (es decir, “almacenar”) los bits del paquete. Solo después de que el router haya recibido *todos* los bits del paquete, podrá empezar a transmitir (es decir, “reenviar”) el paquete a través del enlace de salida. Para comprender mejor la transmisión de almacenamiento y reenvío, calculemos ahora la cantidad de tiempo que transcurre desde el momento en que el origen comienza a enviar el paquete, hasta que el destino termina de recibirlo. (Aquí vamos a ignorar el retardo de propagación —el tiempo que los bits necesitan para viajar a través del cable a una velocidad cercana a la de la luz—, del que hablaremos en la Sección 1.4.) El origen comienza a transmitir en el instante 0; en el instante L/R segundos, el origen habrá transmitido todo el paquete, el cual habrá sido recibido y almacenado en el router (puesto que no hay retardo de propagación). En ese mismo instante L/R segundos, como el router ha recibido ya todo el paquete, puede comenzar a transmitirlo hacia el destino a través del enlace de salida; en el instante $2L/R$, el router habrá transmitido todo el paquete y este habrá sido recibido en su totalidad por el destino. Por tanto, el retardo total es de $2L/R$. Si, en lugar de comportarse así, el router reenviara los bits a medida que van llegando (sin esperar a recibir primero todo el paquete), entonces el retardo total sería L/R , puesto que los bits no se verían retenidos en el router. Pero, como veremos en la Sección 1.4, los routers necesitan recibir, almacenar y *procesar* el paquete completo antes de reenviarlo.

Calculemos ahora la cantidad de tiempo que transcurre desde el momento en que el origen empieza a enviar el primer paquete, hasta que el destino ha recibido los tres paquetes. Al igual que antes, en el instante L/R el router empieza a reenviar el primer paquete. Pero también en el instante L/R , el origen comenzará a enviar el segundo paquete, puesto que acaba de terminar de enviar el primer paquete completo. Por tanto, en el instante $2L/R$ el destino ha terminado de recibir el primer paquete y el router ha recibido el segundo. De forma similar, en el instante $3L/R$ el destino ha terminado de recibir los dos primeros paquetes y el router ha recibido el tercero. Finalmente, en el instante $4L/R$ el destino habrá recibido los tres paquetes.

Consideremos ahora el caso general consistente en enviar un paquete desde el origen al destino a través de una ruta compuesta por N enlaces, cada uno de ellos de velocidad R (y en la que, por tanto, hay $N-1$ routers entre el origen y el destino). Aplicando la misma lógica anterior, vemos que el retardo extremo a extremo es:

$$d_{extremo-extremo} = N \frac{L}{R} \quad (1.1)$$

Pruebe a intentar determinar cuál sería el retardo para P paquetes enviados a través de una serie de N enlaces.

Retardos de cola y pérdida de paquetes

Cada conmutador de paquetes tiene varios enlaces conectados a él y para cada enlace conectado, el conmutador de paquetes dispone de un **buffer de salida** (también denominado **cola de salida**), que almacena los paquetes que el router enviará a través de dicho enlace. El buffer de salida desempeña

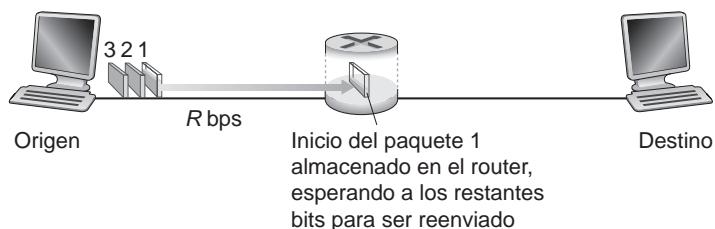


Figura 1.11 ♦ Comutación de paquetes con almacenamiento y reenvío.

un papel clave en la comutación de paquetes. Si un paquete entrante tiene que ser transmitido a través de un enlace, pero se encuentra con que el enlace está ocupado transmitiendo otro paquete, el paquete entrante tendrá que esperar en el buffer de salida. Por tanto, además de los retardos de almacenamiento y reenvío, los paquetes se ven afectados por los **retardos de cola** del buffer de salida. Estos retardos son variables y dependen del nivel de congestión de la red. Puesto que la cantidad de espacio en el buffer es finita, un paquete entrante puede encontrarse con que el buffer está completamente lleno con otros paquetes que esperan a ser transmitidos. En este caso, se producirá una **pérdida de paquetes**: el paquete que acaba de llegar o uno que ya se encuentre en la cola será descartado.

La Figura 1.12 ilustra una red de comutación de paquetes simple. Al igual que en la Figura 1.11, los paquetes se han representado mediante bloques tridimensionales. El ancho de un bloque representa el número de bits que contiene el paquete. En esta figura, todos los paquetes tienen el mismo ancho y, por tanto, la misma longitud. Suponga ahora que los hosts A y B están enviando paquetes al host E. En primer lugar, los hosts A y B envían sus paquetes a través de los enlaces Ethernet a 100 Mbps hasta el primer router. A continuación, este dirige los paquetes al enlace de 15 Mbps. Si, durante un corto intervalo de tiempo, la velocidad de llegada de los paquetes al router (medida en bits por segundo) excede los 15 Mbps, se producirá congestión en el router, a medida que los paquetes se ponen en cola en el buffer del enlace de salida, antes de poder ser transmitidos a través de él. Por ejemplo, si tanto el Host A como el Host B envían simultáneamente una ráfaga de cinco paquetes seguidos, entonces la mayoría de esos paquetes necesitarán esperar un cierto tiempo en la cola. Esta situación es, de hecho, completamente análoga a muchas situaciones de la vida cotidiana —por ejemplo, cuando nos vemos obligados a esperar en la cola de un cajero o en la del peaje de una autopista. En la Sección 1.4 examinaremos más detalladamente el retardo de cola.

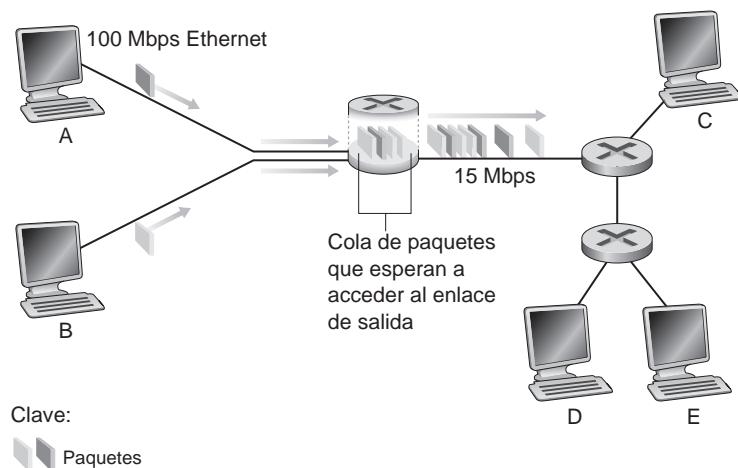


Figura 1.12 ♦ Comutación de paquetes.

Tablas de reenvío y protocolos de enrutamiento

Anteriormente hemos dicho que un router toma un paquete entrante a través de uno de sus enlaces de comunicaciones y reenvía dicho paquete a través de otro de sus enlaces de comunicaciones. ¿Pero cómo determina el router cuál es el enlace a través del cual deberá reenviar el paquete? En la práctica, el reenvío de paquetes se lleva a cabo de distinta manera en los diferentes tipos de redes de computadoras. En este capítulo, vamos a describir brevemente el método empleado por Internet.

En Internet, cada sistema terminal tiene asignada una dirección, denominada dirección IP. Cuando un sistema terminal de origen quiere enviar un paquete a un sistema terminal de destino, el origen incluye la dirección IP de destino en la cabecera del paquete. Al igual que las direcciones postales, esta dirección tiene una estructura jerárquica. Cuando un paquete llega a un router de la red, el router examina una parte de la dirección de destino del paquete y lo reenvía a un router adyacente. Más específicamente, cada router dispone de una tabla de reenvío que asigna las direcciones de destino (o una parte de las mismas) a los enlaces salientes de ese router. Cuando llega un paquete a un router, este examina la dirección y busca en su tabla de reenvío esa dirección de destino, para determinar el enlace de salida apropiado. A continuación, el router dirige el paquete a ese enlace de salida.

El proceso de enrutamiento terminal a terminal es análogo al que sigue el conductor de un automóvil que no utiliza un mapa, sino que prefiere preguntar cómo llegar hasta una determinada dirección. Por ejemplo, suponga que Juan sale de Filadelfia y tiene que llegar al 156 de la calle Lakeside Drive en Orlando, Florida. Lo primero que hace Juan es dirigirse a la estación de servicio más próxima y preguntar cómo llegar a su destino. El empleado se queda con el nombre del estado, Florida, y le dice que debe tomar la autopista interestatal I-95 Sur y que existe una entrada a la misma nada más salir de la estación de servicio. También le dice a Juan que una vez que haya entrado en Florida, pregunte a alguien cómo llegar a su destino. Así, Juan toma la I-95 Sur hasta Jacksonville, Florida, lugar donde vuelve a preguntar en otra estación de servicio. El dependiente extrae de la dirección la información que hace referencia a Orlando y le dice que debe continuar por la I-95 hasta Daytona Beach y que luego pregunte. En otra estación de servicio de Daytona Beach, el empleado extrae de nuevo la información referente a Orlando y le dice que tomando la I-4 llegará directamente a Orlando. Juan toma la I-4 y la abandona en la salida correspondiente a Orlando. De nuevo se detiene en otra gasolinera y esta vez el dependiente extrae la parte de la información de la dirección referente a Lakeside Drive, y le indica la carretera que debe seguir para llegar allí. Una vez que Juan se encuentra en Lakeside Drive, pregunta a un niño que va en bicicleta cómo llegar a su destino. El niño extrae el dato 156 de la dirección y le señala una casa. Por fin, Juan ha llegado a su destino. En esta analogía, los dependientes de las estaciones de servicio y el niño de la bicicleta son los routers.

Acabamos de ver que un router utiliza la dirección de destino de un paquete para indexar una tabla de reenvío y determinar el enlace de salida apropiado. Pero esta afirmación nos lleva a la siguiente pregunta: ¿cómo se definen las tablas de reenvío? ¿Se configuran manualmente en cada router o Internet utiliza un procedimiento más automatizado? Estas cuestiones se abordan en detalle en el Capítulo 5, pero para ir abriendo boca, diremos que Internet dispone de una serie de protocolos de enrutamiento especiales que se utilizan para definir automáticamente las tablas de reenvío. Por ejemplo, un protocolo de enrutamiento puede determinar la ruta más corta desde cada router hasta cada destino y usar esas rutas más cortas para configurar las tablas de reenvío en los routers.

¿Cómo podríamos ver la ruta terminal a terminal que siguen los paquetes en Internet? Le invitamos a que se ponga manos a la obra, utilizando el programa Traceroute; simplemente, visite el sitio <http://www.traceroute.org>, elija un origen en el país que quiera y trace la ruta desde ese origen hasta su computadora. (Para obtener más información acerca de Traceroute, consulte la Sección 1.4.).

1.3.2 Conmutación de circuitos

Existen dos métodos fundamentales que permiten transportar los datos a través de una red de enlaces y conmutadores: la **conmutación de circuitos** y la **conmutación de paquetes**. Ya hemos hablado de las redes de conmutación de paquetes en la subsección anterior, así que ahora fijaremos nuestra atención en las redes de conmutación de circuitos.

En las redes de conmutación de circuitos, los recursos necesarios a lo largo de una ruta (buffers, velocidad de transmisión del enlace) que permiten establecer la comunicación entre los sistemas terminales están *reservados* durante el tiempo que dura la sesión de comunicación entre dichos sistemas terminales. En las redes de conmutación de paquetes, estos recursos no están reservados; los mensajes de una sesión utilizan los recursos bajo petición y, en consecuencia, pueden tener que esperar (es decir, ponerse en cola) para poder acceder a un enlace de comunicaciones. Veamos una sencilla analogía. Piense en dos restaurantes, en uno de los cuales es necesario hacer reserva, mientras que en el otro no se necesita hacer reserva ni tampoco las admiten. Para comer en el restaurante que precisa reserva, tenemos que molestarnos en llamar por teléfono antes de salir de casa, pero al llegar allí, en principio, podremos sentarnos y pedir nuestro menú al camarero de manera inmediata. En el restaurante que no admite reservas, no tenemos que molestarnos en reservar mesa, pero al llegar allí, es posible que tengamos que esperar a que haya una mesa disponible, antes de poder sentarnos.

Las redes telefónicas tradicionales son ejemplos de redes de conmutación de circuitos. Considere lo que ocurre cuando una persona desea enviar información (de voz o faxsímil) a otra a través de una red telefónica. Antes de que el emisor pueda transmitir la información, la red debe establecer una conexión entre el emisor y el receptor. Se trata de una conexión *de buena fe* en la que los conmutadores existentes en la ruta entre el emisor y el receptor mantienen el estado de la conexión para dicha comunicación. En la jerga del campo de la telefonía, esta conexión se denomina **circuito**. Cuando la red establece el circuito, también reserva una velocidad de transmisión constante en los enlaces de la red (que representa una fracción de la capacidad de transmisión de cada enlace) para el tiempo que dure la conexión. Dado que ha reservado una determinada velocidad de transmisión para esta conexión emisor-receptor, el emisor puede transferir los datos al receptor a la velocidad constante *garantizada*.

La Figura 1.13 ilustra una red de conmutación de circuitos. En esta red, los cuatro conmutadores de circuitos están interconectados mediante cuatro enlaces. Cada uno de los enlaces tiene cuatro circuitos, por lo que cada enlace puede dar soporte a cuatro conexiones simultáneas. Cada uno de los hosts (por ejemplo, PCs y estaciones de trabajo) está conectado directamente a uno de los conmutadores. Cuando dos hosts desean comunicarse, la red establece una **conexión extremo a extremo** dedicada entre ellos. Por tanto, para que el host A se comunique con el host B, la red tiene que reservar en primer lugar un circuito en cada uno de los dos enlaces. En este ejemplo, la conexión extremo a extremo dedicada usa el segundo circuito en el primer enlace y el cuarto circuito en el segundo. Dado que cada enlace tiene cuatro circuitos, para cada enlace utilizado por la conexión extremo a extremo, esa conexión obtiene un cuarto de la capacidad total de transmisión del enlace mientras dure la conexión. Así, por ejemplo, si cada enlace entre dos conmutadores adyacentes tiene una velocidad de transmisión de 1 Mbps, entonces cada conexión extremo a extremo de conmutación de circuitos obtiene 250 kbps de velocidad de transmisión dedicada.

Por el contrario, veamos qué ocurre cuando un host desea enviar un paquete a otro host a través de una red de conmutación de paquetes, como por ejemplo Internet. Al igual que con la conmutación de circuitos, el paquete se transmite a través de una serie de enlaces de comunicaciones. Pero, a diferencia de la conmutación de circuitos, el paquete se envía a la red sin haber reservado ningún tipo de recurso de enlace. Si uno de los enlaces está congestionado porque otros paquetes tienen que ser transmitidos a través de él al mismo tiempo, entonces nuestro paquete tendrá que esperar en un buffer en el lado del enlace de transmisión correspondiente al emisor y sufrirá un retardo. Internet realiza el máximo esfuerzo para suministrar los paquetes a tiempo, pero no ofrece ningún tipo de garantía.

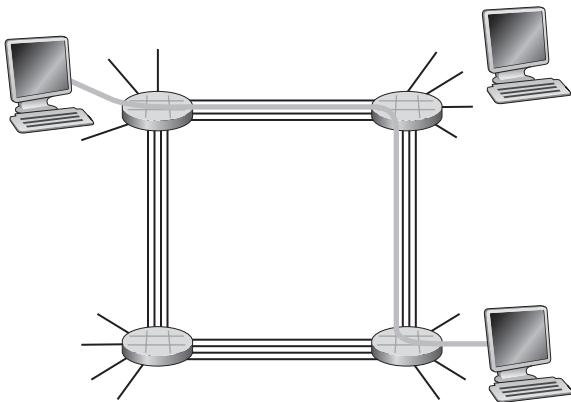


Figura 1.13 ♦ Una red simple de conmutación de circuitos, compuesta por cuatro conmutadores y cuatro enlaces.

Multiplexación en redes de conmutación de circuitos

Los circuitos de un enlace se implementan, bien mediante **multiplexación por división de frecuencia (FDM, Frequency-Division Multiplexing)** o bien mediante **multiplexación por división en el tiempo (TDM, Time-Division Multiplexing)**. Con FDM, el espectro de frecuencia de un enlace se reparte entre las conexiones establecidas a través del enlace. Específicamente, el enlace dedica una banda de frecuencias a cada conexión durante el tiempo que esta dure. En las redes telefónicas, esta banda de frecuencias normalmente tiene un ancho de 4 kHz (es decir, 4.000 hercios o 4.000 ciclos por segundo). El ancho de esta banda se denomina, como cabría esperar, **ancho de banda**. Las estaciones de radio FM también emplean la multiplexación FDM para compartir el espectro de frecuencias entre 88 MHz y 108 MHz, teniendo cada estación asignada una banda de frecuencias específica.

En un enlace TDM, el tiempo se divide en marcos de duración fija y cada marco se divide en un número fijo de particiones. Cuando la red establece una conexión a través de un enlace, la red dedica una partición de cada marco a dicha conexión. Estas particiones están reservadas para uso exclusivo de dicha conexión, habiendo una partición disponible (en cada marco) para transmitir los datos de esa conexión.

La Figura 1.14 ilustra las multiplexaciones FDM y TDM para un enlace de red específico que da soporte a cuatro circuitos. En el caso de la multiplexación por división de frecuencia, el dominio de frecuencia se segmenta en cuatro bandas, siendo el ancho de banda de cada una de ellas de 4 kHz. En el caso de la multiplexación TDM, el dominio del tiempo se divide en marcos, conteniendo cada uno de ellos cuatro particiones. A cada circuito se le asigna la misma partición dedicada dentro de los marcos, de forma cíclica. En la multiplexación TDM, la velocidad de transmisión de un circuito es igual a la velocidad de marco multiplicada por el número de bits existentes en una partición. Por ejemplo, si el enlace transmite 8.000 marcos por segundo y cada partición consta de 8 bits, entonces la velocidad de transmisión de un circuito es igual a 64 kbps.

Los partidarios de la tecnología de conmutación de paquetes siempre han argumentado que la conmutación de circuitos desperdicia recursos, porque los circuitos dedicados no se usan para nada durante los **periodos de inactividad**. Por ejemplo, cuando una persona deja de hablar durante una llamada telefónica, los recursos de red inactivos (bandas de frecuencia o particiones temporales en los enlaces situados a lo largo de la ruta de la conexión) no pueden ser empleados por otras conexiones en curso. Otro ejemplo de cómo estos recursos pueden ser infravalorados sería un radiólogo que empleara una red de conmutación de circuitos para acceder remotamente a una serie de radiografías de rayos X. El radiólogo establece una conexión, solicita una imagen, la contempla y luego solicita otra imagen. Los recursos de la red están asignados a la conexión, pero no se

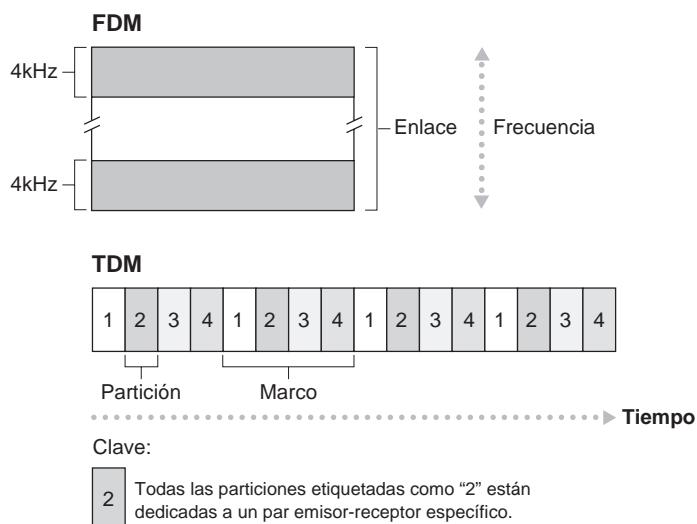


Figura 1.14 ♦ Con FDM, cada circuito obtiene de forma continua una fracción del ancho de banda. Con TDM, cada circuito dispone de todo el ancho de banda periódicamente durante breves intervalos de tiempo (es decir, durante las particiones).

utilizan (es decir, se desperdician) durante el tiempo que el radiólogo contempla las imágenes. Los partidarios de la conmutación de paquetes también disfrutan apuntando que el establecimiento de circuitos extremo a extremo y la reserva de capacidad de transmisión extremo a extremo son procesos complicados, que requieren el uso de software de señalización complejo para coordinar el funcionamiento de los conmutadores a lo largo de la ruta extremo a extremo.

Antes de terminar con esta exposición acerca de la conmutación de circuitos, vamos a ver un ejemplo numérico que debería arrojar más luz sobre este tema. Consideremos el tiempo que se tarda en enviar un archivo de 640.000 bits desde el host A al host B a través de una red de conmutación de circuitos. Supongamos que todos los enlaces de la red utilizan multiplexación TDM con 24 particiones y tienen una velocidad de bit de 1,536 Mbps. Supongamos también que se tardan 500 milisegundos en establecer el circuito extremo a extremo antes de que el host A pueda comenzar a transmitir el archivo. ¿Cuánto tiempo se tarda en transmitir el archivo? La velocidad de transmisión de cada circuito es $(1,536 \text{ Mbps})/24 = 64 \text{ kbps}$, por lo que se precisan $(640.000 \text{ bits})/(64 \text{ kbps}) = 10 \text{ segundos}$ para transmitir el archivo. A estos 10 segundos tenemos que sumarles el tiempo de establecimiento del circuito, lo que da como resultado 10,5 segundos de tiempo total de transmisión del archivo. Observe que el tiempo de transmisión es independiente del número de enlaces: el tiempo de transmisión será 10 segundos independientemente de que el circuito extremo a extremo pase a través de un enlace o de cien enlaces. (El retardo real extremo a extremo también incluye un retardo de propagación; véase la Sección 1.4.).

Comparación entre la conmutación de paquetes y la conmutación de circuitos

Ahora que ya hemos descrito las tecnologías de conmutación de circuitos y de paquetes, vamos a pasar a compararlas. Los detractores de la tecnología de conmutación de paquetes a menudo han argumentado que esta tecnología no es adecuada para los servicios en tiempo real, como por ejemplo las llamadas telefónicas y las videoconferencias, porque sus retardos extremo a extremo son variables e impredecibles (a causa principalmente de que los retardos de cola de los paquetes son variables e impredecibles). Por otro lado, los partidarios de la conmutación de paquetes argumentan que (1) ofrece una mejor compartición de la capacidad de transmisión existente que

la tecnología de conmutación de circuitos y (2) es más sencilla, más eficiente y menos cara de implementar que la conmutación de circuitos. Puede encontrar una comparación interesante de la conmutación de paquetes y la conmutación de circuitos en [Molinero-Fernández 2002]. En términos generales, las personas a las que no les gusta reservar mesa en un restaurante prefieren la conmutación de paquetes a la conmutación de circuitos.

¿Por qué es más eficiente la conmutación de paquetes? Veamos un ejemplo sencillo. Suponga que varios usuarios comparten un enlace de 1 Mbps. Suponga también que cada usuario alterna entre períodos de actividad (cuando genera datos a una velocidad constante de 100 kbps) y períodos de inactividad (cuando no genera datos). Además, suponga que un usuario solo está activo un 10 por ciento del tiempo (y está inactivo tomando café durante el 90 por ciento del tiempo restante). Con la tecnología de conmutación de circuitos, tienen que *reservarse para cada usuario* 100 kbps en todo momento. Por ejemplo, en una red de conmutación de circuitos con multiplexación TDM, si un marco de un segundo se divide en 10 particiones de 100 ms, entonces cada usuario tendría asignada una partición por marco.

Por tanto, el enlace de conmutación de circuitos solo podrá dar soporte a 10 (= 1 Mbps/100 kbps) usuarios simultáneamente. En el caso de utilizar la conmutación de paquetes, la probabilidad de que un determinado usuario esté activo es 0,1 (es decir, del 10 por ciento). Si hay 35 usuarios, la probabilidad de que 11 o más usuarios estén activos simultáneamente es aproximadamente igual a 0,0004. (El Problema de Repaso R8 indica cómo se obtiene esta probabilidad.) Cuando hay 10 o menos usuarios activos a la vez (lo que ocurre con una probabilidad del 0,9996), la velocidad acumulada de llegada de los datos es menor o igual a 1 Mbps, la velocidad de salida del enlace. Por tanto, cuando el número de usuarios activos es 10 o menos, los paquetes fluyen a través del enlace prácticamente sin retardo, como en el caso de la tecnología de conmutación de circuitos. Cuando hay más de 10 usuarios activos simultáneamente, entonces la velocidad acumulada de llegada de paquetes excede la capacidad de salida del enlace y la cola de salida comenzará a crecer. (Continúa creciendo hasta que la velocidad acumulada de entrada cae por debajo de 1 Mbps, punto en el que la longitud de la cola comenzará a disminuir.) Puesto que la probabilidad de que haya más de 10 usuarios conectados a la vez es muy baja en este ejemplo, la conmutación de paquetes proporciona prácticamente el mismo rendimiento que la conmutación de circuitos, *pero lo hace permitiendo que haya un número de usuarios más de tres veces superior*.

Consideremos ahora otro ejemplo sencillo. Suponga que hay 10 usuarios y que de repente un usuario genera 1.000 paquetes de 1.000 bits, mientras que los usuarios restantes permanecen inactivos y no generan paquetes. Con la tecnología de conmutación de circuitos con multiplexación TDM con 10 particiones por marco y con cada partición formada por 1.000 bits, el usuario activo solo puede emplear su partición por marco para transmitir los datos, mientras que las restantes nueve particiones de cada marco permanecen inactivas. Transcurrirán 10 segundos antes de que el millón de bits de datos del usuario activo hayan sido transmitidos. Sin embargo, con la conmutación de paquetes, el usuario activo puede enviar de forma continuada sus paquetes a la velocidad máxima del enlace de 1 Mbps, ya que no hay ningún otro usuario generando paquetes que tengan que ser multiplexados con los paquetes del usuario activo. En este caso, todos los datos del usuario activo se transmitirán en un segundo.

Los ejemplos anteriores ilustran dos casos en los que el rendimiento de la tecnología de conmutación de paquetes puede resultar superior al de la conmutación de circuitos. También ha quedado patente la diferencia crucial entre las dos formas de compartir la velocidad de transmisión del enlace entre varios flujos de datos. La conmutación de circuitos preasigna el uso del enlace de transmisión independientemente de la demanda, con lo que el tiempo de enlace asignado, pero innecesario, se desperdicia. Por el contrario, la conmutación de paquetes asigna el uso del enlace *bajo demanda*. La capacidad de transmisión del enlace se compartirá paquete a paquete solo entre aquellos usuarios que tengan paquetes que transmitir a través del enlace.

Aunque la conmutación de circuitos y la de paquetes son ambas prevalentes en las actuales redes de telecomunicaciones, se está tendiendo claramente hacia las redes de conmutación de paquetes. Incluso muchas de las redes de telefonía de conmutación de circuitos actuales se están migrando

lentamente a redes de conmutación de paquetes. En particular, las redes telefónicas suelen emplear la conmutación de paquetes en la parte internacional de las llamadas, que es la más cara.

1.3.3 Una red de redes

Anteriormente hemos visto que los sistemas terminales (PC de usuario, teléfonos inteligentes, servidores web, servidores de correo electrónico, etc.) se conectan a Internet a través de un ISP de acceso. El ISP de acceso puede proporcionar conectividad cableada o inalámbrica, mediante una amplia variedad de tecnologías de acceso, entre las que se incluyen DSL, cable, FTTH, Wi-Fi y la tecnología celular. Observe que el ISP de acceso no tiene por qué ser una compañía de telecomunicaciones o de cable: puede ser, por ejemplo, una universidad (que proporciona acceso a Internet a los estudiantes, al personal y a las facultades) o una empresa (que proporciona acceso a sus estudiantes, empleados y profesores) o una empresa (que proporciona acceso a su personal). Pero la conexión de los usuarios finales y de los proveedores de contenido a un ISP de acceso es solo una pequeña parte del problema de conectar los miles de millones de sistemas terminales que conforman Internet. Para completar este puzzle, los propios ISP de acceso deben interconectarse. Esto se hace creando una *red de redes* —comprender esta frase es la clave para entender Internet.

A lo largo de los años, la red de redes que es Internet ha evolucionado hasta formar una estructura muy compleja. Buena parte de esta evolución está dictada por razones económicas y políticas nacionales, en vez de por consideraciones de rendimiento. Para comprender la estructura de red actual de Internet, vamos a construir incrementalmente una serie de estructuras de red, siendo cada una de ellas una mejor aproximación a la Internet compleja que tenemos hoy en día. Recuerde que el objetivo fundamental es interconectar los ISP de acceso de modo que todos los sistemas terminales puedan intercambiar paquetes. Un enfoque simplista consistiría en que cada ISP de acceso se conectara *directamente* con todos los restantes ISP de acceso. Ese diseño en forma de malla resulta, por supuesto, demasiado costoso para los ISP de acceso, ya que requeriría que cada uno de ellos dispusiera de un enlace de comunicación separado con cada uno de los restantes cientos de miles de ISP de acceso existentes en el mundo.

Nuestra primera estructura de red, a la que llamaremos *Estructura de Red 1*, interconecta todos los ISP de acceso mediante un único *ISP global de tránsito*. Nuestro (imaginario) ISP global de tránsito sería una red de routers y enlaces de comunicaciones que no solo cubriría todo el planeta, sino que dispondría además de al menos un router cerca de cada uno de los cientos de miles de ISP de acceso. Por supuesto, sería muy costoso para el ISP global construir una red tan extensa. Para ser rentable, tendría por supuesto que cobrar por la conectividad a cada uno de los ISP de acceso, debiendo ese precio depender (aunque no necesariamente de forma directamente proporcional) de la cantidad de tráfico que el ISP de acceso intercambie con el ISP global. Puesto que el ISP de acceso paga al ISP global de tránsito, decimos que el ISP de acceso es un **cliente** y que el ISP global de tránsito es un **proveedor**.

Ahora, si alguna empresa construye y opera de forma rentable un ISP global de tránsito, entonces resulta natural que otras empresas construyan sus propios ISP globales de tránsito y compitan con el ISP global de tránsito original. Esto conduce a la *Estructura de Red 2*, que está formada por los cientos de miles de ISP de acceso y *múltiples* ISP globales de tránsito. Los ISP de acceso prefieren, por supuesto, la Estructura de Red 2 a la Estructura de Red 1, dado que ahora pueden elegir entre los proveedores globales de tránsito competidores, en función de sus precios y de los servicios que ofrezcan. Observe, sin embargo, que los propios ISP globales de tránsito deben interconectarse unos con otros: en caso contrario, los ISP de acceso conectados a uno de los proveedores globales de tránsito no serían capaces de comunicarse con los ISP de acceso conectados a los restantes proveedores globales de tránsito.

La Estructura de red 2 que acabamos de describir es una jerarquía en dos niveles, en la que los proveedores globales de tránsito residen en el nivel superior y los ISP de acceso, en el inferior. Esta estructura presupone que los ISP globales de tránsito no solo son capaces de estar cerca de cada

uno de los ISP de acceso, sino que también encuentran económicamente deseable el hacerlo. En realidad, aunque algunos ISP tienen una impresionante cobertura global y se conectan directamente con muchos ISP de acceso, no hay ningún ISP que tenga presencia en todas y cada una de las ciudades del mundo. En lugar de ello, en cualquier región determinada, puede haber un **ISP regional** al que se conecten los ISP de acceso de esa región. Cada ISP regional se conecta entonces con los **ISP de nivel 1**. Los ISP de nivel 1 son similares a nuestros (imaginarios) ISP globales de tránsito; pero los ISP de nivel 1, que sí que existen, no tienen presencia en todas las ciudades del mundo. Existe aproximadamente una docena de ISP de nivel 1, incluyendo Level 3 Communications, AT&T, Sprint y NTT. Resulta interesante observar que no hay ningún organismo que otorgue oficialmente el estatus de nivel 1; como dice el refrán, si necesitas preguntar si eres miembro de un grupo, probablemente no lo eres.

Volviendo a esta red de redes, no solo hay múltiples ISP competidores de nivel 1, sino que también puede haber múltiples ISP regionales competidores en una determinada región. En dicha jerarquía, cada ISP de acceso paga al ISP regional al que está conectado, y cada ISP regional paga al ISP de nivel 1 con el que se conecta. (Un ISP de acceso también puede conectarse directamente a un ISP de nivel 1, en cuyo caso le paga a él.) Por tanto, hay una relación cliente-proveedor en cada nivel de la jerarquía. Observe que los ISP de nivel 1 no pagan a nadie, ya que se encuentran en lo alto de la jerarquía. Para complicar las cosas todavía más, en algunas regiones puede haber un ISP regional de mayor tamaño (que posiblemente abarque un país completo) al que se conectan los ISP regionales más pequeños de esa región; el ISP regional de mayor tamaño se conecta a su vez a un ISP de nivel 1. Por ejemplo, en China existen ISP de acceso en cada ciudad, que se conectan a ISP provinciales, que a su vez se conectan a ISP nacionales, que finalmente se conectan a ISP de nivel 1 [Tian 2012]. A esta jerarquía multinivel, que sigue siendo tan solo una aproximación burda de la Internet actual, la denominamos *Estructura de red 3*.

Para construir una red que se asemeje más a la Internet de hoy en día, necesitamos añadir a la Estructura de Red 3 jerárquica los puntos de presencia (PoP, *Point of Presence*), la multidomiciliación (*multihoming*), la conexión entre pares (*peering*) y los puntos de intercambio Internet (IXP, *Internet eXchange Point*). Los PoP existen en todos los niveles de la jerarquía, salvo en el nivel inferior (ISP de acceso). Un PoP es, simplemente, un grupo de uno o más routers (en una misma ubicación) de la red del proveedor, a través de los cuales los ISP clientes pueden conectarse al ISP proveedor. Para conectarse al PoP del proveedor, la red del cliente puede arrendar un enlace de alta velocidad a un proveedor de telecomunicaciones cualquiera, con el fin de conectar directamente uno de sus routers a un router del PoP. Cualquier ISP (salvo los ISP de nivel 1) puede optar por la multidomiciliación, es decir, por conectarse a dos o más ISP proveedores. Así, por ejemplo, un ISP de acceso puede multidomiciliarse con dos ISP regionales, o con dos ISP regionales y un ISP de nivel 1. De forma similar, un ISP regional puede multidomiciliarse con varios ISP de nivel 1. Cuando un ISP opta por la multidomiciliación, puede continuar enviando y recibiendo paquetes a través de Internet incluso si uno de sus proveedores sufre un fallo.

Como hemos visto, los ISP clientes pagan a su ISP proveedor para obtener una interconectividad global a través de Internet. La cantidad que un ISP cliente paga a un ISP proveedor refleja la cantidad de tráfico que intercambia con el proveedor. Para reducir estos costes, dos ISP próximos, situados en el mismo nivel de la jerarquía, pueden establecer una conexión entre pares, es decir, pueden conectar directamente sus redes, de modo que todo el tráfico que se intercambien pase por la conexión directa, en lugar de a través de intermediarios situados aguas arriba. Cuando dos ISP efectúan una conexión entre pares, suele ser libre de cargo, es decir, ninguno de los dos ISP paga al otro. Como hemos dicho anteriormente, los ISP de nivel 1 también establecen conexiones entre pares, libres de cargo. En [Van der Berg 2008] podrá encontrar una explicación comprensible de las conexiones entre pares y las relaciones cliente-proveedor. Siguiendo estos mismos principios, una empresa puede crear un **punto de intercambio Internet** (IXP, *Internet Exchange Point*), que es un punto de reunión en el que múltiples ISP pueden establecer conexiones entre pares. El IXP se localiza normalmente en su propio edificio, con sus propios comutadores [Ager 2012]. Hoy en

día, en Internet hay más de 400 IXP [IXP List 2016]. A este ecosistema —compuesto por los ISP de acceso, los ISP regionales, los ISP de nivel 1, los PoP, la multidomiciliación, las conexiones entre pares y los IXP— lo denominamos *Estructura de red 4*.

Con esto llegamos, finalmente, a la *Estructura de red 5*, que describe la Internet de hoy en día. La Estructura de Red 5, ilustrada en la Figura 1.15, se construye sobre la Estructura de red 4, añadiendo las **redes de proveedores de contenido**. Google es, actualmente, uno de los ejemplos punteros de dichas redes de provisión de contenidos. En el momento de escribir estas líneas, se estima que Google dispone de 50–100 centros de datos por Norteamérica, Europa, Asia, Sudamérica y Australia. Algunos de estos centros de datos albergan más de cien mil servidores, mientras que otros son más pequeños, con solo unos centenares de servidores. Todos los centros de datos de Google están interconectados a través de la propia red TCP/IP privada de Google, que abarca todo el planeta pero está separada de la Internet pública. Resulta importante destacar que la red privada de Google solo transporta tráfico hacia/desde los servidores de Google. Como se muestra en la Figura 1.15, la red privada de Google trata de “puentejar” los niveles superiores de Internet, estableciendo conexiones entre pares (libres de cargo) con los ISP de nivel inferior, bien conectándose directamente a ellos o bien conectándose con ellos en algún IXP [Labovitz 2010]. Sin embargo, puesto que a muchos ISP de acceso solo se puede llegar pasando por redes de nivel 1, la red de Google también se conecta con los ISP de nivel 1 y paga a esos ISP por el tráfico que intercambia con ellos. Al crear su propia red, un proveedor de contenido no solo reduce sus pagos a los ISP de nivel superior, sino que también tiene un mayor control sobre el modo en que se prestan sus servicios en último término a los usuarios finales. La infraestructura de la red de Google se describe con mayor detalle en la Sección 2.6.

En resumen, la Internet de hoy en día —una red de redes— es compleja, estando compuesta por aproximadamente una docena de ISP de nivel 1 y cientos de miles de ISP de nivel inferior. La cobertura geográfica de los ISP es muy variable, habiendo algunos que abarcan múltiples continentes y océanos, mientras que otros están limitados a pequeñas regiones. Los ISP de nivel inferior se conectan con los de nivel superior y los de nivel superior se interconectan entre sí. Los usuarios y los proveedores de contenido son clientes de los ISP de nivel inferior y estos son clientes de los ISP de nivel superior. En los últimos años, los principales proveedores de contenidos han creado también sus propias redes y se conectan directamente, siempre que pueden, a los ISP de nivel inferior.

1.4 Retardos, pérdidas y tasa de transferencia en las redes de conmutación de paquetes

En la Sección 1.1 hemos dicho que Internet puede verse como una infraestructura que proporciona servicios a aplicaciones distribuidas que se ejecutan en sistemas terminales. Idealmente, desearíamos que los servicios de Internet pudieran transportar tantos datos como quisieramos entre cualesquiera dos sistemas terminales de forma instantánea y sin que tuviera lugar ninguna pérdida de datos. Evidentemente, en la realidad, este objetivo es inalcanzable, ya que necesariamente las redes de computadoras restringen la tasa de transferencia (la cantidad de datos por segundo que pueden transmitir) entre sistemas terminales, introducen retardos entre los sistemas terminales y pueden, de hecho, perder paquetes. Por una parte, es una pena que las leyes de la Física introduzcan retardos y pérdidas, así como que restrinjan las tasas de transferencia, pero, por otra parte, puesto que las redes de computadoras presentan estos problemas, existen muchas cuestiones interesantes relacionadas con cómo abordarlos —¡más que suficientes como para llenar un curso sobre redes de computadoras y para motivar miles de tesis doctorales! En esta sección comenzaremos examinando y cuantificando los retardos, las pérdidas y la tasa de transferencia en las redes de computadoras.

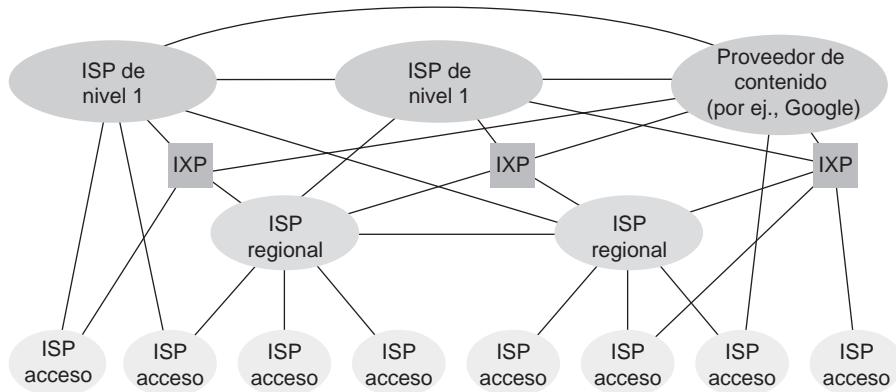


Figura 1.15 ♦ Interconexión de los ISP.

1.4.1 El retardo en las redes de conmutación de paquetes

Recordemos que los paquetes se inicián en un host (el origen), atraviesan una serie de routers y terminan su viaje en otro host (el destino). Cuando un paquete viaja de un nodo (host o router) al siguiente nodo (host o router) a lo largo de esa ruta, el paquete sufre varios tipos de retraso en cada uno de los nodos de dicha ruta. Los más importantes de estos retardos son: el **retardo de procesamiento nodal**, el **retardo de cola**, el **retardo de transmisión** y el **retardo de propagación**; todos estos retardos se suman para proporcionar el **retardo nodal total**. El rendimiento de muchas aplicaciones de Internet —como las de búsqueda, exploración web, correo electrónico, mapas, mensajería instantánea y voz sobre IP— se ve seriamente afectado por los retardos de red. Para adquirir un conocimiento profundo de la tecnología de conmutación de paquetes y de las redes de computadoras, es preciso comprender la naturaleza e importancia de estos retardos.

Tipos de retraso

Utilizaremos la Figura 1.16 para explorar estos retardos. Como parte de la ruta extremo a extremo entre el origen y el destino, un paquete se envía desde el nodo situado aguas arriba a través del router A y hasta el router B. Nuestro objetivo es caracterizar el retraso nodal en el router A. Observe que el router A dispone de un enlace de salida que lleva hasta el router B. Este enlace está precedido por una cola (o *buffer*). Cuando el paquete llega al router A procedente del nodo situado aguas arriba, el router A examina la cabecera del paquete para determinar cuál es el enlace de salida apropiado para el paquete y luego dirige dicho paquete a ese enlace. En este ejemplo, el enlace de salida para el paquete es el enlace que lleva hasta el router B. Un paquete puede transmitirse a

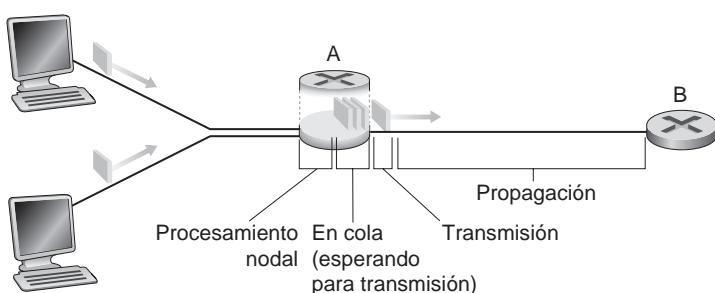


Figura 1.16 ♦ Retardo nodal en el router A.

través de un enlace solo si actualmente no se está transmitiendo ningún otro paquete a través de él y si no hay otros paquetes que le precedan en la cola; si el enlace está ocupado actualmente o si existen otros paquetes en la cola esperando para ese enlace, entonces el paquete recién llegado tendrá que ponerse a la cola.

Retardo de procesamiento

El tiempo requerido para examinar la cabecera del paquete y determinar dónde hay que enviarlo es parte del **retardo de procesamiento**. El retardo de procesamiento puede también incluir otros factores, como el tiempo necesario para comprobar los errores de nivel de bit del paquete que se hayan producido al transmitir los bits del paquete desde el nodo situado aguas arriba hasta el router A. Los retardos de procesamiento en los routers de alta velocidad suelen ser del orden de los microsegundos o menores. Una vez efectuado el procesamiento nodal, el router dirige el paquete a la cola situada antes del enlace que lleva al router B. (En el Capítulo 4 estudiaremos los detalles acerca de cómo funciona un router.)

Retardo de cola

En la cola, el paquete experimenta un **retardo de cola** mientras espera para ser transmitido a través del enlace. La duración del retardo de cola para un determinado paquete dependerá del número de paquetes que hayan llegado antes a la cola y que están esperando para ser transmitidos por el enlace. Si la cola está vacía y no se está transmitiendo ningún paquete actualmente, entonces el retardo de cola de nuestro paquete será cero. Por el contrario, si hay mucho tráfico y muchos paquetes están esperando también para ser transmitidos, el retardo de cola será grande. Como veremos en breve, el número de paquetes con que un paquete entrante puede esperar encontrarse es una función de la intensidad y de la naturaleza del tráfico que llega a la cola. En la práctica, los retardos de cola pueden ser del orden de microsegundos a milisegundos.

Retardo de transmisión

Suponiendo que los paquetes se transmiten de manera que el primero que llega es el primero que sale, como suele ser común en las redes de conmutación de paquetes, nuestro paquete solo puede ser transmitido después de que todos los paquetes que hayan llegado antes que él hayan sido transmitidos. Sea la longitud del paquete igual a L bits y la velocidad de transmisión del enlace del router A hasta el router B igual a R bits/segundo. Por ejemplo, para un enlace Ethernet a 10 Mbps, la velocidad es $R = 10$ Mbps; para un enlace Ethernet a 100 Mbps, la velocidad será $R = 100$ Mbps. El **retardo de transmisión** será igual a L/R . Este es el tiempo necesario para introducir (es decir, transmitir) todos los bits del paquete en el enlace. Normalmente, en la práctica, los retardos de transmisión son del orden de microsegundos a milisegundos.

Retardo de propagación

Una vez que un bit ha entrado en el enlace, tiene que propagarse hasta el router B. El tiempo necesario para propagarse desde el principio del enlace hasta el router B es el **retardo de propagación**. El bit se propaga a la velocidad de propagación del enlace, que depende del medio físico del enlace (es decir, de que el medio sea fibra óptica, cable de cobre de par trenzado, etc.) y está comprendido en el rango entre

$$2 \cdot 10^8 \text{ m/s} \text{ y } 3 \cdot 10^8 \text{ m/s}$$

que es igual, o ligeramente inferior, a la velocidad de la luz. El retardo de propagación es igual a la distancia entre dos routers dividida por la velocidad de propagación. Es decir, el retardo de

propagación es igual a d/s , donde d es la distancia entre el router A y el router B y s es la velocidad de propagación del enlace. Una vez que el último bit del paquete se ha propagado hasta el nodo B, este y todos los bits anteriores del paquete se almacenan en el router B. A continuación, el proceso continúa, encargándose el router B de llevar a cabo el reenvío. En las redes de área extensa, los retardos de propagación son del orden de milisegundos.



Exploración de los retardos de propagación y transmisión

Comparación de los retardos de transmisión y de propagación

Los recién llegados al campo de las redes de computadoras en ocasiones tienen dificultades para comprender la diferencia entre el retardo de transmisión y el de propagación. Esta diferencia es sutil, pero importante. El retardo de transmisión es la cantidad de tiempo necesario para que el router saque fuera el paquete; es una función de la longitud del paquete y de la velocidad de transmisión del enlace, pero no tiene nada que ver con la distancia existente entre los dos routers. Por el contrario, el retardo de propagación es el tiempo que tarda un bit en propagarse de un router al siguiente; es una función de la distancia entre los dos routers, pero no tiene nada que ver con la longitud del paquete ni con la velocidad de transmisión del enlace.

Veamos una analogía que nos va a permitir clarificar los conceptos de retardo de transmisión y de retardo de propagación. Imagine una autopista en la que hay un puesto de peaje cada 100 kilómetros, como se muestra en la Figura 1.17. Podemos imaginar que los segmentos de autopista entre peajes son los enlaces y las casetas de peaje son los routers. Suponga que los automóviles viajan (es decir, se propagan) por la autopista a una velocidad de 100 km/hora (es decir, cuando un coche sale de un peaje, instantáneamente acelera hasta adquirir la velocidad de 100 km/hora y la mantiene entre puestos de peaje sucesivos). Supongamos ahora que hay 10 coches que viajan en caravana unos detrás de otros en un orden fijo. Podemos pensar que cada coche es un bit y que la caravana es un paquete. Supongamos también que cada puesto de peaje da servicio (es decir, transmite) a los coches a una velocidad de un coche cada 12 segundos y que es tarde por la noche, por lo que en la autopista solo se encuentra nuestra caravana de coches. Por último, supongamos que cuando el primer coche de la caravana llega a un peaje, espera en la entrada hasta que los otros nueve coches han llegado y se han detenido detrás de él (es decir, la caravana completa tiene que almacenarse en el peaje antes de poder ser reenviada). El tiempo necesario para que el peaje deje pasar a la caravana completa hacia el siguiente tramo de autopista es igual a $(10 \text{ coches})/(5 \text{ coches/minuto}) = 2 \text{ minutos}$. Este tiempo es análogo al retardo de transmisión de un router. El tiempo necesario para que un coche se desplace desde la salida del peaje hasta el siguiente puesto de peaje es $100 \text{ km}/(100 \text{ km/hora}) = 1 \text{ hora}$. Este tiempo es análogo al retardo de propagación. Por tanto, el tiempo que transcurre desde que la caravana queda colocada delante de un peaje hasta que vuelve a quedar colocada delante del siguiente peaje es la suma del tiempo de transmisión y el tiempo de propagación (en este caso, dicho tiempo será igual a 62 minutos).

Profundicemos un poco más en esta analogía. ¿Qué ocurriría si el tiempo de servicio del puesto de peaje para una caravana fuera mayor que el tiempo que tarda un coche en viajar de un peaje al siguiente? Por ejemplo, supongamos que los coches viajan a una velocidad de 1.000 km/hora y que los peajes operan a una velocidad de un coche por minuto. Entonces, el retardo correspondiente al hecho de desplazarse entre dos puestos de peaje será de 6 minutos y el tiempo que tarda el puesto de peaje en dar servicio a una caravana será de 10 minutos. En este caso, los primeros coches de la

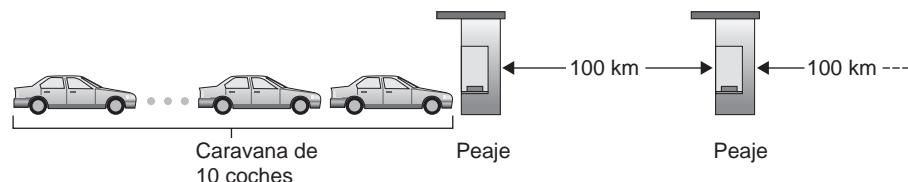


Figura 1.17 ♦ Analogía de la caravana.

caravana llegarán al segundo puesto de peaje antes de que los últimos coches de la caravana hayan salido del primer peaje. Esta situación también se produce en las redes de conmutación de paquetes: los primeros bits de un paquete pueden llegar a un router mientras que muchos de los bits restantes del paquete todavía están esperando a ser transmitidos por el router anterior.

Si una imagen vale más que mil palabras, entonces una animación vale más que un millón de palabras. En el sitio web de este libro de texto se proporciona un applet Java interactivo que ilustra y compara convenientemente los retardos de transmisión y de propagación. Animamos a los lectores a visitar este applet. [Smith 2009] también proporciona una explicación bastante comprensible de los retardos de propagación, de cola y de transmisión.

Sean d_{proc} , d_{cola} , d_{trans} y d_{prop} los retardos de procesamiento, de cola, de transmisión y de propagación, respectivamente. Entonces el retardo nodal total estará dado por:

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{cola}} + d_{\text{trans}} + d_{\text{prop}}$$

Las contribuciones de estos componentes de retardo pueden variar significativamente. Por ejemplo, d_{prop} puede ser despreciable (digamos que un par de microsegundos) para un enlace que conecte dos routers del mismo campus universitario; sin embargo, d_{prop} será de cientos de milisegundos para dos routers interconectados mediante un enlace vía satélite geoestacionario y puede llegar a ser el término dominante en d_{nodal} . Del mismo modo, d_{trans} puede ser despreciable o significativo. Su contribución normalmente es despreciable para velocidades de transmisión de 10 Mbps y superiores (por ejemplo, para las redes LAN); sin embargo, puede ser igual a cientos de milisegundos para paquetes grandes de Internet enviados a través de enlaces de acceso telefónico que usen modems de baja velocidad. El retardo de procesamiento, d_{proc} , suele ser despreciable; sin embargo, tiene una gran influencia sobre la tasa de transferencia máxima del router, que es la velocidad máxima a la que un router puede reenviar los paquetes.

1.4.2 Retardo de cola y pérdida de paquetes

El componente más complejo e interesante del retardo nodal es el retardo de cola, d_{cola} . De hecho, el retardo de cola es tan importante e interesante en las redes de computadoras que se han escrito miles de artículos y numerosos libros sobre él [Bertsekas 1991; Daigle 1991; Kleinrock 1975, Kleinrock 1976; Ross 1995]. Aquí solo vamos a abordarlo de forma intuitiva y panorámica; los lectores más curiosos pueden echar un vistazo a algunos de los libros que se ocupan de este tema (¡o incluso pueden escribir una tesis doctoral sobre el asunto!). A diferencia de los otros tres retardos (d_{proc} , d_{trans} y d_{prop}), el retardo de cola puede variar de un paquete a otro. Por ejemplo, si llegan 10 paquetes a una cola vacía al mismo tiempo, el primer paquete transmitido no sufrirá retardo de cola, mientras que el último paquete transmitido sufrirá un retardo de cola relativamente largo (mientras espera a que los restantes nueve paquetes sean transmitidos). Por tanto, al caracterizar el retardo de cola, suelen emplearse medidas estadísticas, como el retardo medio de cola, la varianza del retardo de cola y la probabilidad de que el retardo de cola exceda un cierto valor especificado.

¿En qué casos el retardo de cola es grande y en qué casos es insignificante? La respuesta a esta pregunta depende de la velocidad a la que llega el tráfico a la cola, de la velocidad de transmisión del enlace y de la naturaleza del tráfico entrante, es decir, de si el tráfico llega periódicamente o a ráfagas. Vamos a profundizar en este punto. Sea a la velocidad media a la que llegan los paquetes a la cola (a se expresa en paquetes/segundo). Recuerde que R es la velocidad de transmisión; es decir, es la velocidad (en bits/segundo) a la que los bits salen de la cola. Con el fin de simplificar, supongamos también que todos los paquetes constan de L bits. Entonces, la velocidad media a la que llegan los bits a la cola es igual a La bits/segundo. Supongamos por último que la cola es muy grande, por lo que podemos decir que puede almacenar un número infinito de bits. El cociente La/R , denominado intensidad de tráfico, suele desempeñar un papel importante a la hora de estimar la magnitud del retardo de cola. Si $La/R > 1$, entonces la velocidad media a la que los bits llegan a la cola excede la velocidad a la que los bits pueden ser transmitidos desde la cola. En esta desafortunada situación, la cola tenderá a aumentar sin límite ¡y el retardo de cola se aproximará a infinito! Por tanto, una de las

reglas de oro en la ingeniería de tráfico es: *diseñe su sistema de modo que la intensidad de tráfico no sea mayor que 1.*

Veamos ahora el caso en que $La/R \leq 1$. Aquí, la naturaleza del tráfico entrante influye sobre el retardo de cola. Por ejemplo, si los paquetes llegan periódicamente —es decir, si llega un paquete cada L/R segundos—, entonces todos los paquetes llegarán a una cola vacía y no habrá retardo de cola. Por el contrario, si los paquetes llegan a ráfagas pero de forma periódica, puede haber un retardo medio de cola significativo. Por ejemplo, supongamos que llegan simultáneamente N paquetes cada $(L/R)N$ segundos. En este caso, el primer paquete transmitido no tiene asociado ningún retardo de cola, el segundo paquete transmitido presentará un retardo de cola de L/R segundos y, de forma más general, *el n -ésimo paquete transmitido presentará un retardo de cola de $(n-1)L/R$ segundos.* Dejamos como ejercicio para el lector el cálculo del retardo medio de cola para este ejemplo.

Los dos ejemplos de llegada periódica de paquetes que acabamos de describir se corresponden con casos teóricos. Normalmente, *el proceso de llegada a una cola es aleatorio*; es decir, las llegadas no siguen ningún patrón y los paquetes quedan separados por períodos de tiempo aleatorios. En este caso más realista, la cantidad La/R normalmente no es suficiente para caracterizar completamente las estadísticas del retardo de cola. Aun así, resulta útil para tener una idea intuitiva de la magnitud del retardo de cola. En particular, si la intensidad de tráfico es próxima a cero, entonces las llegadas de paquetes serán pocas y estarán bastante espaciadas entre sí, por lo que será improbable que un paquete que llegue a la cola se encuentre con que hay otro paquete en la cola. Por tanto, el retardo medio de cola será próximo a cero. Por el contrario, cuando la intensidad de tráfico es próxima a 1, habrá intervalos de tiempo en los que la velocidad de llegada exceda a la capacidad de transmisión (a causa de las variaciones en la velocidad de llegada de los paquetes), por lo que se formará una cola durante estos períodos de tiempo; cuando la velocidad de llegada sea menor que la capacidad de transmisión, la longitud de la cola disminuirá. Sin embargo, a medida que la intensidad de tráfico se aproxime a 1, la longitud media de la cola será cada vez mayor. La dependencia cualitativa del retardo medio de cola con respecto a la intensidad de tráfico se muestra en la Figura 1.18.

Un aspecto importante de la Figura 1.18 es el hecho de que, cuando la intensidad de tráfico se aproxima a 1, el retardo medio de cola aumenta rápidamente. Un pequeño porcentaje de aumento en la intensidad dará lugar a un incremento porcentual muy grande del retardo. Es posible que el lector haya experimentado este fenómeno en una autopista. Si viaja regularmente por una autopista que habitualmente está congestionada, quiere decir que la intensidad de tráfico en esa autopista es próxima a 1. En el caso de que se produzca un suceso que dé lugar a una cantidad de tráfico solo ligeramente mayor que la usual, los retardos que se experimenten pueden llegar a ser enormes.

Con el fin de que entienda bien lo que son los retardos de cola, animamos de nuevo al lector a visitar el sitio web dedicado a este libro, donde se proporciona un applet de Java interactivo que

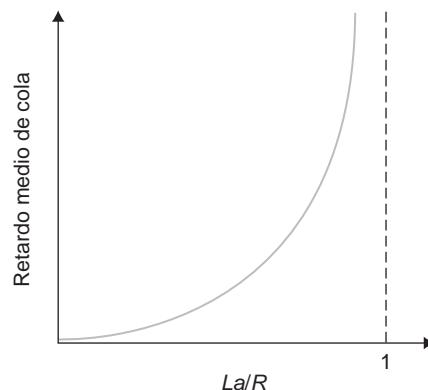


Figura 1.18 ♦ Dependencia del retardo medio de cola con respecto a la intensidad de tráfico.

muestra una cola. Si establece una velocidad de llegada de los paquetes lo suficientemente alta como para que la intensidad de tráfico sea mayor que 1, comprobará que la cola aumenta lentamente con el tiempo.

Pérdida de paquetes

En las explicaciones hasta ahora, hemos supuesto que la cola es capaz de almacenar un número infinito de paquetes. En la práctica, una cola para acceder a un enlace tiene una capacidad finita, aunque la capacidad de la cola depende fundamentalmente del diseño y del coste del router. Puesto que la capacidad de cola es finita, los retardos de los paquetes realmente no se aproximan a infinito a medida que la intensidad de tráfico se aproxima a 1. En su lugar, un paquete puede llegar y encontrarse con que la cola está llena. Al no tener sitio para almacenar dicho paquete, el router lo elimina; es decir, el paquete se pierde. Este desbordamiento de una cola puede verse también en el applet de Java, cuando la intensidad de tráfico es mayor que 1.

Desde el punto de vista de un sistema terminal, un paquete perdido es un paquete que ha sido transmitido al núcleo de la red pero que nunca sale de la red para llegar a su destino. El porcentaje de paquetes perdidos aumenta a medida que crece la intensidad de tráfico. Por tanto, el rendimiento de un nodo suele medirse no solo en función del retardo, sino también en función de la probabilidad de pérdida de paquetes. Como veremos en los siguientes capítulos, un paquete perdido puede retransmitirse de extremo a extremo para garantizar que todos los datos sean transferidos desde el origen hasta el destino.

1.4.3 Retardo extremo a extremo

Hasta el momento nos hemos centrado en el retardo nodal, es decir, el retardo en un único router. Ahora vamos a ocuparnos del retardo total entre el origen y el destino. Para entender este concepto, suponga que hay $N-1$ routers entre el host de origen y el host de destino. Suponga también, por el momento, que la red no está congestionada (por lo que los retardos de cola son despreciables), que el retardo de procesamiento en cada router y en el host de origen es d_{proc} , que la velocidad de transmisión de salida de cada router y del host de origen es de R bits/segundo y que el retardo de propagación en cada enlace es igual a d_{prop} . Los retardos nodales se suman para proporcionar el retardo extremo a extremo, luego

$$d_{\text{extremo-extremo}} = N(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}}) \quad (1.2)$$

donde, de nuevo, $d_{\text{trans}} = L/R$, siendo L el tamaño del paquete. Observe que la Ecuación 1.2 es una generalización de la Ecuación 1.1, en la que no se tenían en cuenta los retardos de procesamiento y de propagación. Dejamos para el lector el ejercicio de generalizar esta fórmula para el caso en que los retardos en los nodos sean diferentes y exista un retardo medio de cola en cada nodo.

Traceroute

Para ver el orden de magnitud del retardo extremo a extremo de una red de computadoras, podemos utilizar el programa Traceroute. Se trata de un programa simple que se puede ejecutar en cualquier host de Internet. Cuando el usuario especifica un nombre de host de destino, el programa del host de origen envía al destino varios paquetes especiales. A medida que estos paquetes se dirigen a su destino, pasan a través de una serie de routers. Cuando un router recibe uno de estos paquetes especiales, devuelve al origen un mensaje corto que contiene el nombre y la dirección del router.

Más concretamente, suponga que hay $N-1$ routers entre el origen y el destino. Entonces el origen enviará N paquetes especiales a la red, todos ellos dirigidos al destino final. Estos N paquetes especiales se marcan de 1 (el primero) a N (el último). Cuando el router n -ésimo recibe el paquete n -ésimo marcado como n , el router no reenvía el paquete hacia su destino, sino que devuelve un



Nota de video

Utilización de Traceroute para descubrir rutas de red y medir el retardo de la red.

mensaje al origen. Cuando el host de destino recibe el paquete N -ésimo, también devuelve un mensaje al origen. El origen registra el tiempo transcurrido entre el momento en que envió un paquete y el momento en que recibe el correspondiente mensaje de respuesta; también registra el nombre y la dirección del router (o del host de destino) que devuelve el mensaje. De esta forma, el origen puede reconstruir la ruta seguida por los paquetes que fluyen desde el origen hasta el destino, y puede también determinar los retardos de ida y vuelta para todos los routers que intervienen en el proceso. En la práctica, Traceroute repite el proceso que acabamos de describir tres veces, de modo que el origen realmente envía $3 \cdot N$ paquetes al destino. El documento RFC 1393 describe en detalle el programa Traceroute.

He aquí un ejemplo de la salida proporcionada por el programa Traceroute, en el que se ha trazado la ruta desde el host de origen `gaia.cs.umass.edu` (en la Universidad de Massachusetts) al host `cis.poly.edu` (en la Universidad Politécnica de Brooklyn). La salida consta de seis columnas: la primera de ellas contiene el valor n descrito anteriormente, es decir, el número del router a lo largo de la ruta; la segunda columna especifica el nombre del router; la tercera indica la dirección del router (con el formato `xxx.xxx.xxx.xxx`); las tres últimas columnas especifican los retardos de ida y vuelta correspondientes a los tres experimentos. Si el origen recibe menos de tres mensajes de cualquier router (debido a la pérdida de paquetes en la red), Traceroute incluye un asterisco justo después del número de router y proporciona menos de tres tiempos de ida y vuelta para dicho router.

```

1 cs-gw (128.119.240.254) 1.009 ms 0.899 ms 0.993 ms
2 128.119.3.154 (128.119.3.154) 0.931 ms 0.441 ms 0.651 ms
3 -border4-rt-gi-1-3.gw.umass.edu (128.119.2.194) 1.032 ms 0.484 ms 0.451 ms
4 -acrl-ge-2-1-0.Boston.cw.net (208.172.51.129) 10.006 ms 8.150 ms 8.460 ms
5 -agr4-loopback.NewYork.cw.net (206.24.194.104) 12.272 ms 14.344 ms 13.267 ms
6 -acr2-loopback.NewYork.cw.net (206.24.194.62) 13.225 ms 12.292 ms 12.148 ms
7 -pos10-2.core2.NewYork1.Level3.net (209.244.160.133) 12.218 ms 11.823 ms 11.793 ms
8 -gige9-1-52.hsipaccess1.NewYork1.Level3.net (64.159.17.39) 13.081 ms 11.556 ms 13.297 ms
9 -p0-0.polyu.bbnplanet.net (4.25.109.122) 12.716 ms 13.052 ms 12.786 ms
10 cis.poly.edu (128.238.32.126) 14.080 ms 13.035 ms 12.802 ms

```

Podemos ver en esta traza que existen nueve routers entre el origen y el destino. La mayor parte de estos routers tiene un nombre y todos ellos tienen direcciones. Por ejemplo, el nombre del Router 3 es `border4-rt-gi-1-3.gw.umass.edu` y su dirección es `128.119.2.194`. Si observamos los datos proporcionados para este mismo router, vemos que en la primera de las tres pruebas el retardo de ida y vuelta entre el origen y el router ha sido de 1,03 milisegundos. Los retardos de ida y vuelta para las dos pruebas siguientes han sido 0,48 y 0,45 milisegundos, respectivamente. Estos retardos de ida y vuelta contienen todos los retardos que acabamos de estudiar, incluyendo los retardos de transmisión, de propagación, de procesamiento del router y de cola. Puesto que el retardo de cola varía con el tiempo, el retardo de ida y vuelta del paquete n enviado al router n puede, en ocasiones, ser mayor que el retardo de ida y vuelta del paquete $n + 1$ enviado al router $n + 1$. De hecho, puede observar este fenómeno en el ejemplo anterior: ¡los retardos correspondientes al Router 6 son mayores que los correspondientes al Router 7!

¿Desea probar el programa Traceroute? Le recomendamos *vivamente* que visite el sitio <http://www.traceroute.org>, donde se proporciona una interfaz web a una extensa lista de orígenes para el trazado de rutas. Seleccione un origen y especifique el nombre de host de cualquier destino. El programa Traceroute hará entonces todo el trabajo. Hay disponibles diversos programas software gratuitos que proporcionan una interfaz gráfica para Traceroute; uno de nuestros programas favoritos es PingPlotter [PingPlotter 2016].

Retardos de los sistemas terminales, de las aplicaciones y otros

Además de los retardos de procesamiento, de transmisión y de propagación, en los sistemas terminales pueden existir retardos adicionales significativos. Por ejemplo, un sistema terminal que quiera transmitir un paquete a través de un medio compartido (por ejemplo, como sucede en un escenario que incluya WiFi o un módem por cable) puede retardar su transmisión *a propósito* como parte de su protocolo, para compartir el medio con otros sistemas terminales; en el Capítulo 6 analizaremos en detalle tales protocolos. Otro retardo importante es el retardo de empaquetamiento del medio, que aparece en las aplicaciones de Voz sobre IP (VoIP, *Voice-over-IP*). En VoIP, el lado emisor debe, en primer lugar, llenar un paquete con voz digitalizada codificada antes de pasar el paquete a Internet. El tiempo que se tarda en llenar un paquete (denominado retardo de empaquetamiento) puede ser significativo y puede repercutir en la calidad percibida por el usuario de una llamada VoIP. Este problema se abordará más detalladamente en uno de los problemas del final del capítulo.

No
entendí

1.4.4 Tasa de transferencia en las redes de computadoras

Además de los retardos y la pérdida de paquetes, otra medida crítica de rendimiento de las redes de computadoras es la tasa de transferencia de extremo a extremo. Para definir la tasa de transferencia, consideremos la transferencia de un archivo de gran tamaño desde el host A al host B a través de una red. Por ejemplo, esta transferencia podría consistir en transferir un clip de vídeo de gran tamaño desde un par (*peer*) a otro en un sistema de compartición de archivos P2P. La tasa de transferencia instantánea en cualquier instante de tiempo es la velocidad (en bits/segundo) a la que el host B recibe el archivo. (Muchas aplicaciones, incluyendo muchos sistemas de compartición de archivos P2P, muestran la tasa de transferencia instantánea durante las descargas en la interfaz de usuario; seguro que el lector ya se ha fijado anteriormente en este detalle.) Si el archivo consta de F bits y la transferencia dura T segundos hasta que el host B recibe los F bits, entonces la tasa media de transferencia del archivo es igual a F/T bits/segundo. En algunas aplicaciones, tales como la telefonía por Internet, es deseable tener un retardo pequeño y una tasa de transferencia instantánea que esté constantemente por encima de un cierto umbral (por ejemplo, por encima de 24 kbps para ciertas aplicaciones de telefonía por Internet y por encima de 256 kbps para las aplicaciones de vídeo en tiempo real). Para otras aplicaciones, entre las que se incluyen aquéllas que implican la transferencia de archivos, el retardo no es crítico, pero es deseable que la tasa de transferencia sea lo más alta posible.

Con el fin de comprender mejor el importante concepto de tasa de transferencia, vamos a ver algunos ejemplos. La Figura 1.19(a) muestra dos sistemas terminales, un servidor y un cliente, conectados mediante dos enlaces de comunicaciones y un router. Consideremos la tasa de transferencia para transmitir un archivo desde el servidor al cliente. Sea R_s la velocidad del enlace entre el servidor y el router, y sea R_c la velocidad del enlace entre el router y el cliente. Supongamos que los únicos bits que están siendo enviados a través de la red son los que van desde el servidor al cliente. En este escenario ideal, ¿cuál es la tasa de transferencia del servidor al cliente? Para responder a esta pregunta, podemos pensar en los bits como en un *flujo* y en los enlaces de comunicaciones como si fueran *tuberías*. Evidentemente, el servidor no puede bombar los bits a través de su enlace a una velocidad mayor que R_s bps; y el router no puede reenviar los bits a una velocidad mayor que R_c bps. Si $R_s < R_c$, entonces los bits bombeados por el servidor “fluirán” a través del router y llegarán al cliente a una velocidad de R_s bps, obteniéndose una tasa de transferencia de R_s bps. Si, por el contrario, $R_c < R_s$, entonces el router no podrá reenviar los bits tan rápidamente como los recibe. En este caso, los bits abandonarán el router a una velocidad de solo R_c , dando lugar a una tasa de transferencia de extremo a extremo igual a R_c . (Observe también que si continúan llegando bits al router a una velocidad R_s , y siguen abandonando el router a una velocidad igual a R_c , la cantidad de bits en el router que están esperando a ser transmitidos hacia el cliente aumentará constantemente, lo que es una situación nada deseable.) Por tanto, en esta sencilla red de dos enlaces, la tasa de transferencia es $\min\{R_c, R_s\}$, es decir, es igual a la velocidad de transmisión del enlace *cuello*.

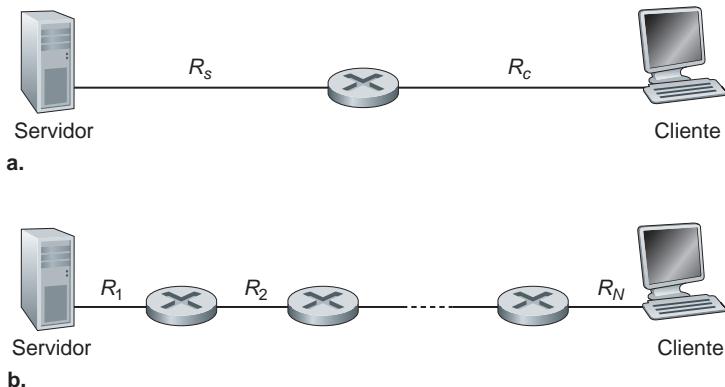


Figura 1.19 ♦ Tasa de transferencia para la transmisión de un archivo desde un servidor a un cliente.

de botella. Una vez determinada la tasa de transferencia, podemos aproximar el tiempo que se tardará en transferir un archivo de gran tamaño de F bits desde el servidor al cliente, mediante la fórmula $F/\min\{R_s, R_c\}$. Veamos un ejemplo concreto. Suponga que está descargando un archivo MP3 de $F = 32$ millones de bits, que el servidor tiene una velocidad de transmisión $R_s = 2$ Mbps y que disponemos de un enlace de acceso con $R_c = 1$ Mbps. El tiempo necesario para transferir el archivo será igual a 32 segundos. Por supuesto, estas expresiones para la tasa de transferencia y el tiempo de transferencia son únicamente aproximaciones, ya que no se han tenido en cuenta los retardos de almacenamiento y reenvío y de procesamiento, ni las cuestiones relativas al protocolo.

La Figura 1.19(b) muestra una red con N enlaces entre el servidor y el cliente, siendo las velocidades de transmisión de los N enlaces iguales a R_1, R_2, \dots, R_N . Aplicando el mismo análisis que para la red de dos enlaces, podemos determinar que la tasa de transferencia para enviar un archivo desde el servidor al cliente es $\min\{R_1, R_2, \dots, R_N\}$, que es de nuevo la velocidad de transmisión del enlace cuello de botella existente en la ruta entre el servidor y el cliente.

Veamos ahora otro ejemplo inspirado en la red Internet de hoy día. La Figura 1.20(a) muestra dos sistemas terminales, un servidor y un cliente, conectados a una red de computadoras. Considere la tasa de transferencia para transmitir un archivo desde el servidor al cliente. El servidor está conectado a la red mediante un enlace de acceso cuya velocidad es R_s y el cliente está conectado a la red mediante un enlace de acceso de velocidad R_c . Supongamos ahora que todos los enlaces existentes en el núcleo de la red de comunicaciones tienen velocidades de transmisión muy altas, muy por encima de R_s y R_c . Ciertamente, hoy en día, el núcleo de Internet está sobredimensionado, con enlaces de alta velocidad que experimentan una congestión muy baja. Supongamos también que únicamente se están enviando a través de la red esos bits que se transfieren desde el servidor al cliente. Dado que el núcleo de la red es como una tubería ancha en este ejemplo, la velocidad a la que los bits pueden fluir desde el origen hasta el destino es, de nuevo, el mínimo de R_s y R_c , es decir, tasa de transferencia = $\min\{R_s, R_c\}$. Por tanto, en la actualidad, el factor de restricción de la tasa de transferencia en Internet es, normalmente, la red de acceso.

Veamos un último ejemplo. Vamos a utilizar la Figura 1.20(b); en ella vemos que hay 10 servidores y 10 clientes conectados al núcleo de la red de computadoras. En este ejemplo, tienen lugar 10 descargas simultáneas, lo que implica a 10 pares cliente-servidor. Supongamos que estas 10 descargas son el único tráfico existente en la red en este momento. Como se muestra en la figura, hay un enlace en el núcleo que es atravesado por las 10 descargas. Sea R la velocidad de transmisión de este enlace R . Supongamos que los enlaces de acceso de todos los servidores tienen la misma velocidad R_s , que los enlaces de acceso de todos los clientes tienen la misma velocidad R_c y que las velocidades de transmisión de todos los enlaces del núcleo (excepto el enlace común de velocidad R) son mucho mayores que R_s , R_c y R . Ahora, deseamos saber cuáles son las tasas de transferencia de las descargas. Evidentemente, si la velocidad del enlace común, R ,

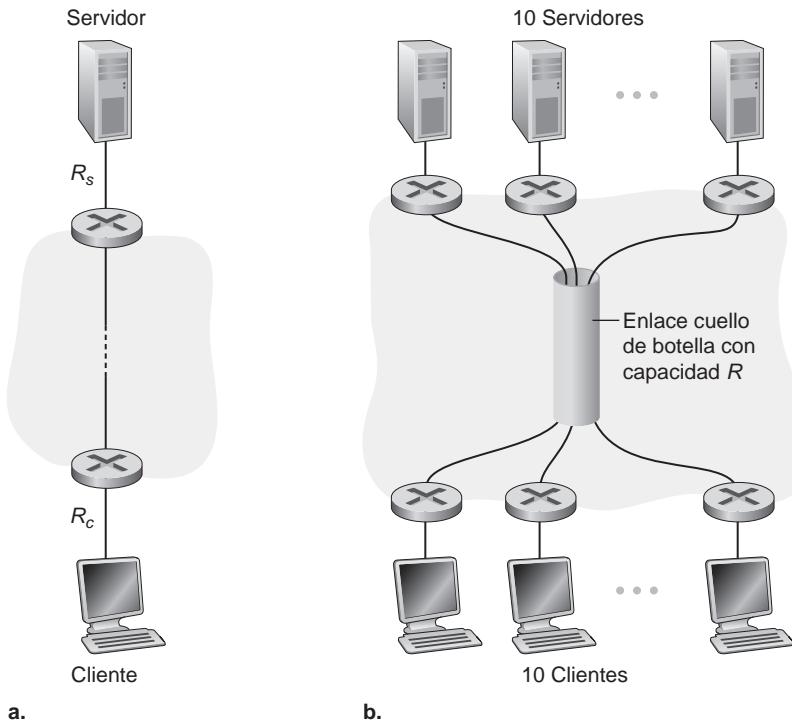


Figura 1.20 ♦ Tasa de transferencia terminal a terminal: (a) un cliente descarga un archivo de un servidor; (b) 10 clientes descargando con 10 servidores.

es grande (por ejemplo, cien veces mayor que R_s y R_c), entonces la tasa de transferencia de cada descarga será igual, de nuevo, a $\min\{R_s, R_c\}$. Pero, ¿qué ocurre si la velocidad del enlace común es del mismo orden que R_s y R_c ? ¿Cuál será la tasa de transferencia en este caso? Veamos un ejemplo concreto. Supongamos que $R_s = 2$ Mbps, $R_c = 1$ Mbps, $R = 5$ Mbps y que el enlace común divide su velocidad de transmisión en partes iguales entre las 10 descargas. Entonces, el cuello de botella para cada descarga ya no se encuentra en la red de acceso, sino en el enlace compartido del núcleo, que solo proporciona una tasa de transferencia de 500 kbps a cada descarga. Luego la tasa de transferencia extremo a extremo para cada descarga ahora se habrá reducido a 500 kbps.

Los ejemplos de las Figuras 1.19 y 1.20(a) demuestran que la tasa de transferencia depende de las velocidades de transmisión de los enlaces a través de los que fluyen los datos. Hemos visto que cuando no existe ningún otro tráfico, la tasa de transferencia puede simplemente aproximarse mediante la velocidad mínima de transmisión a lo largo de la ruta entre el origen y el destino. El ejemplo de la Figura 1.20(b) demuestra que, en el caso más general, la tasa de transferencia depende no solo de las velocidades de transmisión de los enlaces a lo largo de la ruta, sino también del tráfico existente. En particular, un enlace con una velocidad de transmisión alta puede ser, de todos modos, el enlace cuello de botella para la transferencia de un archivo si hay muchos otros flujos de datos atravesando también ese enlace. Examinaremos más detalladamente la tasa de transferencia de las redes de computadoras en los problemas de repaso y en los capítulos siguientes.

1.5 Capas de protocolos y sus modelos de servicio

Después de lo que hemos visto hasta aquí, resulta evidente que Internet es un sistema *extremadamente* complicado. Hemos visto que son muchas las piezas que conforman Internet: numerosas aplicaciones y protocolos, distintos tipos de sistemas terminales, dispositivos de comutación de

paquetes y diversos tipos de medios para los enlaces. Dada esta enorme complejidad, ¿tenemos alguna esperanza de poder organizar una arquitectura de red o al menos nuestra exposición sobre la misma? Afortunadamente, la respuesta a ambas preguntas es sí.

1.5.1 Arquitectura en capas

Antes de intentar organizar nuestras ideas sobre la arquitectura de Internet, vamos a ver una analogía humana. Realmente, de forma continua estamos tratando con sistemas complejos en nuestra vida cotidiana. Imagine que alguien le pide que describa, por ejemplo, cómo funciona el sistema de líneas aéreas. ¿Qué estructura utilizaría para describir este complejo sistema que emplea vendedores de billetes, personal de facturación de equipajes, personal para las puertas de embarque, pilotos, aviones, control de tráfico aéreo y un sistema de ámbito mundial para dirigir los aviones? Una forma de describir este sistema podría consistir en describir la serie de acciones que usted realiza (o que otros realizan para usted) cuando vuela en un avión. En primer lugar, usted compra un billete, luego factura el equipaje, se dirige a la puerta de embarque y por último sube al avión. El avión despegá y se dirige a su destino. Una vez que el avión ha tomado tierra, usted desembarca y recoge su equipaje. Si el viaje ha sido malo, se quejará de ello en el mostrador de venta de billetes (lo que, por supuesto, no le servirá de nada). Este escenario se muestra en la Figura 1.21.

Podemos ver inmediatamente algunas analogías con las redes de computadoras: la compañía área le traslada desde un origen hasta un destino, al igual que Internet transporta un paquete desde un host de origen hasta un host de destino. Pero esta no es la analogía que estábamos buscando. Lo que queremos es encontrar una cierta *estructura* en la Figura 1.21. Si nos fijamos en esta figura, observaremos que hay una función Billete en cada extremo; también existe una función Equipaje para los pasajeros que tienen un billete y una función Embarque para los pasajeros que tienen billete y han facturado su equipaje. Para los pasajeros que han embarcado (es decir, que han adquirido un billete, han facturado las maletas y han embarcado), existe una función de despegue y aterrizaje y, durante el vuelo, hay una función de control del avión. Esto sugiere que podemos fijarnos en las funcionalidades señaladas en la Figura 1.21 de forma *horizontal*, como se muestra en la Figura 1.22.

En la Figura 1.22 se han separado las distintas funciones de la compañía aérea en capas, proporcionando un marco de trabajo en el que podemos explicar cómo se realiza un viaje en avión. Observe que cada capa, combinada con las capas que tiene por debajo, implementa una cierta funcionalidad, un cierto *servicio*. En las capas Billete e inferiores se lleva a cabo la transferencia de una persona de un mostrador de línea aérea a otro. En las capas Equipaje e inferiores se realiza la transferencia de una persona y su equipaje desde el punto de facturación hasta la recogida del equipaje. Observe que la capa Equipaje solo proporciona este servicio a las personas que ya han

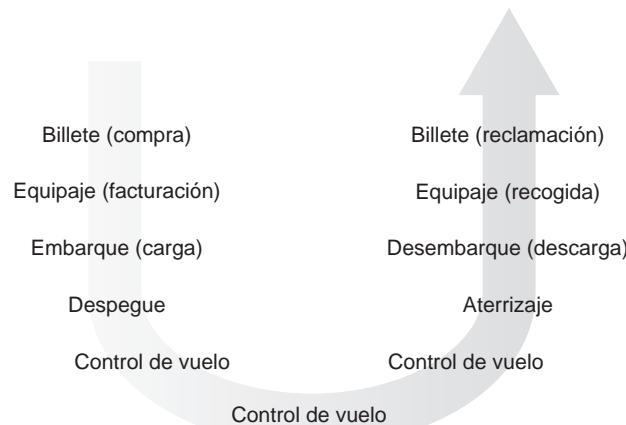


Figura 1.21 ♦ Acciones para realizar un viaje en avión.

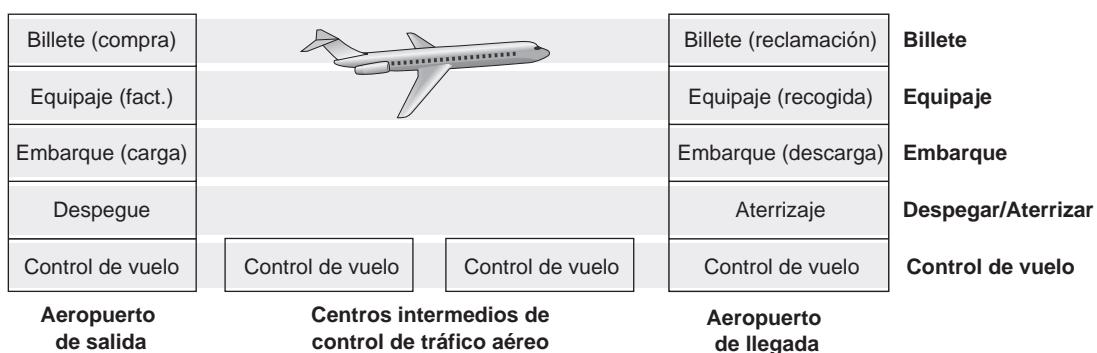


Figura 1.22 ♦ Disposición horizontal de capas para las funcionalidades de una compañía área.

adquirido un billete. En la capa Embarque, se realiza la transferencia embarque/desembarque de una persona y su equipaje. En la capa Despegue/ Aterrizaje, se realiza la transferencia pista a pista de personas y equipajes. Cada capa proporciona su servicio (1) llevando a cabo determinadas acciones dentro de dicha capa (por ejemplo, en la capa Embarque, se hace subir y bajar al pasaje del avión) y (2) utilizando los servicios de la capa que tiene directamente debajo de ella (por ejemplo, en la capa Embarque, utilizando el servicio de transferencia de pasajeros pista a pista de la capa Despegue/ Aterrizaje).

Una arquitectura de capas nos permite estudiar una parte específica y bien definida de un sistema más grande y complejo. Esta simplificación tiene un valor considerable por sí misma, al proporcionar modularidad, haciendo mucho más fácil modificar la implementación del servicio suministrado por la capa. Mientras que la capa proporcione el mismo servicio a la capa que tiene por encima de ella y emplee los mismos servicios de la capa que tiene por debajo, el resto del sistema permanecerá invariable cuando se modifique la implementación de una capa. (¡Tenga en cuenta que cambiar la implementación de un servicio es muy diferente a cambiar el propio servicio!) Por ejemplo, si se modificara la función Embarque para que las personas embarcaran y desembarcaran por alturas, el resto del sistema de la compañía aérea no se vería afectado, ya que la capa Embarque continuaría llevando a cabo la misma función (cargar y descargar personas); simplemente, implementa dicha función de una forma diferente después de realizar el cambio. En sistemas complejos de gran tamaño que se actualizan constantemente, la capacidad de modificar la implementación de un servicio sin afectar al resto de los componentes del sistema es otra importante ventaja de la disposición en capas.

Capas de protocolos

Pero ya hemos hablado suficiente de compañías aéreas. Dirijamos ahora nuestra atención a los protocolos de red. Para proporcionar una estructura al diseño de protocolos de red, los diseñadores de redes organizan los protocolos (y el hardware y el software de red que implementan los protocolos) en capas. Cada protocolo pertenece a una de las capas, del mismo modo que cada función en la arquitectura de la compañía aérea de la Figura 1.22 pertenecía a una capa. De nuevo, estamos interesados en los servicios que ofrece una capa a la capa que tiene por encima, lo que se denomina **modelo de servicio de la capa**. Como en el caso del ejemplo de la compañía aérea, cada capa proporciona su servicio (1) llevando a cabo ciertas acciones en dicha capa y (2) utilizando los servicios de la capa que tiene directamente debajo de ella. Por ejemplo, los servicios proporcionados por la capa n pueden incluir la entrega fiable de mensajes de un extremo de la red al otro. Esto podría implementarse mediante un servicio no fiable de entrega de mensajes extremo a extremo de la capa $n - 1$, y añadiendo una funcionalidad de la capa n para detectar y retransmitir los mensajes perdidos.

Una capa de protocolo puede implementarse por software, por hardware o mediante una combinación de ambos. Los protocolos de la capa de aplicación, como HTTP y SMTP, casi siempre

se implementan por software en los sistemas terminales, al igual que los protocolos de la capa de transporte. Puesto que la capa física y las capas de enlace de datos son responsables de manejar la comunicación a través de un enlace específico, normalmente se implementan en las tarjetas de interfaz de red (por ejemplo, tarjetas de interfaz Ethernet o WiFi) asociadas con un determinado enlace. La capa de red a menudo es una implementación mixta de hardware y software. Observe también que, al igual que las funciones de la arquitectura de la compañía aérea estaban distribuidas entre los distintos aeropuertos y centros de control de vuelo que formaban el sistema, también un protocolo de capa n está *distribuido* entre los sistemas terminales, los dispositivos de conmutación de paquetes y los restantes componentes que conforman la red. Es decir, suele haber una parte del protocolo de capa n en cada uno de estos componentes de red.

Las capas de protocolos presentan ventajas conceptuales y estructurales [RFC 3439]. Como hemos visto, las capas proporcionan una forma estructurada de estudiar los componentes del sistema. Además, la modularidad facilita la actualización de los componentes del sistema. Sin embargo, es preciso señalar que algunos investigadores e ingenieros de redes se oponen vehementemente a la estructura de capas [Wakeman 1992]. Un potencial inconveniente de la estructura de capas es que una capa puede duplicar la funcionalidad de la capa inferior. Por ejemplo, muchas pilas de protocolos proporcionan una función de recuperación de errores tanto para cada enlace, como extremo a extremo. Un segundo inconveniente potencial es que la funcionalidad de una capa puede precisar información (por ejemplo, un valor de una marca temporal) que solo existe en otra capa, y esto viola el objetivo de la separación en capas.

Cuando los protocolos de las distintas capas se toman en conjunto se denominan **pila de protocolos**. La pila de protocolos de Internet consta de cinco capas: capa física, capa de enlace, capa de red, capa de transporte y capa de aplicación, como se muestra en la Figura 1.23(a). Si examina el Contenido, comprobará que hemos organizado a grandes rasgos el libro utilizando las capas de la pila de protocolos de Internet. Vamos a aplicar un **enfoque descendente**, abordando en primer lugar la capa de aplicación y continuando hacia abajo por la pila.

Capa de aplicación

La capa de aplicación es donde residen las aplicaciones de red y sus protocolos de nivel de aplicación. La capa de aplicación de Internet incluye muchos protocolos, tales como el protocolo HTTP (que permite la solicitud y transferencia de documentos web), SMTP (que permite la transferencia de mensajes de correo electrónico) y FTP (que permite la transferencia de archivos entre dos sistemas terminales). Tendremos oportunidad de ver que determinadas funciones de red, como la traducción de los nombres legibles que utilizamos las personas para los sistemas terminales de Internet (por ejemplo, www.ietf.org) en direcciones de red de 32 bits se realiza también con la ayuda de un

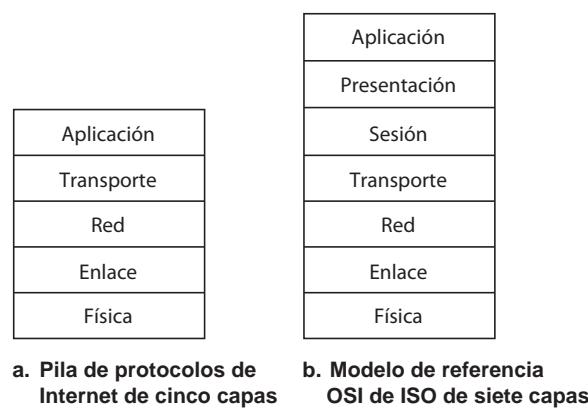


Figura 1.23 ♦ Pila de protocolos de Internet (a) y modelo de referencia OSI (b).

protocolo específico de la capa de aplicación: en concreto, mediante el Sistema de nombres de dominio (DNS, *Domain Name System*). En el Capítulo 2 veremos que es muy fácil crear e implantar nuestros propios protocolos nuevos de la capa de aplicación.

Un protocolo de la capa de aplicación está distribuido entre varios sistemas terminales, con la aplicación en un sistema terminal utilizando el protocolo para intercambiar paquetes de información con la aplicación que se ejecuta en otro sistema terminal. A este paquete de información de la capa de aplicación lo denominaremos **mensaje**.

Capa de transporte

La capa de transporte de Internet transporta los mensajes de la capa de aplicación entre los puntos terminales de la aplicación. En Internet, existen dos protocolos de transporte, TCP y UDP, pudiendo cada uno de ellos transportar los mensajes de la capa de aplicación. TCP ofrece a sus aplicaciones un servicio orientado a la conexión. Este servicio incluye el suministro garantizado de los mensajes de la capa de aplicación al destino y un mecanismo de control del flujo (es decir, una adaptación de las velocidades del emisor y el receptor). TCP también divide los mensajes largos en segmentos más cortos y proporciona un mecanismo de control de congestión, de manera que un emisor regula su velocidad de transmisión cuando la red está congestionada. El protocolo UDP proporciona a sus aplicaciones un servicio sin conexión. Es un servicio básico que no ofrece ninguna fiabilidad, ni control de flujo, ni control de congestión. En este libro, denominaremos a los paquetes de la capa de transporte **segmentos**.

Capa de red

La capa de red de Internet es responsable de trasladar los paquetes de la capa de red, conocidos como **datagramas**, de un host a otro. El protocolo de la capa de transporte Internet (TCP o UDP) de un host de origen pasa un segmento de la capa de transporte y una dirección de destino a la capa de red, al igual que nosotros damos al servicio de correo postal una carta con una dirección de destino. Luego, la capa de red proporciona el servicio de suministrar el segmento a la capa de transporte del host de destino.

La capa de red de Internet incluye el conocido protocolo IP, que define los campos del datagrama, así como la forma en que actúan los sistemas terminales y los routers sobre estos campos. Existe un único protocolo IP y todos los componentes de Internet que tienen una capa de red deben ejecutar el protocolo IP. La capa de red de Internet también contiene protocolos de enrutamiento, que determinan las rutas que los datagramas siguen entre los orígenes y los destinos. Internet dispone de muchos protocolos de enrutamiento. Como hemos visto en la Sección 1.3, Internet es una red de redes y, dentro de una red, el administrador de la red puede ejecutar cualquier protocolo de enrutamiento que deseé. Aunque la capa de red contiene tanto el protocolo IP como numerosos protocolos de enrutamiento, suele hacerse referencia a ella simplemente como la capa IP, lo que refleja el hecho de que IP es el pegamento que une toda la red Internet.

Capa de enlace

La capa de red de Internet encamina un datagrama a través de una serie de routers entre el origen y el destino. Para trasladar un paquete de un nodo (host o router) al siguiente nodo de la ruta, la capa de red confía en los servicios de la capa de enlace. En concreto, en cada nodo, la capa de red pasa el datagrama a la capa de enlace, que entrega el datagrama al siguiente nodo existente a lo largo de la ruta. En ese siguiente nodo, la capa de enlace pasa el datagrama a la capa de red.

Los servicios proporcionados por la capa de enlace dependen del protocolo concreto de la capa de enlace que se emplee en el enlace. Por ejemplo, algunos protocolos de la capa de enlace proporcionan una entrega fiable desde el nodo transmisor, a través de un enlace y hasta el nodo receptor.

Observe que este servicio de entrega fiable es diferente del servicio de entrega fiable de TCP, que lleva a cabo una entrega fiable desde un sistema terminal a otro. Entre los ejemplos de protocolos de la capa de enlace se incluyen Ethernet, WiFi y el protocolo DOCSIS de la red de acceso por cable. Puesto que normalmente los datagramas necesitan atravesar varios enlaces para viajar desde el origen hasta el destino, un datagrama puede ser manipulado por distintos protocolos de la capa de enlace en los distintos enlaces disponibles a lo largo de la ruta. Por ejemplo, un datagrama puede ser manipulado por Ethernet en un enlace y por PPP en el siguiente enlace. La capa de red recibirá un servicio diferente por parte de cada uno de los distintos protocolos de la capa de enlace. En este libro, denominaremos a los paquetes de la capa de enlace **tramas**.

Capa física

Mientras que el trabajo de la capa de enlace es mover tramas completas de un elemento de la red hasta el elemento de red adyacente, el trabajo de la capa física consiste en mover de un nodo al siguiente los *bits individuales* que forman la trama. Los protocolos de esta capa son, de nuevo, dependientes del enlace y, por tanto, dependen del medio de transmisión del enlace (por ejemplo, cable de cobre de par trenzado o fibra óptica monomodo). Por ejemplo, Ethernet dispone de muchos protocolos de la capa física: uno para cable de cobre de par trenzado, otro para cable coaxial, otro para fibra, etc. En cada caso, los bits se desplazan a través del enlace de forma diferente.

El modelo OSI

Una vez vista en detalle la pila de protocolos de Internet, deberíamos mencionar que no es la única pila de protocolos existente. En concreto, a finales de la década de 1970, la Organización Internacional de Estandarización (*ISO, International Organization for Standardization*) propuso que las redes de computadoras fueran organizadas utilizando siete capas, en lo que se vino a denominar **modelo OSI** (*Open Systems Interconnection, Interconexión de sistemas abiertos*) [ISO 2016]. El modelo OSI tomó forma cuando los protocolos que se convertirían en los protocolos de Internet estaban en su infancia y eran simplemente uno de los muchos conjuntos de protocolos diferentes que estaban en desarrollo; de hecho, probablemente los inventores del modelo OSI original no estaban pensando en Internet cuando lo crearon. No obstante, a partir de finales de la década de 1970, muchos cursos universitarios y de formación, siguiendo las recomendaciones de ISO, organizaron cursos sobre el modelo de siete capas. A causa de su temprano impacto sobre la formación en redes, el modelo de siete capas todavía perdura en algunos libros de texto y cursos de formación sobre redes.

Las siete capas del modelo de referencia OSI, mostrado en la Figura 1.23(b), son: **capa de aplicación, capa de presentación, capa de sesión, capa de transporte, capa de red, capa de enlace de datos y capa física**. La funcionalidad de cinco de estas capas es básicamente la misma que sus contrapartidas del mismo nombre de Internet. Por tanto, vamos a centrarnos en las dos capas adicionales del modelo de referencia OSI: la capa de presentación y la capa de sesión. La función de la capa de presentación es la de proporcionar servicios que permitan a las aplicaciones que se comunican interpretar el significado de los datos intercambiados. Estos servicios incluyen la compresión y el cifrado de los datos (funciones cuyos nombres son autoexplicativos), así como la descripción de los datos (lo que libera a la aplicación de tener que preocuparse por el formato interno en el que los datos se representan y almacenan, formatos que pueden diferir de una computadora a otra). La capa de sesión permite delimitar y sincronizar el intercambio de datos, incluyendo los medios para crear un punto de restauración y un esquema de recuperación.

El hecho de que en Internet falten dos de las capas existentes en el modelo de referencia OSI plantea un par de cuestiones interesantes: ¿acaso los servicios proporcionados por estas dos capas no son importantes? ¿Qué ocurre si una aplicación *necesita* uno de estos servicios? La respuesta de Internet a ambas preguntas es la misma: es problema del desarrollador de la aplicación. El desarrollador de la aplicación tiene que decidir si un servicio es importante y si lo es, será su problema el incorporar dicha funcionalidad a la aplicación.

1.5.2 Encapsulación

La Figura 1.24 muestra la ruta física que siguen los datos al descender por la pila de protocolos de un sistema terminal emisor, al ascender y descender por las pilas de protocolos de un switch de la capa de enlace y de un router, para finalmente ascender por la pila de protocolos del sistema terminal receptor. Como veremos más adelante en el libro, los routers y los switches de la capa de enlace operan como dispositivos de commutación de paquetes. De forma similar a los sistemas terminales, los routers y los switches de la capa de enlace organizan su hardware y software de red en capas. Pero estos dispositivos no implementan *todas* las capas de la pila de protocolos; habitualmente solo implementan las capas inferiores. Como se muestra en la Figura 1.24, los switches de la capa de enlace implementan las capas 1 y 2; y los routers implementan las capas 1 a 3. Esto significa, por ejemplo, que los routers de Internet son capaces de implementar el protocolo IP (un protocolo de la capa 3), mientras que los switches de la capa de enlace no. Veremos más adelante que aunque los switches de la capa de enlace no reconocen las direcciones IP, pueden reconocer las direcciones de la capa 2, como por ejemplo las direcciones Ethernet. Observe que los hosts implementan las cinco capas, lo que es coherente con la idea de que la arquitectura de Internet es mucho más compleja en las fronteras de la red.

La Figura 1.24 también ilustra el importante concepto de **encapsulación**. En el host emisor, un mensaje de la capa de aplicación (M en la Figura 1.24) se pasa a la capa de transporte. En el caso más simple, la capa de transporte recibe el mensaje y añade información adicional (denominada información de cabecera de la capa de transporte, H_1 en la Figura 1.24) que será utilizada por la capa de transporte del lado receptor. El mensaje de la capa de aplicación y la información de cabecera de la capa de transporte constituyen el segmento de la capa de transporte. El segmento de la capa de transporte encapsula, por tanto, el mensaje de la capa de aplicación. La información añadida podría incluir información que permita a la capa de transporte del lado receptor entregar el mensaje a la aplicación apropiada, así como bits de detección de errores que permitan al receptor determinar si los bits del mensaje han cambiado a lo largo de la ruta. A continuación, la capa de transporte pasa el segmento a la capa de red, que añade información de cabecera de la capa de red (H_n en la Figura 1.24), como por ejemplo las direcciones de los sistemas terminales de origen y de destino, creando un datagrama de la capa de red. Este datagrama se pasa entonces a la capa de enlace, que (¡por supuesto!) añadirá su propia información de cabecera dando lugar a una trama de la capa de enlace. Así, vemos que en cada capa, un paquete está formado por dos tipos de campos: campos de cabecera y un campo de carga útil. Normalmente, la carga útil es un paquete de la capa superior.

Una buena analogía para ilustrar este tema sería el envío de un informe interno de empresa desde una sucursal a otra a través del servicio postal público. Supongamos que Alicia, que se encuentra en una sucursal, quiere enviar un informe a Benito, que se encuentra en la otra sucursal. El *informe* es análogo al *mensaje de la capa de aplicación*. Alicia introduce el informe en un sobre de correo interno de la empresa y escribe en él el nombre y el departamento de Benito. El *sobre de correo interno* es análogo a un *segmento de la capa de transporte*: contiene información de cabecera (el nombre y el departamento de Benito) y encapsula el mensaje de la capa de aplicación (el informe). Cuando el departamento de correo de la sucursal emisora recibe este sobre, lo mete en otro sobre adecuado para enviar el informe mediante el servicio público de correos. En la sala de correo de la empresa también se escriben en el segundo sobre las direcciones postales tanto de la sucursal emisora como de la sucursal receptora. Aquí, el *sobre postal* es análogo al *datagrama*: encapsula el segmento de la capa de transporte (el sobre de correo interno), que a su vez encapsula el mensaje original (el informe). El servicio postal entrega el sobre postal al departamento de correo de la sucursal receptora, donde comienza el proceso de desencapsulación. En ese departamento se extrae el sobre de correo interno y se envía a Benito. Por último, Benito abre el sobre de correo interno y saca el informe.

El proceso de encapsulación puede ser más complejo que el que acabamos de describir. Por ejemplo, un mensaje largo puede dividirse en varios segmentos de la capa de transporte (los

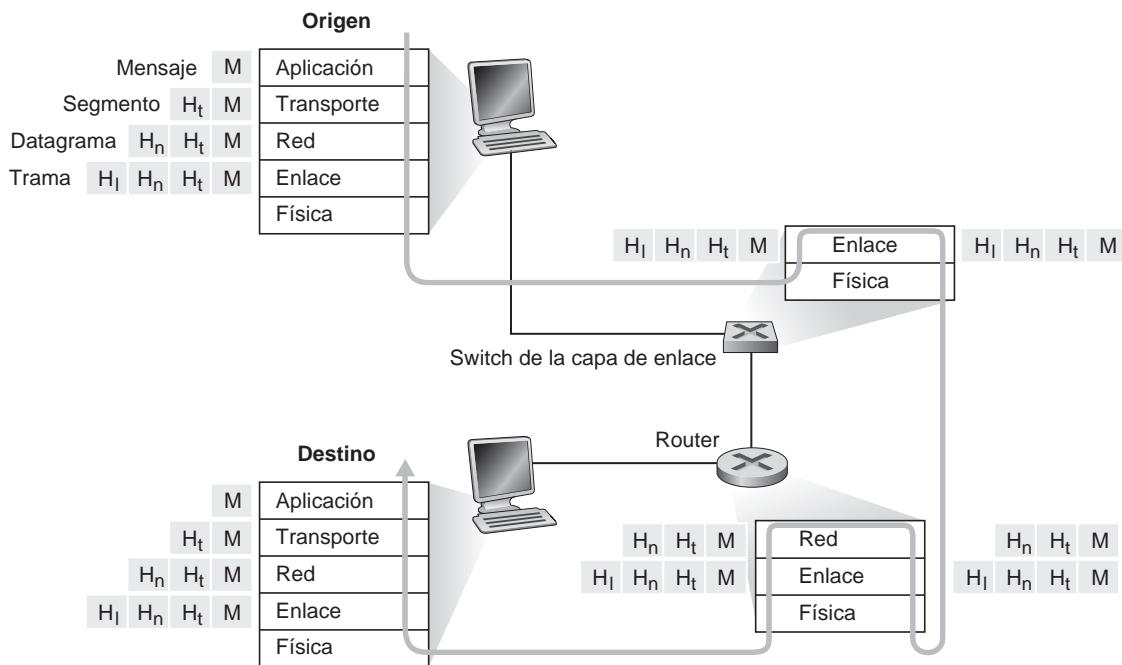


Figura 1.24 ♦ Hosts, routers y switches de la capa de enlace. Cada uno de ellos contiene un conjunto distinto de capas, lo que refleja sus distintas funcionalidades.

cuales a su vez pueden dividirse en varios datagramas de la capa de red). En el extremo receptor, cada segmento deberá entonces ser reconstruido a partir de sus datagramas constituyentes.

1.6 Ataques a las redes

Internet se ha convertido en una herramienta crítica para muchas instituciones actuales, incluyendo empresas pequeñas y medianas, universidades y organismos gubernamentales. Muchas personas individuales también confían en Internet para llevar a cabo muchas de sus actividades profesionales, sociales y personales. Miles de millones de “cosas” (incluyendo dispositivos corporales y electrodomésticos) se conectan hoy en día a Internet. Pero detrás de todas estas utilidades y toda esta excitación, hay un lado oscuro, un lado donde “los malos” intentan hacer estragos en nuestras vidas diarias, dañando nuestras computadoras conectadas a Internet, violando nuestra intimidad y volviendo inoperables los servicios de Internet de los que dependemos.

El campo de la seguridad de red se ocupa de ver cómo “los malos” pueden atacar a las redes de computadoras y de cómo nosotros, que pronto seremos expertos en redes, podemos defendernos frente a estos ataques; o mejor todavía, de cómo diseñar nuevas arquitecturas que sean inmunes a tales ataques. Dada la frecuencia y variedad de ataques existentes, así como la amenaza de nuevos y más destructivos ataques futuros, la seguridad de red se ha convertido en un tema crucial en el campo de las redes de computadoras. Una de las características de este libro de texto es que lleva las cuestiones sobre la seguridad en las redes a primer plano.

Puesto que todavía no tenemos experiencia ni en redes ni en los protocolos de Internet, comenzaremos haciendo un repaso de algunos de los problemas de seguridad que predominan en la actualidad, con el fin de abrir boca para las explicaciones más sustanciosas que proporcionaremos en los capítulos siguientes. Así que empezaremos preguntándonos simplemente, ¿qué es lo que puede ir mal? ¿En qué sentido son vulnerables las redes de computadoras? ¿Cuáles son los principales tipos de ataques hoy día?

Los “malos” pueden introducir software malicioso en nuestro host a través de Internet

Conectamos nuestros dispositivos a Internet porque deseamos enviar y recibir datos hacia/desde Internet, lo que incluye todo tipo de cosas legítimas, como publicaciones Instagram, resultados de búsquedas en Internet, flujos de música, videoconferencias, películas de cine, etc. Pero, lamentablemente, junto con todos estos elementos legítimos también existen elementos maliciosos, lo que se conoce de forma colectiva como software malicioso o **malware**, que puede también acceder a nuestros dispositivos e infectarlos. Una vez que el malware ha infectado un dispositivo, puede hacer todo tipo de maldades, como por ejemplo borrar nuestros archivos o instalar software espía que recopile nuestra información personal, como el número de la seguridad social, contraseñas y pulsaciones de teclas, y luego envíe esos datos (¡a través de Internet, por supuesto!) a “los malos”. Nuestro host comprometido también puede ser reclutado como parte de una red de miles de dispositivos comprometidos de forma similar, lo que se conoce de forma colectiva como **botnet** (red robot), que los atacantes controlan y aprovechan para la distribución de correo electrónico basura (*spam*) o para llevar a cabo ataques distribuidos de denegación de servicio (que pronto explicaremos) contra los hosts objetivo.

Gran parte del malware que existe actualmente es **auto-replicante**: una vez que infecta un host, busca cómo acceder desde dicho host a otros hosts a través de Internet, y desde esos hosts que acaba de infectar, busca cómo acceder a otros. De esta forma, el malware auto-replicante puede extenderse rápidamente de forma exponencial. El malware puede extenderse en forma de virus o de gusano. Un **virus** es un malware que requiere cierta interacción del usuario para infectar el dispositivo. El ejemplo clásico es un adjunto de correo electrónico que contiene código ejecutable malicioso. Si un usuario recibe y abre un adjunto de este tipo, ejecutará inadvertidamente el malware en el dispositivo. Normalmente, tales virus enviados en los mensajes de correo electrónico son auto-replicantes: una vez que se ha ejecutado, el virus puede enviar un mensaje idéntico con el mismo adjunto malicioso a, por ejemplo, todos los contactos de nuestra libreta de direcciones. Un **gusano** es un tipo de malware que puede entrar en un dispositivo sin ninguna interacción explícita por parte del usuario. Por ejemplo, un usuario puede estar ejecutando una aplicación de red vulnerable a la que un atacante puede enviar software malicioso. En algunos casos, sin que el usuario intervenga, la aplicación puede aceptar el malware de Internet y ejecutarlo, creando un gusano. El gusano instalado ahora en el dispositivo recién infectado explora entonces Internet, buscando otros hosts que ejecuten la misma aplicación de red vulnerable. Cuando encuentra otros hosts vulnerables, envía una copia de sí mismo a esos hosts. Por último, un caballo de Troya es un malware que está oculto dentro de otro software que es útil. Hoy día, el malware está generalizado y es costoso defenderse de él. A lo largo del libro, le animaremos a que piense en la siguiente cuestión: ¿qué pueden hacer los diseñadores de redes de computadoras para defender a los dispositivos conectados a Internet de los ataques de malware?

Los “malos” pueden atacar a los servidores y a la infraestructura de red

Otra amplia clase de amenazas de seguridad son los que se conocen como **ataques de denegación de servicio (DoS, Denial-of-Service)**. Como su nombre sugiere, un ataque DoS vuelve inutilizable una red, un host o cualquier otro elemento de la infraestructura para los usuarios legítimos. Los servidores web, los servidores de correo electrónico, los servidores DNS (que se estudian en el Capítulo 2) y las redes institucionales pueden ser, todos ellos, objeto de ataques DoS. Los ataques DoS son muy comunes en Internet, produciéndose miles de ataques de este tipo cada año [Moore 2001]. El sitio web Digital Attack Map permite a los visitantes visualizar los principales ataques DoS de ese día, a nivel mundial [DAM 2016]. La mayoría de los ataques DoS en Internet pueden clasificarse dentro de una de las tres categorías siguientes:

- **Ataque de vulnerabilidad.** Este ataque implica el envío de unos pocos mensajes especialmente diseñados a una aplicación o sistema operativo vulnerable que esté ejecutándose en un host

objetivo. Si se envía la secuencia de paquetes correcta a una aplicación o un sistema operativo vulnerables, el servicio puede detenerse o, lo que es peor, el host puede sufrir un fallo catastrófico.

- **Inundación del ancho de banda.** El atacante envía una gran cantidad de paquetes al host objetivo, de modo que se inunda el enlace de acceso del objetivo, impidiendo que los paquetes legítimos puedan alcanzar al servidor.
- **Inundación de conexiones.** El atacante establece un gran número de conexiones TCP completamente abiertas o semi-abiertas (las conexiones TCP se estudian en el Capítulo 3) con el host objetivo. El host puede llegar a atascarse con estas conexiones fraudulentas, impidiéndose así que acepte las conexiones legítimas.

Vamos a ver a continuación más detalladamente el **ataque por inundación del ancho de banda**. Recordando el análisis que hemos realizado en la Sección 1.4.2 sobre los retardos y la pérdida de paquetes, es evidente que si el servidor tiene una velocidad de acceso de R bps, entonces el atacante tendrá que enviar el tráfico a una velocidad de aproximadamente R bps para causar daños. Si R es muy grande, es posible que un único origen de ataque no sea capaz de generar el tráfico suficiente como para dañar al servidor. Además, si todo el tráfico procede de un mismo origen, un router situado en un punto anterior de la ruta podría detectar el ataque y bloquear todo el tráfico procedente de ese origen, antes de que llegue a逼近 al servidor. En un ataque **DoS distribuido (DDoS, Distributed DoS)**, como el mostrado en la Figura 1.25, el atacante controla varios orígenes y hace que cada uno de ellos bombardee el objetivo con tráfico. Con este método, la tasa acumulada de tráfico para todos los orígenes controlados tiene que ser aproximadamente igual a R para inutilizar el servicio. Actualmente, se producen de forma común ataques DDoS que utilizan botnets con miles de hosts comprometidos [DAM 2016]. Los ataques DDoS son mucho más difíciles de detectar y contrarrestar que los ataques DoS procedentes de un único host.

Le animamos a que piense en la siguiente pregunta según vaya leyendo el libro: ¿qué pueden hacer los diseñadores de redes de computadoras para defenderlas de los ataques DoS? Veremos que son necesarias diferentes defensas para cada uno de los tres tipos de ataques DoS.

Los “malos” pueden examinar y analizar los paquetes

Actualmente, muchos usuarios acceden a Internet a través de dispositivos inalámbricos, tales como computadoras portátiles con conexión WiFi o dispositivos de mano con conexiones Internet

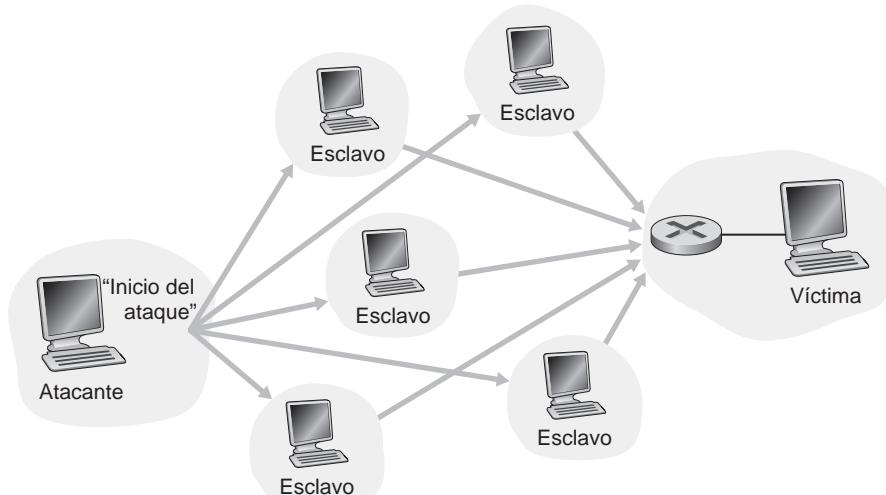


Figura 1.25 ♦ Ataque de denegación de servicio distribuido.

móviles (de lo que hablaremos en el Capítulo 7). Aunque el omnipresente acceso a Internet es extremadamente útil y hace posibles maravillosas aplicaciones nuevas para los usuarios móviles, también provoca una importante vulnerabilidad: **¡colocando un receptor pasivo en las vecindades del transmisor inalámbrico, se puede recibir una copia de todos los paquetes que se están transmitiendo!** Estos paquetes pueden contener todo tipo de información confidencial, incluyendo contraseñas, números de la seguridad social, secretos comerciales y mensajes personales privados. **Un receptor pasivo que registra una copia de todos los paquetes que pasan por él se denomina *sniffer*** (husmeador de paquetes).

Los sniffers también pueden implantarse en entornos cableados. En entornos cableados de multidifusión, como en muchas redes LAN Ethernet, un sniffer puede obtener copias de todos los paquetes enviados a través de la LAN. Como se ha descrito en la Sección 1.2, las tecnologías de acceso por cable también difunden paquetes y son, por tanto, vulnerables a la monitorización y el análisis. Además, un atacante que consiga acceder al router de acceso o al enlace de acceso a Internet de una institución, puede colocar un sniffer que haga una copia de todos los paquetes entrantes y salientes de la organización. Los paquetes así monitorizados pueden ser analizados después en busca de información confidencial.

Hay software sniffer disponible de forma gratuita en varios sitios web y en forma de productos comerciales. Hay profesores que imparten cursos sobre redes y realizan prácticas de laboratorio que implican escribir un programa sniffer y un programa de reconstrucción de datos de la capa de aplicación. **¡De hecho, las prácticas de laboratorio con Wireshark [Wireshark 2016] asociadas con este texto (véase la práctica de laboratorio introductoria de Wireshark al final del capítulo) utilizan precisamente un programa sniffer de ese tipo!**

Puesto que los programas sniffer son pasivos (es decir, no inyectan paquetes en el canal), son difíciles de detectar. Por tanto, cuando enviamos paquetes a un canal inalámbrico, tenemos que aceptar que existe la posibilidad de que algún atacante pueda registrar copias de nuestros paquetes. Como habrá adivinado, una de las mejores formas de defenderse frente a los programas sniffer son las técnicas criptográficas. En el Capítulo 8 veremos cómo se aplica la criptografía a la seguridad de la red.

Los “malos” pueden suplantar la identidad de nuestros conocidos

Es sorprendentemente fácil (*¡y el lector tendrá los conocimientos necesarios para hacerlo muy pronto, a medida que vaya leyendo este texto!*) crear un paquete con una dirección de origen, un contenido de paquete y una dirección de destino arbitrarios y luego transmitir dicho paquete a Internet, que reenviará el paquete a su destino. Imagine que el receptor confiado (por ejemplo, un router de Internet) que recibe tal paquete, toma la dirección de origen (falsa) como buena y luego ejecuta algún comando integrado en el contenido del paquete (por ejemplo, modifica la tabla de reenvío). **La capacidad de inyectar paquetes en Internet con una dirección de origen falsa se conoce como *suplantación IP*** y es una de las muchas formas en las que un usuario puede hacerse pasar por otro.

Para resolver este problema, **necesitaremos aplicar un medio de autenticación en el punto terminal**, es decir, un mecanismo que nos permita determinar con seguridad si un mensaje tiene su origen donde creemos que lo tiene. De nuevo, animamos a los lectores a que, a medida que avanzan por los capítulos del libro, piensen en cómo pueden hacer esto las aplicaciones y protocolos de red. En el Capítulo 8 exploraremos los mecanismos de autenticación en el punto terminal.

Para terminar con esta sección, vale la pena comentar cuál es la razón de que Internet se haya convertido en un lugar inseguro. Básicamente, la respuesta es que Internet fue diseñada originalmente de esa manera, ya que se basaba en el modelo de un “grupo de usuarios que confiaban entre sí, conectados a una red transparente” [Blumenthal 2001] —un modelo en el que (por definición) no había necesidad de pensar en la seguridad. Muchos aspectos de la arquitectura de Internet original reflejan profundamente esta idea de confianza mutua. Por ejemplo, la posibilidad de que un usuario envíe un paquete a cualquier otro usuario es la opción predeterminada, en lugar de ser una capacidad

solicitada/concedida, al igual que lo normal es creer que la identidad del usuario es la que declara, en lugar de autenticarle por defecto.

Pero actualmente Internet no implica realmente “usuarios de confianza mutua”. Sin embargo, los usuarios de hoy día siguen necesitando comunicarse aunque no necesariamente confíen entre sí, pueden querer comunicarse de forma anónima, pueden comunicarse indirectamente a través de terceros (por ejemplo, cachés web, que estudiaremos en el Capítulo 2, o asistentes de movilidad, que veremos en el Capítulo 7) y pueden desconfiar del hardware, del software e incluso del aire a través del que se comunican. A lo largo del libro vamos a encontrarnos con muchos retos relacionados con la seguridad: buscaremos formas de defendernos frente a los sniffer, frente a la suplantación de identidades de los puntos terminales, frente a los ataques por interposición, frente a los ataques DDoS, frente al malware, etc. Tenemos que tener presente que la comunicación entre usuarios de mutua confianza es la excepción, más que la regla. ¡Bienvenido al mundo de las redes modernas de comunicaciones!

1.7 Historia de Internet y de las redes de computadoras

En las Secciones 1.1 a 1.6 se ha hecho una presentación de las tecnologías utilizadas en las redes de comunicaciones e Internet. ¡Ahora ya sabe lo suficiente como para impresionar a sus familiares y amigos! Sin embargo, si realmente desea causar una gran impresión en la siguiente fiesta a la que asista, debería salpicar su discurso con algunos detalles interesantes acerca de la fascinante historia de Internet [Segaller 1998].

1.7.1 El desarrollo de la conmutación de paquetes: 1961–1972

El campo de las redes de computadoras y de la actual Internet tiene sus inicios a principios de la década de 1960, cuando la red telefónica era la red de comunicaciones dominante en el mundo. Recordemos de la Sección 1.3 que la red telefónica utiliza mecanismos de conmutación de circuitos para transmitir la información entre un emisor y un receptor, una opción apropiada en la que la voz se transmite a una velocidad constante entre el emisor y el receptor. Debido a la creciente importancia de las computadoras en los primeros años de la década de 1960 y a la aparición de las computadoras de tiempo compartido, quizás fue natural considerar la cuestión de cómo enlazar las computadoras con el fin de que pudieran ser compartidas entre usuarios geográficamente distribuidos. El tráfico generado por esos usuarios tenía una tendencia a ser a *ráfagas*, compuesto por períodos de actividad como el envío de un comando a una computadora remota, seguidos de períodos de inactividad mientras se espera a obtener una respuesta o mientras se contempla la respuesta recibida.

Tres grupos de investigación repartidos por el mundo, cada uno de ellos ignorante de la existencia de los otros [Leiner 1998], comenzaron a trabajar en la conmutación de paquetes como alternativa eficiente y robusta a la conmutación de circuitos. El primer trabajo publicado sobre las técnicas de conmutación de paquetes fue el de Leonard Kleinrock [Kleinrock 1961; Kleinrock 1964], por aquél entonces un estudiante de posgrado en el MIT. Basándose en la teoría de colas, el trabajo de Kleinrock demostraba de forma elegante la efectividad de la técnica de conmutación de paquetes para las fuentes que generaban tráfico a ráfagas. En 1964, Paul Baran [Baran 1964], en el Rand Institute, había comenzado a investigar el uso de la conmutación de paquetes para comunicaciones de voz seguras en redes militares y, en el National Physical Laboratory (NPL) de Inglaterra, Donald Davies y Roger Scantlebury también estaban desarrollando sus ideas acerca de la conmutación de paquetes.

Los trabajos realizados en el MIT, en el instituto Rand y en los laboratorios NPL establecieron las bases de la red Internet actual. Pero Internet también tiene una larga tradición de “hagámoslo y demostremos que funciona” que también se remonta a la década de 1960. J. C. R. Licklider

[DEC 1990] y Lawrence Roberts, ambos colegas de Kleinrock en el MIT, dirigieron el programa de Ciencias de la Computación en la Agencia de Proyectos de Investigación Avanzada (ARPA, Advanced Research Projects Agency) de Estados Unidos. Roberts publicó un plan global para la red ARPAnet [Roberts 1967], la primera red de computadoras de conmutación de paquetes y un ancestro directo de la red Internet pública actual. El Día del Trabajo de 1969, se instaló el primer dispositivo de conmutación de paquetes en UCLA bajo la supervisión de Kleinrock y poco después se instalaron tres dispositivos más de conmutación de paquetes en el Instituto de Investigación de Stanford (SRI), en la Universidad de California en Santa Barbara y en la Universidad de Utah (Figura 1.26). Hacia finales de 1969, la naciente red precursora de Internet estaba formada por cuatro nodos. Kleinrock recuerda la primera vez que utilizó la red para llevar a cabo un inicio de sesión remoto desde UCLA en el SRI, consiguiendo que el sistema fallara [Kleinrock 2004].

Hacia 1972, la red ARPAnet había crecido aproximadamente hasta 15 nodos y la primera demostración pública ya había sido realizada por Robert Kahn. También se completó por aquel entonces el primer protocolo host a host entre sistemas terminales de ARPAnet, conocido como el protocolo de control de red (NCP, *Network Control Protocol*) [RFC 001]. Disponiendo de un protocolo extremo a extremo, ahora podían escribirse aplicaciones. Ray Tomlinson escribió el primer programa de correo electrónico en 1972.

1.7.2 Redes propietarias e interredes: 1972–1980

La red inicial ARPAnet era una única red cerrada. Para establecer una comunicación con un host de la red ARPAnet, había que estar físicamente conectado a otro dispositivo de conmutación de



Figura 1.26 ♦ Uno de los primeros dispositivos de conmutación de paquetes.

paquetes de ARPAnet. A mediados de la década de 1970, comenzaron a surgir otras redes de conmutación de paquetes autónomas además de ARPAnet: ALOHANet, una red de microondas que enlazaba universidades de las islas Hawái [Abramson 1970], así como redes de conmutación de paquetes vía satélite [RFC 829] y vía radio [Kahn 1978] de DARPA; Telenet, una red de conmutación de paquetes comercial de BBN basada en la tecnología de ARPAnet; Cyclades, una red de conmutación de paquetes francesa dirigida por Louis Pouzin [Think 2009]; redes de tiempo compartido tales como Tymnet y la red de Servicios de información de GE, entre otras, a finales de la década de 1960 y principios de la década de 1970 [Schwartz 1977] y la red SNA de IBM (1969-1974), que constituía un desarrollo paralelo al de ARPAnet [Schwartz 1977].

El número de redes estaba creciendo. Retrospectivamente, podemos ver que había llegado el momento de desarrollar una arquitectura completa para la interconexión de redes. El trabajo pionero sobre interconexión de redes (realizado bajo el patrocinio de la Agencia DARPA, *Defense Advanced Research Projects Agency*), que consistía en esencia en la creación de una *red de redes*, fue realizado por Vinton Cerf y Robert Kahn [Cerf 1974]; el término *internetworking* (interredes o interconexión de redes) fue acuñado para describir este trabajo.

Estos principios arquitectónicos fueron integrados en TCP. Sin embargo, las primeras versiones de TCP eran bastante distintas al TCP de hoy día. Esas primeras versiones combinaban una entrega fiable en secuencia de los datos mediante retransmisiones por parte del sistema terminal (esta función todavía la realiza TCP hoy día) con funciones de reenvío (que actualmente son realizadas por IP). Los primeros experimentos con TCP, combinados con el reconocimiento de la importancia de disponer de un servicio de transporte extremo a extremo no fiable y sin control de flujo para aplicaciones tales como voz empaquetada, llevaron a separar IP de TCP y al desarrollo del protocolo UDP. Los tres protocolos clave de Internet que se emplean actualmente (TCP, UDP e IP) ya habían sido concebidos a finales de la década de 1970.

Además de las investigaciones relativas a Internet de la agencia DARPA, también estaban en marcha muchos otros importantes proyectos relacionados con las redes. En Hawái, Norman Abramson estaba desarrollando ALOHANet, una red de paquetes vía radio que permitía a múltiples sitios remotos de las islas Hawái comunicarse entre sí. El protocolo ALOHA [Abramson 1970] fue el primer protocolo de acceso múltiple, que permitió a usuarios geográficamente distribuidos compartir un mismo medio de comunicación de difusión (una frecuencia de radio). Metcalfe y Boggs se basaron en el protocolo de acceso múltiple de Abramson para desarrollar el protocolo Ethernet [Metcalfe 1976], para redes de difusión compartidas basadas en cable; véase la Figura 1.27. Es interesante comentar que el desarrollo del protocolo Ethernet de Metcalfe y Boggs fue motivado por la necesidad de conectar varios PC, impresoras y discos compartidos [Perkins 1994]. Hace veinticinco años, bastante antes de la revolución de los PC y de la explosión de las redes, Metcalfe y Boggs estaban sentando las bases para las redes LAN de computadoras PC actuales.

1.7.3 Proliferación de las redes: 1980–1990

A finales de la década de 1970, había aproximadamente unos doscientos hosts conectados a la red ARPAnet. A finales de la década de 1980, el número de hosts conectados a la red Internet pública, una confederación de redes similar a la Internet actual, llegaría a los cien mil. La década de 1980 fue una época de enorme crecimiento.

Gran parte de este crecimiento fue el resultado de varios y distintos esfuerzos por crear redes de computadoras que enlazaran universidades. BITNET proporcionaba servicios de correo electrónico y de transferencia de archivos entre varias universidades del noreste de los Estados Unidos. CSNET (Computer Science Network) se formó para que investigadores universitarios que no tenían acceso a la red ARPAnet pudieran comunicarse. En 1986 se creó NSFNET para proporcionar acceso a los centros de supercomputación patrocinados por la NSF. Con una velocidad en la red troncal de 56 kbps inicialmente, la red troncal de NSFNET llegaría a operar a 1,5 Mbps a finales de la década y serviría como una red troncal primaria para enlazar redes regionales.

En la comunidad ARPAnet, muchas de las piezas finales de la arquitectura de Internet actual fueron encajando. El 1 de enero de 1983 se llevó a cabo el lanzamiento oficial de TCP/IP como el nuevo protocolo de host estándar para ARPAnet (reemplazando al protocolo NCP). La transición [RFC 801] de NCP a TCP/IP fue un suceso señalado: a todos los hosts se les requirió pasar a utilizar TCP/IP ese día. A finales de la década de 1980, se realizaron importantes extensiones en TCP con el fin de implementar un control de congestión basado en host [Jacobson 1988]. También se desarrolló el sistema DNS, que se emplea para establecer la correspondencia entre los nombres de Internet que usamos las personas (por ejemplo, gaia.cs.umass.edu) y sus direcciones IP de 32 bits [RFC 1034].

En paralelo con este desarrollo de ARPAnet (realizado fundamentalmente en Estados Unidos), a principios de la década de 1980, los franceses lanzaron el proyecto Minitel, un ambicioso plan para llevar las redes de datos a todos los hogares. Patrocinado por el gobierno francés, el sistema Minitel consistía en una red pública de commutación de paquetes (basada en la serie de protocolos X.25), servidores Minitel y terminales económicos que incorporaban modems de baja velocidad. Minitel alcanzó un gran éxito en 1984 cuando el gobierno francés proporcionó un terminal Minitel gratuito a todo aquel que deseara tener uno en su casa. Los sitios Minitel incluían sitios gratuitos, como por ejemplo el directorio telefónico, y sitios privados, que cobraban sus tarifas basándose en la utilización del servicio. Minitel alcanzó su pico de uso a mediados de la década de 1990, llegándose a ofrecer más de 20.000 servicios, que iban desde servicios de banca hasta bases de datos de investigación especializadas. Minitel se utilizaba en un gran porcentaje de hogares de Francia 10 años antes de que la mayoría de los americanos hubiera oído siquiera hablar de Internet.

1.7.4 La explosión de Internet: década de 1990

La década de 1990 estuvo marcada por una serie de acontecimientos que simbolizaron la continua evolución y la pronta llegada de la comercialización de Internet. ARPAnet, el progenitor de Internet, dejó de existir. En 1991, NSFNET retiró sus restricciones sobre el uso de NSFNET para propósitos comerciales. La propia NSFNET fue eliminada del servicio activo en 1995, pasando el tráfico troncal de Internet a ser transportado por proveedores de servicios Internet (ISP) comerciales.

Sin embargo, el principal acontecimiento de la década de 1990 fue la aparición de la aplicación World Wide Web, que llevaría Internet a los hogares y negocios de millones de personas de todo el mundo. La Web sirvió como plataforma para posibilitar e implantar cientos de nuevas aplicaciones, que hoy damos por sentadas, incluyendo las búsquedas (por ejemplo, Google y Bing), el comercio Internet (por ejemplo, Amazon y eBay) y las redes sociales (por ejemplo, Facebook).

La Web fue inventada en el CERN por Tim Berners-Lee entre 1989 y 1991 [Berners- Lee 1989], basándose en las ideas de los primeros trabajos acerca del hipertexto desarrolladas en la década de 1940 por Vannevar Bush [Bush 1945] y luego a partir de la década de 1960 por Ted Nelson [Xanadu 2009]. Berners-Lee y sus colegas desarrollaron las versiones iniciales de HTML, HTTP, un servidor web y un navegador (los cuatro componentes clave de la Web). Hacia finales del año 1993 estaban operativos aproximadamente doscientos servidores web. Esta colección de servidores solo sería un presagio de lo que estaba por venir. Al mismo tiempo, había varios investigadores desarrollando navegadores web con interfaces GUI, entre los que se encontraba Marc Andreessen, que fundó junto con Jim Clark la empresa Mosaic Communications, que más tarde se convertiría en Netscape Communications Corporation [Cusumano 1998; Quittner 1998]. Hacia 1995, los estudiantes universitarios empleaban los navegadores Netscape para navegar por la Web diariamente. También por esa época, las empresas, grandes y pequeñas, comenzaron a trabajar con servidores web y a realizar transacciones comerciales a través de la Web. En 1996, Microsoft empezó a desarrollar navegadores y comenzaría la guerra de los navegadores entre Netscape y Microsoft, que terminaría ganando Microsoft unos pocos años después [Cusumano 1998].

La segunda mitad de la década de 1990 fue un periodo de gran crecimiento e innovación para Internet, con grandes corporaciones y miles de empresas de nueva creación desarrollando productos y servicios Internet. A finales del milenio, Internet daba soporte a cientos de aplicaciones populares, entre las que se incluyen las cuatro de más éxito:

- Correo electrónico, incluyendo los adjuntos y el correo electrónico accesible a través de la Web.
- La Web, incluyendo la navegación web y el comercio por Internet.
- La mensajería instantánea, con listas de contactos.
- La compartición igualitaria de archivos MP3, de la que Napster fue la pionera.

Es interesante saber que las dos primeras de estas aplicaciones estrella fueron creadas por la comunidad de investigadores, mientras que las dos últimas lo fueron por unos pocos jóvenes emprendedores.

El periodo comprendido entre 1995 y 2001 fue un paseo en montaña rusa para Internet en los mercados financieros. Antes de que fueran incluso rentables, cientos de nuevas empresas de Internet realizaron ofertas públicas de venta y comenzaron a cotizar en los mercados bursátiles. Muchas empresas fueron valoradas en miles de millones de dólares sin tener ingresos significativos. Las acciones de Internet se hundieron en 2000-2001 y muchas de esas empresas de nueva creación cerraron. No obstante, algunas de ellas emergieron como los grandes ganadores en el espacio Internet, entre las que se incluyen Microsoft, Cisco, Yahoo, e-Bay, Google y Amazon.

1.7.5 El nuevo milenio

La innovación en el campo de las redes de computadoras continúa a gran velocidad. Se están haciendo avances en todos los frentes, incluyendo la implantación de routers más rápidos y el uso de mayores velocidades de transmisión, tanto en redes de acceso como en las redes troncales. Pero los siguientes desarrollos merecen especial atención:

- Desde el principio del milenio, hemos sido testigos de la agresiva implantación de accesos Internet de banda ancha en las viviendas —no solo modems de cable y líneas DSL, sino también enlaces de fibra óptica—, como se explica en la Sección 1.2. Este acceso a Internet de alta velocidad ha preparado el terreno para una amplia variedad de aplicaciones de vídeo, incluyendo la distribución de videos generados por el usuario (por ejemplo, YouTube), la distribución de películas y programas de televisión a la carta (por ejemplo, Netflix) y videoconferencias multipersona (por ejemplo, Skype, Facetime y Google Hangouts).
- La creciente ubicuidad de las redes WiFi públicas de alta velocidad (54 Mbps y superiores) y de los accesos a Internet a velocidad media (decenas de Mbps) a través de redes de telefonía móvil 4G, no solo está haciendo posible que permanezcamos continuamente conectados mientras nos movemos, sino que también permite nuevas aplicaciones específicas de la ubicación, como Yelp, Tinder, Yik Yak y Waz. El número de dispositivos inalámbricos conectados a Internet sobre pasó en 2011 al número de dispositivos cableados. Este acceso inalámbrico de alta velocidad ha hecho posible la rápida aparición de computadoras de mano (iPhones, Androids, iPads, etc.), que disfrutan de un acceso a Internet constante y sin ataduras.
- Las redes sociales en línea —como Facebook, Instagram, Twitter y WeChat (inmensamente popular en China)— han dado lugar a la aparición de enormes redes de personas basadas en Internet. Muchas de estas redes sociales son ampliamente utilizadas tanto para mensajería, como para compartición de fotografías. Muchos usuarios actuales de Internet “viven” principalmente dentro de una o más redes sociales. A través de sus APIs, las redes sociales en línea crean plataformas para nuevas aplicaciones en red y juegos distribuidos.
- Como hemos explicado en la Sección 1.3.3, los proveedores de servicios en línea, como Google y Microsoft, han implantado sus propias redes privadas de gran envergadura, que no solo conectan entre sí sus centros de datos globalmente distribuidos, sino que se utilizan también para puentejar Internet en la medida de lo posible, comunicándose directamente con los ISP de nivel inferior. Como resultado, Google proporciona resultados de búsquedas y acceso al correo electrónico casi instantáneos, como si sus centros de datos se estuvieran ejecutando en nuestras propias computadoras.

- Muchas empresas de comercio electrónico a través de Internet ejecutan ahora sus aplicaciones en la “nube” —como en el EC2 de Amazon, en el Application Engine de Google o en el Azure de Microsoft—. Muchas empresas y universidades han migrado también sus aplicaciones Internet (por ejemplo, el correo electrónico o el hospedaje web) a la nube. Las empresas de servicios de computación en la nube no solo proporcionan entornos de almacenamiento y ejecución escalables para las aplicaciones, sino que también permiten a estas un acceso implícito a sus redes privadas de altas prestaciones.

1.8 Resumen

¡En este capítulo hemos presentado una gran cantidad de material! Hemos hablado de los distintos componentes hardware y software que forman Internet, en particular, y las redes de computadoras en general. Hemos partido de la frontera de la red, fijándonos en los sistemas terminales y las aplicaciones, y en el servicio de transporte proporcionado a las aplicaciones que se ejecutan en los sistemas terminales. También hemos hablado de las tecnologías de la capa de enlace y de los medios físicos que pueden encontrarse normalmente en la red de acceso. A continuación, nos hemos adentrado en el interior de la red, en su núcleo, y hemos identificado los mecanismos de conmutación de paquetes y de conmutación de circuitos como los dos métodos básicos utilizados para el transporte de datos a través de una red de telecomunicaciones, y hemos examinado las fortalezas y las debilidades de cada método. También hemos examinado la estructura global de Internet, y hemos aprendido que es una red de redes. Hemos visto que la estructura jerárquica de Internet, formada por ISP de nivel superior e inferior, ha permitido que Internet crezca hasta incluir miles de redes.

En la segunda parte de este capítulo de introducción, hemos abordado varios temas fundamentales del campo de las redes. En primer lugar, hemos estudiado las causas de los retardos, la tasa de transferencia y la pérdida de paquetes en una red de conmutación de paquetes. Hemos desarrollado modelos cuantitativos simples para determinar los retardos de transmisión, de propagación y de cola, así como para la tasa de transferencia; haremos un uso exhaustivo de estos modelos de retardo en los problemas de repaso incluidos a lo largo del libro. A continuación, hemos visto las capas de protocolos y los modelos de servicio, las principales claves arquitectónicas de las redes a las que se volverá a hacer referencia a lo largo del libro. Asimismo, hemos repasado los ataques de seguridad más habituales actualmente en Internet. Hemos terminado nuestra introducción con un breve repaso a la historia de las redes de computadoras. Este primer capítulo en sí mismo constituye un minicurso sobre redes de computadoras.

Por tanto, ¡hemos cubierto una enorme cantidad de conceptos básicos en este primer capítulo! Si se siente un poco abrumado, no se preocupe: en los siguientes capítulos revisaremos todas estas ideas, viéndolas con mucho más detalle (¡lo que es una promesa, no una amenaza!). En este punto, esperamos que termine el capítulo teniendo ya una idea sobre las piezas que forman una red, teniendo un conocimiento (que todavía deberá desarrollar) del vocabulario del campo de las redes de computadoras (no se asuste por tener que volver a consultar este capítulo) y habiendo incrementado su deseo por aprender más acerca de las redes. Esta es la tarea que tenemos por delante durante el resto del libro.

Mapa de este libro

Antes de iniciar cualquier viaje, siempre debería echarse un vistazo a un mapa de carreteras, para familiarizarse con las principales carreteras e intersecciones con los que nos encontraremos más adelante. En el viaje en el que nos hemos embarcado nosotros, el destino final es conocer en profundidad el cómo, el qué y el por qué de las redes de computadoras, y nuestro mapa son los capítulos de este libro:

1. Redes de computadoras e Internet
2. La capa de aplicación
3. La capa de transporte
4. La capa de red: plano de datos
5. La capa de red: plano de control
6. La capa de enlace y las redes de área local
7. Redes inalámbricas y móviles
8. Seguridad en las redes de computadoras
9. Redes multimedia

Los Capítulos 2 a 6 son los cinco capítulos centrales del libro. Observará que están organizados según las cuatro capas superiores de la pila de protocolos de Internet de cinco capas. Observará también que nuestro viaje va a comenzar por la parte superior de la pila de protocolos de Internet, es decir, por la capa de aplicación, y luego continuaremos nuestro trabajo descendiendo por la pila. La razón de hacer este recorrido de arriba a abajo es porque, una vez que conozcamos las aplicaciones, estaremos en condiciones de comprender los servicios de red necesarios para dar soporte a esas aplicaciones. Podremos así examinar a continuación las distintas formas en que tales servicios pueden ser implementados por una arquitectura de red. Ocuparnos de las aplicaciones en primer lugar nos va a proporcionar la motivación necesaria para abordar el resto del texto.

En la segunda mitad del libro, Capítulos 7 a 9, se abordan tres temas enormemente importantes (y en cierto modo independientes) en las redes de comunicaciones modernas. En el Capítulo 7 examinaremos las redes inalámbricas y celulares, incluyendo las redes LAN inalámbricas (lo que incluye las tecnologías WiFi y Bluetooth), las redes de telefonía móvil (lo que incluye las redes GSM, 3G y 4G) y la movilidad (tanto en redes IP como GSM). En el Capítulo 8, dedicado a la seguridad en las redes de computadoras, veremos en primer lugar los fundamentos de los mecanismos de cifrado y de seguridad de red y después examinaremos cómo está siendo aplicada la teoría básica a un amplio rango de contextos de Internet. El último capítulo, dedicado a las redes multimedia, examina aplicaciones de audio y de vídeo tales como la telefonía Internet, la videoconferencia y los flujos de información multimedia almacenada. También veremos cómo pueden diseñarse redes de conmutación de paquetes para proporcionar una calidad de servicio coherente a las aplicaciones de audio y de vídeo.

Problemas y cuestiones de repaso

Capítulo 1 Cuestiones de repaso

SECCIÓN 1.1

- R1. ¿Cuál es la diferencia entre un host y un sistema terminal? Enumere distintos tipos de sistemas terminales. ¿Es un servidor web un sistema terminal?
- R2. El término *protocolo* a menudo se emplea para describir las relaciones diplomáticas. ¿Cómo describe Wikipedia un protocolo diplomático?
- R3. ¿Por qué son importantes los estándares para los protocolos?

SECCIÓN 1.2

- R4. Enumere seis tecnologías de acceso. Clasifíquelas como de acceso residencial, acceso empresarial o acceso inalámbrico de área extensa.
- R5. ¿La velocidad de transmisión en un sistema HFC es dedicada o compartida entre los usuarios? ¿Pueden producirse colisiones en un canal de descarga HFC? ¿Por qué o por qué no?
- R6. Enumere las tecnologías de acceso residencial disponibles en su ciudad. Para cada tipo de acceso, detalle la velocidad de descarga ofrecida, la velocidad de carga y el precio mensual.

- R7. ¿Cuál es la velocidad de transmisión en las redes LAN Ethernet?
- R8. Cite algunos de los medios físicos sobre los que se puede emplear la tecnología Ethernet.
- R9. Para el acceso residencial se emplean modems de acceso telefónico, sistemas HFC, DSL y FTTH. Para cada una de estas tecnologías de acceso, detalle el rango de velocidades de transmisión e indique si la velocidad de transmisión es dedicada o compartida.
- R10. Describa las tecnologías de acceso inalámbrico a Internet más populares hoy día. Compárelas e indique sus diferencias.

SECCIÓN 1.3

- R11. Suponga que hay un único dispositivo de conmutación de paquetes entre un host emisor y un host receptor. Las velocidades de transmisión entre el host emisor y el dispositivo de conmutación y entre este y el host receptor son R_1 y R_2 , respectivamente. Suponiendo que el switch utiliza el mecanismo de conmutación de paquetes de almacenamiento y reenvío, ¿cuál es el retardo total extremo a extremo si se envía un paquete de longitud L ? (Ignore los retardos de cola, de propagación y de procesamiento.)
- R12. ¿Qué ventaja presenta una red de conmutación de circuitos frente a una red de conmutación de paquetes? ¿Qué ventajas tiene la multiplexación TDM frente a la multiplexación FDM en una red de conmutación de circuitos?
- R13. Suponga que los usuarios comparten un enlace de 2 Mbps y que cada usuario transmite a una velocidad constante de 1 Mbps, pero solo durante el 20 por ciento del tiempo. (Véase la comparación entre conmutación de circuitos y conmutación de paquetes de la Sección 1.3.)
 - a. Si se utiliza la conmutación de circuitos, ¿a cuántos usuarios se puede dar soporte?
 - b. Para el resto del problema, suponga que se utiliza la conmutación de paquetes. ¿Por qué prácticamente no habrá retardo de cola antes del enlace si dos o menos usuarios transmiten a un mismo tiempo? ¿Por qué existirá retardo de cola si tres usuarios transmiten simultáneamente?
 - c. Calcule la probabilidad de que un usuario dado esté transmitiendo.
 - d. Suponga ahora que hay tres usuarios. Calcule la probabilidad de que en un instante determinado los tres usuarios estén transmitiendo simultáneamente. Halle la fracción de tiempo durante la que la cola crece.
- R14. ¿Por qué dos ISP en un mismo nivel de la jerarquía a menudo se interconectan entre sí? ¿Cómo obtiene sus ingresos un IXP?
- R15. Algunos proveedores de contenido han creado sus propias redes. Describa la red de Google. ¿Qué es lo que lleva a los proveedores de contenido a crear estas redes?

SECCIÓN 1.4

- R16. Considere el envío de un paquete desde un host emisor a un host receptor a través de una ruta fija. Enumere los componentes del retardo extremo a extremo. ¿Cuáles de estos retardos son constantes y cuáles son variables?
- R17. Visite el applet *Transmission Versus Propagation Delay* (retardo de transmisión frente a retardo de propagación) disponible en el sitio web del libro. Utilizando las velocidades, retardos de propagación y tamaños de paquete disponibles, determine una combinación para la cual el emisor termine la operación de transmisión antes de que el primer bit del paquete haya llegado al receptor. Halle otra combinación para la que el primer bit del paquete llegue al receptor antes de que el emisor haya terminado de transmitir.
- R18. ¿Cuánto tiempo tarda un paquete cuya longitud es de 1.000 bytes en propagarse a través de un enlace a una distancia de 2.500 km, siendo la velocidad de propagación igual a $2,5 \cdot 10^8$ m/s y la velocidad de transmisión de 2 Mbps? De forma más general, ¿cuánto tiempo tarda un paquete de longitud L en propagarse a través de un enlace a una distancia d , con una

velocidad de propagación s y una velocidad de transmisión de R bps? ¿Depende este retardo de la longitud del paquete? ¿Depende este retardo de la velocidad de transmisión?

- R19. Suponga que el host A desea enviar un archivo de gran tamaño al host B. La ruta desde el host A al host B está formada por tres enlaces, cuyas velocidades son $R_1 = 500$ kbps, $R_2 = 2$ Mbps y $R_3 = 1$ Mbps.
- Suponiendo que no hay ningún otro tráfico en la red, ¿cuál es la tasa de transferencia para el archivo?
 - Suponga que el tamaño del archivo es de 4 millones de bytes. Dividiendo el tamaño del archivo entre la tasa de transferencia, ¿cuánto tiempo tardará aproximadamente en transferirse el archivo al host B?
 - Repita los apartados (a) y (b), pero ahora con R_2 reducida a 100 kbps.
- R20. Suponga que el sistema terminal A desea enviar un archivo de gran tamaño al sistema terminal B. Sin entrar en detalles, describa cómo crea el sistema terminal A los paquetes a partir del archivo. Cuando uno de estos paquetes llega a un conmutador de paquetes, ¿qué información del paquete utiliza el conmutador para determinar el enlace por el que debe ser reenviado el paquete? ¿Por qué la conmutación de paquetes en Internet es análoga a viajar de una ciudad a otra preguntando por la dirección a la que nos dirigimos?
- R21. Visite el applet *Queuing and Loss* (colas y pérdida de paquetes) en el sitio web del libro. ¿Cuáles son las velocidades de transmisión máxima y mínima? Para esas velocidades, ¿cuál es la intensidad de tráfico? Ejecute el applet con esas velocidades y determine cuánto tiempo tiene que transcurrir para que tenga lugar una pérdida de paquetes. A continuación, repita el experimento una segunda vez y determine de nuevo cuánto tiempo pasa hasta producirse una pérdida de paquetes. ¿Son diferentes los valores obtenidos? ¿Por qué o por qué no?

SECCIÓN 1.5

- R22. Enumere cinco tareas que puede realizar una capa. ¿Es posible que una (o más) de estas tareas pudieran ser realizadas por dos (o más) capas?
- R23. ¿Cuáles son las cinco capas de la pila de protocolos Internet? ¿Cuáles son las responsabilidades principales de cada una de estas capas?
- R24. ¿Qué es un mensaje de la capa de aplicación? ¿Y un segmento de la capa de transporte? ¿Y un datagrama de la capa de red? ¿Y una trama de la capa de enlace?
- R25. ¿Qué capas de la pila de protocolos de Internet procesa un router? ¿Qué capas procesa un switch de la capa de enlace? ¿Qué capas procesa un host?

SECCIÓN 1.6

- R26. ¿Cuál es la diferencia entre un virus y un gusano?
- R27. Describa cómo se puede crear una red robot (botnet) y cómo se puede utilizar en un ataque DDoS.
- R28. Suponga que Alicia y Benito están enviándose paquetes entre sí a través de una red. Imagine que Victoria se introduce en la red de modo que puede capturar todos los paquetes enviados por Alicia y que luego envía lo que ella quiere a Benito. Además, también puede capturar todos los paquetes enviados por Benito y luego enviar a Alicia lo que le parezca. Enumere algunos de los daños que Victoria puede ocasionar desde su posición.

Problemas

- P1. Diseñe y describa un protocolo de nivel de aplicación que será utilizado entre un cajero automático y la computadora central de un banco. El protocolo deberá permitir verificar la tarjeta y la contraseña del usuario, consultar el saldo de la cuenta (que se almacena en la

computadora central) y retirar dinero de la cuenta (entregándose el dinero al usuario). Las entidades del protocolo deben poder manejar el caso, bastante común, de que no haya suficiente saldo en la cuenta como para cubrir el reembolso. Especifique el protocolo enumerando los mensajes intercambiados y las acciones realizadas por el cajero automático o la computadora central del banco al transmitir y recibir mensajes. Haga un boceto del funcionamiento del protocolo para el caso de una retirada de efectivo sin errores, utilizando un diagrama similar al mostrado en la Figura 1.2. Indique explícitamente las suposiciones hechas por el protocolo acerca del servicio de transporte extremo a extremo subyacente.

- P2. La Ecuación 1.1 proporciona una fórmula para calcular el retardo extremo a extremo al enviar un paquete de longitud L a través de N enlaces de velocidad de transmisión R . Generalice esa fórmula para el caso de enviar P paquetes seguidos a través de los N enlaces.
- P3. Considere una aplicación que transmite datos a una velocidad constante (por ejemplo, el emisor genera una unidad de datos de N bits cada k unidades de tiempo, donde k es un valor pequeño y fijo). Además, cuando esta aplicación se inicia, continúa ejecutándose durante un periodo de tiempo relativamente largo. Responda a las siguientes cuestiones, justificando de forma breve su respuesta:
- ¿Qué sería más apropiado para esta aplicación, una red de conmutación de circuitos o una red de conmutación de paquetes? ¿Por qué?
 - Suponga que se utiliza una red de conmutación de paquetes y que el único tráfico que existe en la misma procede de la aplicación que acabamos de describir. Además, suponga que la suma de las velocidades de datos de la aplicación es menor que las capacidades de cada uno de los enlaces. ¿Será necesario algún mecanismo de control de congestión? ¿Por qué?
- P4. Considere la red de conmutación de circuitos de la Figura 1.13. Recuerde que hay 4 circuitos en cada enlace. Etiquete los cuatro dispositivos de conmutación con los nombres A, B, C y D, yendo en el sentido de las agujas del reloj.
- ¿Cuál es el número máximo de conexiones simultáneas que pueden estar en curso en un determinado instante de tiempo en esta red?
 - Suponga que todas las conexiones se encuentran entre los dispositivos de conmutación A y C. ¿Cuál será el número máximo de conexiones simultáneas que puede haber en curso?
 - Suponga que queremos realizar cuatro conexiones entre los dispositivos de conmutación A y C y otras cuatro entre los dispositivos B y D. ¿Podemos enrutar estas llamadas a través de los cuatro enlaces, de forma que se de soporte a las ocho conexiones?
- P5. Repase la analogía de la caravana de coches de la Sección 1.4. Suponga una velocidad de propagación de 100 km/hora.
- Suponga que la caravana se mueve a una velocidad de 150 km, empezando delante de una caseta de peaje, pasando por una segunda caseta de peaje y terminando justo después de una tercera caseta de peaje. ¿Cuál es el retardo extremo a extremo?
 - Repita el apartado (a) suponiendo ahora que en la caravana hay ocho coches en lugar de diez.
- P6. En este problema elemental comenzamos a explorar los retardos de propagación y de transmisión, dos conceptos fundamentales en las redes de datos. Considere dos hosts, A y B, conectados por un único enlace cuya velocidad es de R bps. Suponga que los dos hosts están separados m metros y que la velocidad de propagación a lo largo del enlace es igual a s metros/segundo. El host A tiene que enviar un paquete de tamaño L bits al host B.
- Exprese el retardo de propagación, d_{prop} , en función de m y s .
 - Determine el tiempo de transmisión del paquete, d_{trans} , en función de L y R .
 - Ignorando los retardos de procesamiento y de cola, obtenga una expresión para el retardo extremo a extremo.



Exploración de los
retardos de propagación
y transmisión.

- d. Suponga que el host A comienza a transmitir el paquete en el instante $t = 0$. En el instante $t = d_{\text{trans}}$, ¿dónde estará el último bit del paquete?
- e. Suponga que d_{prop} es mayor que d_{trans} . En el instante $t = d_{\text{trans}}$, ¿dónde estará el primer bit del paquete?
- f. Suponga que d_{prop} es menor que d_{trans} . En el instante $t = d_{\text{trans}}$, ¿dónde estará el primer bit del paquete?
- g. Suponga que $s = 2,5 \cdot 10^8$ m/s, $L = 120$ bits y $R = 56$ kbps. Determine la distancia m de modo que d_{prop} sea igual a d_{trans} .
- P7. En este problema vamos a considerar la transmisión de voz en tiempo real desde el host A al host B a través de una red de conmutación de paquetes (VoIP). El host A convierte sobre la marcha la voz analógica en un flujo de bits digital a 64 kbps. A continuación, el host A agrupa los bits en paquetes de 56 bytes. Entre el host A y el host B existe un enlace, cuya velocidad de transmisión es de 2 Mbps y cuyo retardo de propagación es igual a 10 milisegundos. Tan pronto como el host A forma un paquete, lo envía al host B. Cuando el host B recibe un paquete completo, convierte los bits del paquete en una señal analógica. ¿Cuánto tiempo transcurre desde el momento en que se crea un bit (a partir de la señal analógica en el host A) hasta que se decodifica (como parte de la señal analógica en el host B)?
- P8. Suponga que varios usuarios comparten un enlace de 3 Mbps. Suponga también que cada usuario requiere 150 kbps para transmitir y que solo transmite durante el 10 por ciento del tiempo. (Véase la comparación entre conmutación de circuitos y conmutación de paquetes en la Sección 1.3.)
- Si se utiliza la conmutación de circuitos, ¿a cuántos usuarios se puede dar soporte?
 - Para el resto de este problema, suponga que se utiliza una red de conmutación de paquetes. Halle la probabilidad de que un determinado usuario esté transmitiendo.
 - Suponga que hay 120 usuarios. Determine la probabilidad de que en un instante determinado haya exactamente n usuarios transmitiendo simultáneamente. (*Sugerencia:* utilice la distribución binomial.)
 - Calcule la probabilidad de que haya 21 o más usuarios transmitiendo simultáneamente.
- P9. Consulte la comparación entre conmutación de circuitos y conmutación de paquetes en la Sección 1.3, en la que se proporciona un ejemplo con un enlace a 1 Mbps. Los usuarios están generando datos a una velocidad de 100 kbps cuando están ocupados, pero solo lo están con una probabilidad de $p = 0,1$. Suponga que el enlace a 1 Mbps se sustituye por un enlace a 1 Gbps.
- ¿Cuál es el valor de N , el número máximo de usuarios a los que se les puede dar soporte simultáneamente, cuando se emplea una red de conmutación de circuitos?
 - Considere ahora que se utiliza una red conmutación de paquetes y que el número de usuarios es M . Proporcione una fórmula (en función de p , M , N) para determinar la probabilidad de que más de N usuarios estén enviando datos.
- P10. Considere un paquete de longitud L que tiene su origen en el sistema terminal A y que viaja a través de tres enlaces hasta un sistema terminal de destino. Estos tres enlaces están conectados mediante dos dispositivos de conmutación de paquetes. Sean d_i , s_i y R_i la longitud, la velocidad de propagación y la velocidad de transmisión del enlace i , para $i = 1, 2, 3$. El dispositivo de conmutación de paquetes retarda cada paquete d_{proc} . Suponiendo que no se producen retardos de cola, ¿cuál es el retardo total extremo a extremo del paquete, en función de d_i , s_i , R_i , ($i = 1, 2, 3$) y L ? Suponga ahora que la longitud del paquete es de 1.500 bytes, que la velocidad de propagación en los tres enlaces es igual a $2,5 \cdot 10^8$ m/s, que la velocidad de transmisión en los tres enlaces es de 2 Mbps, que el retardo de procesamiento en el conmutador de paquetes es de 3 milisegundos, que la longitud del primer enlace es de 5.000 km, que la del

segundo es de 4.000 km y que la del último enlace es de 1.000 km. Para estos valores, ¿cuál es el retardo extremo a extremo?

- P11. En el problema anterior, suponga que $R_1 = R_2 = R_3 = R$ y $d_{\text{proc}} = 0$. Suponga también que el conmutador de paquetes no almacena los paquetes y los reenvía, sino que transmite inmediatamente cada bit que recibe, sin esperar a que llegue el paquete completo. ¿Cuál será el retardo extremo a extremo?
- P12. Un conmutador de paquetes recibe un paquete y determina el enlace saliente por el que deberá ser reenviado. Cuando el paquete llega, hay otro paquete que ya ha sido transmitido hasta la mitad por el mismo enlace de salida y además hay otros cuatro paquetes esperando para ser transmitidos. Los paquetes se transmiten según el orden de llegada. Suponga que todos los paquetes tienen una longitud de 1.500 bytes y que la velocidad del enlace es de 2 Mbps. ¿Cuál será el retardo de cola para el paquete? En sentido más general, ¿cuál es el retardo de cola cuando todos los paquetes tienen una longitud L , la velocidad de transmisión es R , x bits del paquete que se está transmitiendo actualmente ya han sido transmitidos y hay n paquetes en la cola esperando a ser transmitidos?
- P13. (a) Suponga que N paquetes llegan simultáneamente a un enlace en el que actualmente no se está transmitiendo ningún paquete, ni tampoco hay ningún paquete en cola. Cada paquete tiene una longitud L y el enlace tiene una velocidad de transmisión R . ¿Cuál es el retardo medio de cola para los N paquetes?
 (b) Ahora suponga que llegan al enlace N de esos paquetes cada LN/R segundos. ¿Cuál será el retardo medio de cola de cada paquete?
- P14. Considere el retardo de cola en el buffer de un router. Sea I la intensidad de tráfico; es decir, $I = La/R$. Suponga que el retardo de cola puede expresarse como $IL/R(1 - I)$ para $I < 1$.
- Determine una fórmula para calcular el retardo total, es decir, el retardo de cola más el retardo de transmisión.
 - Dibuje el retardo total en función de L/R .
- P15. Sea a la velocidad de llegada de los paquetes a un enlace, en paquetes/segundo, y sea μ la velocidad de transmisión del enlace en paquetes/segundo. Basándose en la fórmula del retardo total (es decir, el retardo de cola más el retardo de transmisión) obtenida en el problema anterior, deduzca una fórmula para el retardo total en función de a y μ .
- P16. Considere el buffer de un router que precede a un enlace de salida. En este problema utilizaremos la fórmula de Little, una fórmula famosa en la teoría de colas. Sea N el número medio de paquetes que hay en el buffer más el paquete que está siendo transmitido. Sea a la velocidad a la que los paquetes llegan al enlace. Sea d el retardo medio total (es decir, el retardo de cola más el retardo de transmisión) experimentado por un paquete. La fórmula de Little es $N = a \cdot d$. Suponga que, como media, el buffer contiene 10 paquetes y que el retardo medio de cola de un paquete es igual a 10 ms. La velocidad de transmisión del enlace es igual a 100 paquetes/segundo. Utilizando la fórmula de Little, ¿cuál es la velocidad media de llegada de los paquetes, suponiendo que no se produce pérdida de paquetes?
- P17. a. Generalice la Ecuación 1.2 dada en la Sección 1.4.3 para velocidades de procesamiento, velocidades de transmisión y retardos de propagación heterogéneos.
 b. Repita el apartado (a), pero suponiendo ahora que existe un retardo medio de cola d_{cola} en cada nodo.
- P18. Realice un trazado de la ruta (Traceroute) entre un origen y un destino situados en un mismo continente para tres horas del día diferentes.
- Determine la media y la desviación estándar de los retardos de ida y vuelta para cada una de las horas.
 - Determine el número de routers existente en la ruta para cada una de las horas. ¿Ha variado la ruta para alguna de las horas?



Nota de video
Uso de Traceroute para descubrir rutas de red y medir los retardos de red.

- c. Intente identificar el número de redes de ISP que los paquetes de Traceroute atraviesan desde el origen hasta el destino. Los routers con nombres similares y/o direcciones IP similares deben considerarse como parte del mismo ISP. En sus experimentos, ¿los retardos más largos se producen en las interfaces situadas entre proveedores ISP adyacentes?
- d. Repita el apartado anterior para un origen y un destino situados en diferentes continentes. Compare los resultados para ubicaciones en un mismo continente y en distintos continentes.
- P19. (a) Visite el sitio www.traceroute.org y realice sendos trazados de ruta desde dos ciudades diferentes de Francia a un mismo host de destino en Estados Unidos. ¿Cuántos enlaces coinciden en los dos trazados de ruta? ¿Coincide el enlace trasatlántico?
- (b) Repita el apartado (a), pero esta vez seleccione una ciudad en Francia y otra en Alemania.
- (c) Seleccione una ciudad en los Estados Unidos y realice sendos trazados de ruta a dos hosts situados en ciudades diferentes de China. ¿Cuántos enlaces tienen en común los dos trazados de ruta? ¿Divergen los dos trazados de ruta antes de alcanzar China?
- P20. Considere el ejemplo sobre la tasa de transferencia correspondiente a la Figura 1.20(b). Suponga ahora que hay M parejas cliente-servidor en lugar de 10. Sean R_s , R_c y R las velocidades de los enlaces de servidor, de los enlaces de cliente y del enlace de red. Suponga que todos los restantes enlaces tienen la capacidad suficiente y que no existe otro tráfico en la red que el generado por las M parejas cliente-servidor. Deduzca una expresión general para la tasa de transferencia en función de R_s , R_c , R y M .
- P21. Considere la Figura 1.19(b). Suponga ahora que existen M rutas entre el servidor y el cliente. No hay dos rutas que comparten ningún enlace. La ruta k ($k = 1, \dots, M$) consta de N enlaces con velocidades de transmisión iguales a $R_1^k, R_2^k, \dots, R_N^k$. Si el servidor solo puede utilizar una ruta para enviar datos al cliente, ¿cuál será la máxima tasa de transferencia que puede alcanzar dicho servidor? Si el servidor puede emplear las M rutas para enviar datos, ¿cuál será la máxima tasa de transferencia que puede alcanzar el servidor?
- P22. Considere la Figura 1.19(b). Suponga que cada enlace entre el servidor y el cliente tiene una probabilidad de pérdida de paquetes p y que las probabilidades de pérdida de paquetes de estos enlaces son independientes. ¿Cuál es la probabilidad de que un paquete (enviado por el servidor) sea recibido correctamente por el receptor? Si un paquete se pierde en el camino que va desde el servidor hasta el cliente, entonces el servidor volverá a transmitir el paquete. Como media, ¿cuántas veces tendrá que retransmitir el paquete el servidor para que el cliente lo reciba correctamente?
- P23. Considere la Figura 1.19(a). Suponga que sabemos que el enlace cuello de botella a lo largo de la ruta entre el servidor y el cliente es el primer enlace, cuya velocidad es R_s bits/segundo. Suponga que enviamos un par de paquetes uno tras otro desde el servidor al cliente y que no hay más tráfico que ese en la ruta. Suponga que cada paquete tiene un tamaño de L bits y que ambos enlaces presentan el mismo retardo de propagación d_{prop} .
- ¿Cuál es el periodo entre llegadas de paquetes al destino? Es decir, ¿cuánto tiempo transcurre desde que el último bit del primer paquete llega hasta que lo hace el último bit del segundo paquete?
 - Suponga ahora que el enlace cuello de botella es el segundo enlace (es decir, $R_c < R_s$). ¿Es posible que el segundo paquete tenga que esperar en la cola de entrada del segundo enlace? Explique su respuesta. Suponga ahora que el servidor envía el segundo paquete T segundos después de enviar el primero. ¿Qué valor debe tener T para garantizar que el segundo paquete no tenga que esperar en la cola de entrada del segundo enlace? Explique su respuesta.
- P24. Suponga que necesita enviar de forma urgente 40 terabytes de datos de Boston a Los Ángeles. Dispone de un enlace dedicado a 100 Mbps para la transferencia de datos. ¿Qué preferiría,

transmitir los datos a través del enlace o utilizar FedEx para hacer el envío por la noche? Explique su respuesta.

- P25. Se tienen dos hosts, A y B, separados 20.000 kilómetros y conectados mediante un enlace directo con $R = 2 \text{ Mbps}$. Suponga que la velocidad de propagación por el enlace es igual a $2,5 \cdot 10^8 \text{ m/s}$.
- Calcule el producto ancho de banda-retardo, $R \cdot d_{\text{prop}}$.
 - Supongamos que se envía un archivo cuyo tamaño es de 800.000 bits desde el host A al host B. Suponga que el archivo se envía de forma continua como un mensaje de gran tamaño. ¿Cuál es el número máximo de bits que habrá en el enlace en un instante de tiempo determinado?
 - Haga una interpretación del producto ancho de banda-retardo.
 - ¿Cuál es el ancho (en metros) de un bit dentro del enlace? ¿Es más grande que un campo de fútbol?
 - Deduzca una expresión general para la anchura de un bit en función de la velocidad de propagación s , la velocidad de transmisión R y la longitud del enlace m .
- P26. Continuando con el Problema P25, suponga que podemos modificar R . ¿Para qué valor de R es el ancho de un bit tan grande como la longitud del enlace?
- P27. Considere el Problema P25 pero ahora para un enlace con $R = 1 \text{ Gbps}$.
- Calcule el producto ancho de banda-retardo, $R \cdot d_{\text{prop}}$.
 - Considere el envío de un archivo de 800.000 bits desde el host A al host B. Suponga que el archivo se envía de forma continua como un mensaje de gran tamaño. ¿Cuál es el número máximo de bits que puede haber en el enlace en cualquier instante de tiempo dado?
 - ¿Cuál es el ancho (en metros) de un bit dentro del enlace?
- P28. Haciendo referencia de nuevo al problema P25.
- ¿Cuánto tiempo tarda en enviarse el archivo, suponiendo que se envía de forma continua?
 - Suponga ahora que el archivo se divide en 20 paquetes, conteniendo cada uno de ellos 40.000 bits. Suponga también que el receptor confirma la recepción de cada paquete y que el tiempo de transmisión de un paquete de confirmación es despreciable. Por último, suponga que el emisor no puede transmitir un paquete hasta que el anterior haya sido confirmado. ¿Cuánto tiempo se tardará en enviar el archivo?
 - Compare los resultados obtenidos en los apartados (a) y (b).
- P29. Suponga que existe un enlace de microondas a 10 Mbps entre un satélite geoestacionario y su estación base en la Tierra. El satélite toma una fotografía digital por minuto y la envía a la estación base. Supongamos que la velocidad de propagación es $2,4 \cdot 10^8 \text{ m/s}$.
- ¿Cuál es el retardo de propagación del enlace?
 - ¿Cuál es el producto ancho de banda-retardo, $R \cdot d_{\text{prop}}$?
 - Sea x el tamaño de la fotografía. ¿Cuál es el valor mínimo de x para que el enlace de microondas esté transmitiendo continuamente?
- P30. Considere la analogía de la compañía aérea utilizada en la Sección 1.5 al hablar sobre las capas, y la adición de cabeceras a las unidades de datos del protocolo a medida que fluyen en sentido descendente por la pila de protocolos. ¿Existe algún concepto equivalente a la información de cabecera que se añade a los pasajeros y al equipaje a medida que descienden por la pila de protocolos de la compañía aérea?
- P31. En las redes de conmutación de paquetes modernas, el host de origen segmenta los mensajes largos de la capa de aplicación (por ejemplo, una imagen o un archivo de música) en paquetes más pequeños y los envía a la red. Después, el receptor ensambla los paquetes para formar

el mensaje original. Este proceso se conoce como *segmentación de mensajes*. La Figura 1.27 ilustra el transporte extremo a extremo de un mensaje con y sin segmentación del mensaje. Imagine que se envía un mensaje cuya longitud es de $8 \cdot 10^6$ bits desde el origen hasta el destino mostrados en la Figura 1.27. Suponga que cada enlace de los mostrados en la figura es de 2 Mbps. Ignore los retardos de propagación, de cola y de procesamiento.

- Suponga que el mensaje se transmite desde el origen al destino *sin* segmentarlo. ¿Cuánto tiempo tarda el mensaje en desplazarse desde el origen hasta el primer comutador de paquetes? Teniendo en cuenta que cada comutador de paquetes utiliza el método de conmutación de almacenamiento y reenvío, ¿cuál el tiempo total que invierte el mensaje para ir desde el host de origen hasta el host de destino?
 - Suponga ahora que el mensaje se segmenta en 800 paquetes, siendo la longitud de cada paquete igual a 10.000 bits. ¿Cuánto tiempo tarda el primer paquete en transmitirse desde el origen hasta el primer comutador de paquetes? Mientras se está enviando el primer paquete del primer comutador al segundo, el host de origen está enviando un segundo paquete al primer comutador de paquetes. ¿En qué instante de tiempo habrá recibido el primer comutador el segundo paquete completo?
 - ¿Cuánto tiempo tarda en transmitirse el archivo desde el host de origen al host de destino cuando se emplea la segmentación de mensajes? Compare este resultado con la respuesta del apartado (a) y coméntelo.
 - Además de reducir el retardo, ¿qué otras razones hay para usar la segmentación de mensajes?
 - Comente los inconvenientes de la segmentación de mensajes.
- P32. Experimente con el applet *Message Segmentation* (segmentación de mensajes) disponible en el sitio web del libro. ¿Se corresponden los retardos indicados en el applet con los retardos del problema anterior? ¿Cómo afectan los retardos de propagación del enlace al retardo global extremo a extremo de la conmutación de paquetes (con segmentación de mensajes) y de la conmutación de mensajes?
- P33. Se envía un archivo de gran tamaño de F bits desde el host A al host B. Entre los hosts A y B hay tres enlaces (y dos dispositivos de conmutación) y los enlaces no están congestionados (es decir, no existen retardos de cola). El host A divide el archivo en segmentos de S bits y añade 80 bits de cabecera a cada segmento, formando paquetes de $L = 80 + S$ bits. La velocidad de transmisión de cada enlace es de R bps. Calcule el valor de S que minimiza el retardo al transmitir el archivo desde el host A al host B. No tenga en cuenta el retardo de propagación.

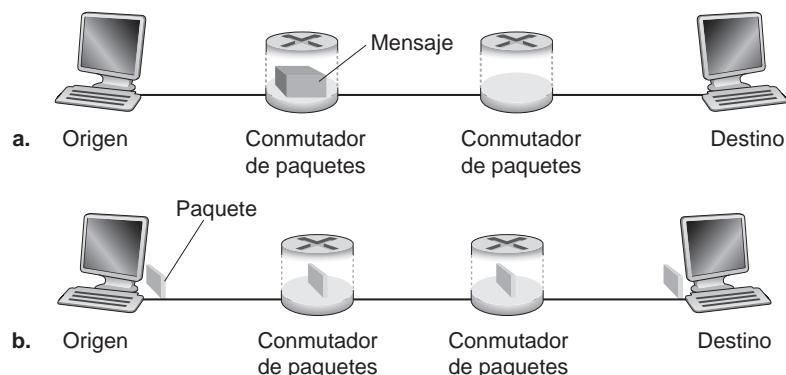


Figura 1.27 ♦ Transporte de mensajes extremo a extremo: (a) sin segmentación de mensajes; (b) con segmentación de mensajes.

- P34. Skype ofrece un servicio que permite realizar una llamada de teléfono desde un PC a un teléfono normal. Esto significa que la llamada de voz debe pasar tanto a través de Internet como a través de una red telefónica. Explique cómo podría hacerse esto.

Prácticas de laboratorio con Wireshark

“Dímelo y lo olvidaré. Enséñamelo y lo recordaré. Implícame y lo entenderé.”
Proverbio chino

Se puede comprender de forma más profunda los protocolos de red viéndolos en acción y jugando con ellos —observando la secuencia de mensajes intercambiados entre dos entidades, examinando los detalles de la operación del protocolo, haciendo que los protocolos lleven a cabo determinadas acciones y observando dichas acciones y sus consecuencias. Esto puede hacerse en escenarios simulados o en un entorno de red real, como Internet. Las applets Java disponibles en el sitio web del libro aplican el primero de estos métodos. En las prácticas de laboratorio con Wireshark se aplicará el segundo método. Se ejecutan aplicaciones de red en diversos escenarios utilizando una computadora doméstica, de una oficina o de un laboratorio de prácticas. Podrá observar los protocolos de red en su equipo, interactuando e intercambiando mensajes con entidades que se ejecutan en algún otro lugar de Internet. Así, usted y su computadora serán una parte integral de estas prácticas de laboratorio. Podrá observar practicando y, de ese modo, aprender.

La herramienta básica para observar los mensajes intercambiados entre entidades que ejecutan protocolos es un **husmeador de paquetes (packet sniffer)**. Como su nombre sugiere, un husmeador de paquetes copia de forma pasiva los mensajes que están siendo enviados y recibidos por nuestra computadora; también muestra el contenido de los distintos campos de protocolo de los mensajes capturados. En la Figura 1.28 se muestra una captura de pantalla del software Wireshark. Wireshark

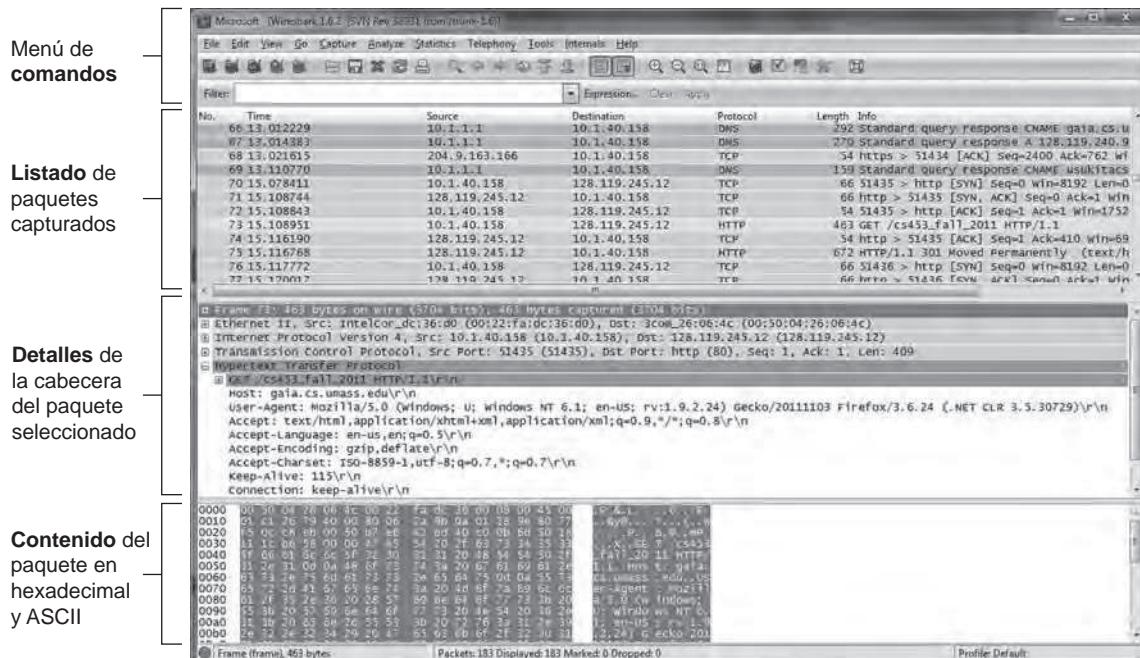


Figura 1.28 ♦ Una captura de pantalla de Wireshark (reimpresa con permiso de Wireshark Foundation).

es un husmeador de paquetes gratuito que se ejecuta en sistemas Windows, Linux/Unix y Mac. A lo largo del libro, encontrará prácticas de laboratorio con Wireshark que le permitirán explorar algunos de los protocolos estudiados en el capítulo. En la primera práctica de laboratorio con Wireshark, tendrá que conseguir e instalar una copia de Wireshark, acceder a un sitio web y capturar y examinar los mensajes de protocolo que estén siendo intercambiados entre su navegador web y el servidor web.

Puede encontrar todos los detalles acerca de esta primera práctica de laboratorio con Wireshark (incluyendo las instrucciones acerca de cómo obtener e instalar Wireshark) en el sitio web <http://www.pearsonhighered.com/cs-resources/>.

UNA ENTREVISTA CON...

Leonard Kleinrock

Leonard Kleinrock es catedrático de Ciencias de la Computación en la Universidad de California en Los Ángeles. En 1969, su computadora en UCLA se convirtió en el primer nodo de Internet. Su definición de los principios de la conmutación de paquetes en 1961 se convirtió en la tecnología en la que hoy se basa Internet. Recibió su licenciatura en el City College de Nueva York (CCNY) y sus títulos de máster y doctor en Ingeniería Eléctrica en el MIT.



¿Qué fue lo que le decidió a especializarse en la tecnología de redes e Internet?

Como estudiante de doctorado en el MIT en 1959, me di cuenta de que la mayor parte de mis compañeros de clase estaban realizando sus investigaciones en el área de la teoría de la información y de la teoría de la codificación. En el MIT se encontraba el gran investigador Claude Shannon, quien había abierto estos campos de investigación y que ya había resuelto la mayor parte de los problemas importantes. Los problemas de investigación que quedaban eran muy complicados o de consecuencias mucho menos importantes. Así que decidí iniciarme en una nueva área en la que nadie había pensado todavía. Recuerde que, en el MIT, yo estaba rodeado de montones de computadoras y vi claramente que pronto esas máquinas necesitarían comunicarse entre sí. En aquel momento, no existía una forma efectiva de hacerlo, por lo que decidí desarrollar la tecnología que permitiera crear redes de datos eficientes y fiables.

¿Cuál fue su primer trabajo en la industria informática? ¿Qué significó para usted?

Entre 1951 y 1957 realicé en el CCNY los estudios de grado en Ingeniería Eléctrica en el turno de tarde. Durante el día, trabajé primero como técnico y luego como ingeniero en una pequeña empresa de electrónica industrial llamada Photobell. Mientras estuve allí, introduje la tecnología digital en sus líneas de productos. Básicamente, utilizábamos dispositivos fotoeléctricos para detectar la presencia de ciertos elementos (cajas, personas, etc.) y el uso de un circuito que por entonces se denominaba *multivibrador biestable* era la clase de tecnología que necesitábamos para llevar el procesamiento digital a este campo de la detección. Estos circuitos resultaron ser la base de las computadoras y ahora se conocen como *flip-flops*, *bistables* o *comutadores* en la jerga actual.

¿Qué pasó por su cabeza cuando envió el primer mensaje de un host a otro (desde UCLA al Stanford Research Institute)?

Francamente, no teníamos ni idea de la importancia de aquel suceso. No teníamos preparado un mensaje especial que pasara a la historia, como tantos otros inventores del pasado (Samuel Morse con “¡Lo que ha hecho Dios!”, Alexander Graham Bell con “Watson, ¡ven aquí! Te necesito” o Neal Armstrong con “Un pequeño paso para el hombre, pero un gran paso para la Humanidad”). ¡Aquellos tipos sí eran *inteligentes*! Conocían a los medios de comunicación y sabían lo que eran las relaciones públicas. Todo lo que nosotros queríamos hacer era iniciar una sesión en la computadora del SRI. De modo que escribimos “L”, lo que fue correctamente recibido, escribimos luego la letra “o”, que fue recibida, y después la letra “g”... ¡que hizo que la computadora host del SRI fallara estrepitosamente! Así que nuestro mensaje fue el más corto de la historia: “Lo”.

Anteriormente, aquel año, yo había sido citado en una nota de prensa de UCLA diciendo que una vez que la red estuviera activa y funcionando, sería posible acceder a las utilidades informáticas desde nuestros hogares y oficinas tan fácilmente como ya era disponer de electricidad y teléfono. Por tanto, mi visión en aquel momento era que Internet estaría en todas partes, siempre en funcionamiento, siempre disponible, y que cualquiera con cualquier dispositivo podría conectarse desde cualquier lugar y que sería invisible. Sin embargo, nunca pude prever que mi madre con 99 años utilizaría Internet, ¡y ya lo creo que la utilizó!

¿Cómo ve el futuro de las redes?

La parte fácil de la visión es predecir la propia infraestructura. Preveo que veremos una considerable implantación de la computación nómada, los dispositivos móviles y los espacios inteligentes. De hecho, la disponibilidad de dispositivos informáticos y de comunicaciones ligeros, baratos, de altas prestaciones y portátiles (combinada con la omnipresencia de Internet) nos ha permitido convertirnos en nómadas. La computación nómada hace referencia a la tecnología que permite a los usuarios finales ir de un lugar a otro, obteniendo acceso a los servicios de Internet de forma transparente, independientemente de por dónde viajen e independientemente del dispositivo que utilicen o que lleven consigo. La parte más complicada de esta visión de futuro es predecir las aplicaciones y servicios, que siempre nos han sorprendido de forma increíble (correo electrónico, tecnologías de búsqueda, la World Wide Web, los blogs, las redes sociales, la generación y compartición de música, fotografías y vídeos por parte de los usuarios, etc.). Nos encontramos en el amanecer de una nueva era de aplicaciones móviles sorprendentes e innovadoras, que podremos utilizar con nuestros dispositivos de mano.

La siguiente ola nos permitirá pasar del mundo virtual del ciberespacio al mundo físico de los espacios inteligentes. Nuestros entornos (mesas, paredes, vehículos, relojes, etc.) cobrarán vida con la tecnología, mediante actuadores, sensores, lógica, procesamiento, almacenamiento, cámaras, micrófonos, altavoces, pantallas y comunicación. Esta tecnología integrada permitirá a nuestro entorno proporcionar los servicios IP que deseemos. Cuando accedamos a una habitación, la habitación sabrá que hemos entrado. Podremos comunicarnos con nuestro entorno de forma natural, hablando en nuestra lengua materna; nuestras solicitudes generarán respuestas que nos presentarán páginas web en pantallas de pared, en los cristales de las gafas, en forma de texto hablado, de hologramas, etc.

Mirando un poco más lejos, preveo un futuro en red que incluya los siguientes componentes adicionales fundamentales. Veo agentes software inteligentes por toda la red, cuya función será hacer minería de datos, actuar de acuerdo con ellos, detectar tendencias y llevar a cabo tareas de forma dinámica y adaptativa. Preveo que habrá una cantidad de tráfico de red considerablemente mayor, generada no tanto por los seres humanos, sino por estos dispositivos integrados y esos agentes software inteligentes. Preveo que habrá grandes conjuntos de sistemas dotados de auto-organización y encargados de controlar esa red tan enorme y tan rápida. Preveo que habrá enormes cantidades de información viajando instantáneamente a través de esa red y viéndose sometida en el camino a enormes cantidades de procesamiento y de filtrado. Internet será, esencialmente, un sistema nervioso global que llegará a todas partes. Preveo que sucederán todas estas cosas y muchas más a medida que nos vayamos adentrando en el siglo XXI.

¿Qué personas le han inspirado profesionalmente?

Con diferencia, el que más me ha inspirado ha sido Claude Shannon del MIT, un brillante investigador que tenía la capacidad de relacionar sus ideas matemáticas con el mundo físico de una forma extremadamente intuitiva. Estuvo en el tribunal de mi tesis doctoral.

¿Tiene algún consejo para los estudiantes que se inician ahora en el campo de las redes y de Internet?

Internet y todo lo que esa red hace posible constituye una nueva frontera de grandes dimensiones, llena de desafíos asombrosos. Hay grandes oportunidades para la innovación y no hay que sentirse restringido por la tecnología actual. Lo que hay que hacer es abrir la mente e imaginar cómo podrían ser las cosas, y luego hacer que eso suceda.

La capa de aplicación

Las aplicaciones de red son la *razón de ser* de una red de computadoras: si no pudiéramos concebir ninguna aplicación útil, no existiría la necesidad de disponer de una infraestructura de red y de unos protocolos para darlas soporte. Desde la aparición de Internet, se han creado muchas aplicaciones de red útiles y entretenidas. Estas aplicaciones han sido la fuerza motriz del éxito de Internet, motivando a las personas en sus domicilios, escuelas, gobiernos y empresas a hacer de Internet una parte integrante de sus actividades cotidianas.

Entre las aplicaciones Internet se incluyen las clásicas aplicaciones basadas en texto que se hicieron populares en las décadas de 1970 y 1980, como son el correo electrónico de texto, el acceso remoto a computadoras, la transferencia de archivos y los grupos de noticias. También se incluye la aplicación por excelencia de mediados de la década de 1990: la World Wide Web, que incluye la navegación web, las búsquedas y el comercio electrónico. También se incluyen la mensajería instantánea y la compartición de archivos P2P, las dos aplicaciones estrella introducidas a finales del pasado milenio. En este siglo que comienza continúan apareciendo aplicaciones nuevas y atractivas, incluyendo voz sobre IP y sistemas de videoconferencia como Skype, Facetime y Google Hangouts; sistemas de vídeo generado por el usuario, como YouTube, y de películas a la carta, como Netflix; y juegos en línea multijugador como Second Life y World of Warcraft. Durante este mismo periodo, hemos visto el nacimiento de una nueva generación de aplicaciones de redes sociales (como Facebook, Instagram, Twitter y WeChat) que han creado adictivas redes de personas por encima de la red de routers y enlaces de comunicaciones de Internet. Y más recientemente, junto con la llegada del teléfono inteligente, ha surgido una profusión de aplicaciones móviles basadas en la ubicación, incluyendo populares aplicaciones de reservas, de citas y de previsiones de tráfico, como Yelp, Tinnder, Waz y Yik Yak. Claramente, no se ha ralentizado el ritmo de aparición de nuevas y excitantes aplicaciones Internet. ¡Quizá algunos de los lectores de este texto sean los que desarrollem la siguiente generación de aplicaciones que reinen en Internet!

En este capítulo vamos a estudiar los aspectos conceptuales y de implementación de las aplicaciones de red. Comenzaremos definiendo los conceptos fundamentales relativos a la capa de aplicación, incluyendo los servicios de red requeridos por las aplicaciones, los clientes y servidores, los procesos y las interfaces de la capa de transporte. Examinaremos en detalle varias aplicaciones

de red, como la Web, el correo electrónico, el sistema DNS, la distribución de archivos en redes entre pares (P2P, *Peer-to-Peer*) y los flujos de vídeo. (En el Capítulo 9 se examinarán más en detalle las aplicaciones multimedia, incluyendo los flujos de vídeo y VoIP.) A continuación, nos ocuparemos del desarrollo de las aplicaciones de red, tanto sobre TCP como UDP. En particular, estudiaremos la interfaz de sockets y echaremos un vistazo a algunas aplicaciones cliente-servidor simples implementadas en Python. También proporcionaremos al final del capítulo varias divertidas e interesantes tareas de programación de sockets.

La capa de aplicación es un lugar particularmente bueno para comenzar nuestro estudio de los protocolos, ya que es un terreno conocido. Estamos familiarizados con muchas de las aplicaciones que se basan en los protocolos que vamos a estudiar. Nos dará una idea adecuada de qué son los protocolos y nos servirá para introducir muchas de las cuestiones que tendremos que volver a ver cuando estudiemos los protocolos de las capas de transporte, de red y de enlace.

2.1 Principios de las aplicaciones de red

Imagine que se le ha ocurrido una idea para desarrollar una nueva aplicación de red. Es posible que esa aplicación llegue a prestar un gran servicio a la Humanidad, o que le guste a su profesor, o que le haga ganar una fortuna o que, simplemente, le divierta desarrollarla. Sea cual sea la motivación, a continuación vamos a ver cómo transformar la idea en una aplicación de red real.

Básicamente, el desarrollo de una aplicación de red implica escribir programas que se ejecuten en distintos sistemas terminales y que se comuniquen entre sí a través de la red. Por ejemplo, en la aplicación Web se emplean dos programas diferentes que se comunican entre sí: el navegador que se ejecuta en el host del usuario (una computadora de escritorio, un portátil, una tableta, un teléfono inteligente, etc.) y el programa del servidor web que se ejecuta en el host que actúa como servidor web. Otro ejemplo sería el caso de un sistema de compartición de archivos P2P, en el que se emplea un programa en cada host que participa en la comunidad de compartición de archivos. En este caso, los programas instalados en los distintos hosts pueden ser similares o idénticos.

Por tanto, al desarrollar su nueva aplicación tendrá que escribir software que se ejecutará en múltiples sistemas terminales. Este software podría escribirse, por ejemplo, en C, Java o Python. Una cuestión importante es que no es necesario escribir software que se ejecute en los dispositivos del núcleo de la red, como por ejemplo los routers o los switches de la capa de enlace. Incluso aunque deseara escribir software de aplicación para estos dispositivos del núcleo de la red, no podría hacerlo. Como hemos visto en el Capítulo 1 y se ilustra en la Figura 1.24, los dispositivos del núcleo de la red no operan en la capa de aplicación, sino en las capas situadas mas abajo: específicamente, en la capa de red e inferiores. Este diseño básico (que confina el software de aplicación a los sistemas terminales), como que se muestra en la Figura 2.1, ha facilitado el rápido desarrollo e implementación de una gran variedad de aplicaciones de red.

2.1.1 Arquitecturas de las aplicaciones de red

Antes de profundizar en la codificación del software, deberíamos disponer de una visión general de la arquitectura de la aplicación. Tenga en cuenta que la arquitectura de una aplicación es muy distinta de la arquitectura de la red (es decir, de la arquitectura de Internet de cinco capas vista en el Capítulo 1). Desde la perspectiva del desarrollador de aplicaciones, la arquitectura de la red es fija y proporciona un conjunto específico de servicios a las aplicaciones. Por otro lado, el desarrollador de aplicaciones diseña la **arquitectura de la aplicación**, que establece cómo debe estructurarse la aplicación en los distintos sistemas terminales. Al seleccionar la arquitectura de la aplicación, el desarrollador probablemente utilizará uno de los dos paradigmas arquitectónicos predominantes en las aplicaciones de red modernas: la **arquitectura cliente-servidor** o la **arquitectura P2P**.

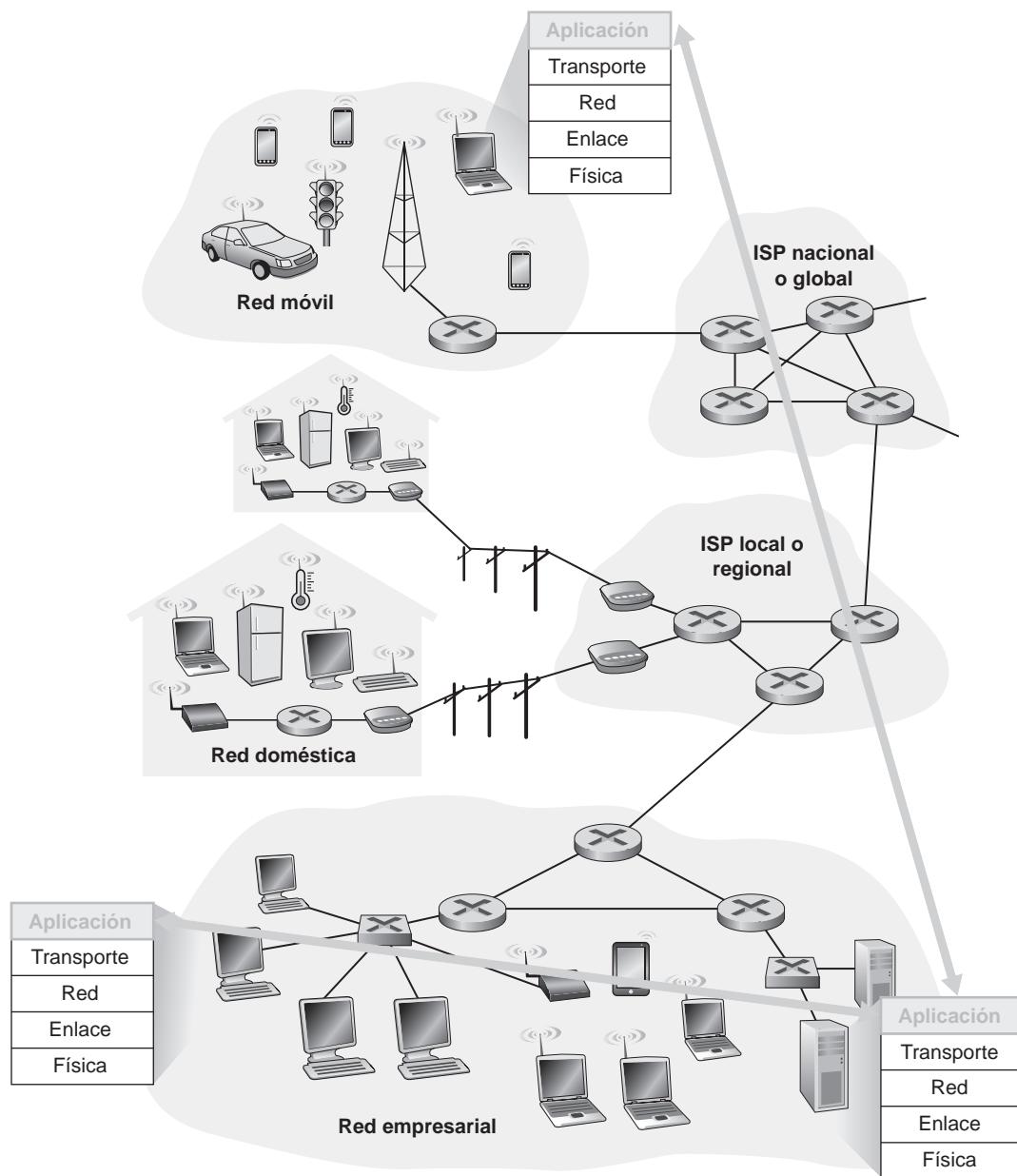


Figura 2.1 ♦ La comunicación de una aplicación de red tiene lugar entre sistemas terminales en la capa de aplicación.

En una **arquitectura cliente-servidor** existe un host siempre activo, denominado *servidor*, que da servicio a las solicitudes de muchos otros hosts, que son los *clientes*. Un ejemplo clásico es la Web, en la que un servidor web siempre activo sirve las solicitudes de los navegadores que se ejecutan en los hosts clientes. Cuando un servidor web recibe una solicitud de un objeto de un host cliente, responde enviándole el objeto solicitado. Observe que, con la arquitectura cliente-servidor, los clientes no se comunican directamente entre sí; por ejemplo, en la aplicación web, dos navegadores no se comunican de forma directa. Otra característica de la arquitectura cliente-servidor es que el servidor tiene una dirección fija y conocida, denominada dirección IP (de la que hablaremos enseguida). Puesto que el servidor tiene una dirección fija y conocida, y siempre está

activo, un cliente siempre puede contactar con él enviando un paquete a su dirección IP. Entre las aplicaciones más conocidas que utilizan la arquitectura cliente-servidor se encuentran la Web, FTP, Telnet y el correo electrónico. En la Figura 2.2(a) se muestra la arquitectura cliente-servidor.

A menudo, en una aplicación cliente-servidor un único host servidor es incapaz de responder a todas las solicitudes de sus clientes. Por ejemplo, el sitio de una red social popular puede verse rápidamente desbordado si sólo dispone de un servidor para gestionar todas las solicitudes. Por esta razón, en las arquitecturas cliente-servidor suele utilizarse un **centro de datos**, que alberga un gran número de hosts, para crear un servidor virtual de gran capacidad. Los servicios internet más populares –como los motores de búsqueda (p. ej. Google, Bing o Baidu), los sitios de comercio por Internet (p. ej. Amazon, e-Bay o Alibaba), el correo electrónico basado en la Web (p. ej. Gmail y Yahoo Mail) o las redes sociales (como Facebook, Instagram, Twitter y WeChat)– emplean uno o más centros de datos. Como se explica en la Sección 1.3.3, Google dispone de entre 30 y 50 centros de datos distribuidos por todo el mundo, que se encargan de gestionar colectivamente las búsquedas, YouTube, Gmail y otros servicios. Un centro de datos puede tener cientos de miles de servidores, a los que hay que suministrar energía y mantener. Además, los proveedores del servicio deben pagar los costes recurrentes de interconexión y de ancho de banda para poder enviar datos desde sus centros de datos.

En una **arquitectura P2P** existe una mínima (o ninguna) dependencia de una infraestructura de servidores dedicados situados en centros de datos. En su lugar, la aplicación explota la comunicación directa entre parejas de hosts conectados de forma intermitente, conocidos como pares (*peers*). Los pares no son propiedad del proveedor del servicio, sino que son computadoras de escritorio y portátiles controladas por los usuarios, encontrándose la mayoría de los pares en domicilios, universidades y oficinas. Puesto que los pares se comunican sin pasar por un servidor dedicado, la arquitectura se denomina arquitectura *peer-to-peer* (P2P). Muchas de las aplicaciones actuales más populares y con un mayor nivel de tráfico están basadas en arquitecturas P2P. Entre estas

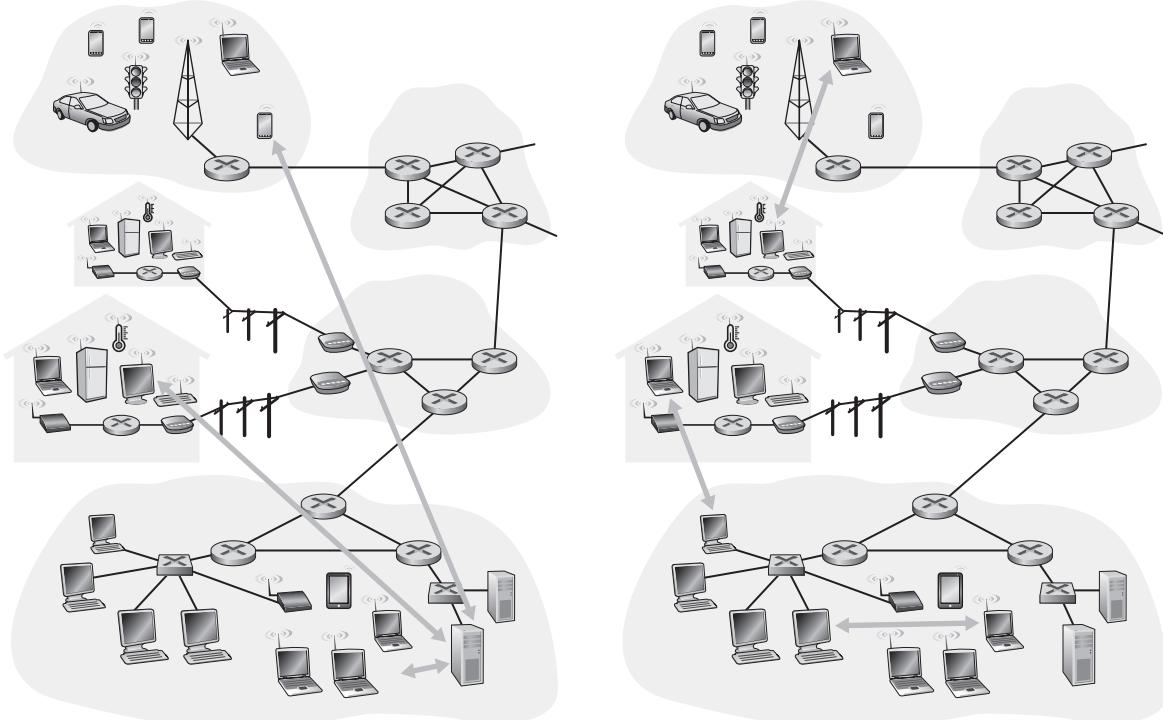


Figura 2.2 ♦ (a) Arquitectura cliente-servidor; (b) Arquitectura P2P.

aplicaciones se incluyen la compartición de archivos (por ejemplo, BitTorrent), la aceleración de descarga con ayuda de pares (por ejemplo, Xunlei) y la telefonía y videoconferencia por Internet (por ejemplo, Skype). En la Figura 2.2(b) se ilustra la arquitectura P2P. Conviene mencionar que algunas aplicaciones tienen arquitecturas híbridas, que combinan elementos cliente-servidor y P2P. Por ejemplo, en muchas aplicaciones de mensajería instantánea, los servidores se utilizan para hacer un seguimiento de las direcciones IP de los usuarios, pero los mensajes de usuario a usuario se envían directamente entre los hosts de dichos usuarios (sin pasar por servidores intermedios).

Una de las características más convincentes de las arquitecturas P2P es su **auto-escalabilidad**. Por ejemplo, en una aplicación de compartición de archivos P2P, aunque cada par genera una carga de trabajo solicitando archivos, también añade capacidad de servicio al sistema, distribuyendo archivos a otros pares. Las arquitecturas P2P también presentan una buena relación coste-prestaciones, ya que normalmente no requieren una infraestructura de servidores significativa ni un gran ancho de banda de servidor (a diferencia de los diseños cliente-servidor con centros de datos). Sin embargo, las aplicaciones P2P plantean problemas de seguridad, rendimiento y fiabilidad, debido a su naturaleza altamente descentralizada.

2.1.2 Comunicación entre procesos

Antes de crear su aplicación de red, también necesita disponer de unos conocimientos básicos sobre cómo se comunican entre sí los programas que se ejecutan en varios sistemas terminales. En la jerga de los sistemas operativos, realmente los que se comunican no son programas, sino **procesos**. Un proceso puede interpretarse como un programa que se ejecuta dentro de un sistema terminal. Cuando los procesos se ejecutan en el mismo sistema terminal, pueden comunicarse entre sí mediante sistemas de comunicación inter-procesos, aplicando reglas gobernadas por el sistema operativo del sistema terminal. Pero, en este libro, no estamos especialmente interesados en cómo se comunican los procesos dentro de un mismo host, sino en cómo se comunican los procesos que se ejecutan en hosts *diferentes* (con sistemas operativos potencialmente diferentes).

Los procesos de dos sistemas terminales diferentes se comunican entre sí intercambiando mensajes a través de la red de computadoras. Un proceso emisor crea y envía mensajes a la red; un proceso receptor recibe estos mensajes y posiblemente responde devolviendo otros mensajes. La Figura 2.1 ilustra que los procesos que se comunican entre sí residen en la capa de aplicación de la pila de protocolos de cinco capas.

Procesos cliente y servidor

Una aplicación de red consta de parejas de procesos que se envían mensajes entre sí a través de una red. Por ejemplo, en la aplicación Web un proceso navegador de un cliente intercambia mensajes con un proceso de un servidor web. En un sistema de compartición de archivos P2P, se transfiere un archivo desde un proceso que se ejecuta en un par, a un proceso de otro par. Normalmente, en una pareja de procesos que están comunicándose, designamos a uno de los procesos como el **cliente** y al otro como el **servidor**. En la Web, un navegador es un proceso cliente y el servidor web es un proceso servidor. En la compartición de archivos P2P, el par que descarga el archivo se designa como el cliente y el host que está cargando el archivo se designa como el servidor.

Es posible que haya observado que en algunas aplicaciones, tales como la compartición de archivos P2P, un proceso puede ser tanto un cliente como un servidor. De hecho, un proceso en un sistema de compartición de archivos P2P puede cargar y descargar archivos. No obstante, en el contexto de cualquier sesión de comunicación específica entre una pareja de procesos, seguimos pudiendo designar a uno de los procesos como el cliente y al otro como el servidor. Definimos los procesos cliente y servidor como sigue:

En el contexto de una sesión de comunicación entre una pareja de procesos, el proceso que inicia la comunicación (es decir, que inicialmente se pone en contacto con el otro proceso al

principio de la sesión) se designa como el cliente. El proceso que espera a ser contactado para comenzar la sesión es el servidor.

En la Web, un proceso navegador inicia el contacto con un proceso servidor web; por tanto, el proceso navegador es el cliente y el proceso servidor web es el servidor. En la compartición de archivos P2P, cuando un par A pide a un par B que le envíe un determinado archivo, el A es el cliente y el B es el servidor en el contexto de esta sesión de comunicación concreta. En ocasiones, cuando no exista ningún tipo de ambigüedad, también emplearemos la terminología “lado del cliente y lado del servidor de una aplicación”. Al final del capítulo, examinaremos paso a paso un código simple tanto para el lado del cliente como para el lado del servidor de las aplicaciones de red.

Interfaz entre el proceso y la red de computadoras

Como hemos mencionado, la mayoría de las aplicaciones constan de parejas de procesos que se comunican, intercambiándose mensajes. Cualquier mensaje enviado de un proceso al otro debe atravesar la red subyacente. **Un proceso envía mensajes a la red y los recibe de la red a través de una interfaz software denominada socket.** Veamos una analogía que nos ayudará a comprender los conceptos de proceso y socket. **Un proceso es análogo a una casa y un socket es análogo a la puerta de la casa.** Cuando un proceso desea enviar un mensaje a otro proceso que se está ejecutando en otro host, envía el mensaje a través de su propia puerta (socket). Este proceso emisor supone que existe una infraestructura de transporte al otro lado de la puerta que llevará el mensaje hasta la puerta del proceso de destino. Una vez que el mensaje llega al host de destino, éste pasa a través de la puerta (socket) del proceso receptor, actuando entonces el proceso receptor sobre el mensaje.

La Figura 2.3 ilustra la comunicación mediante sockets entre dos procesos que se comunican a través de Internet. (En la Figura 2.3 se supone que el protocolo de transporte subyacente utilizado por los procesos es el protocolo TCP de Internet.) Como se muestra en la figura, un socket es la **interfaz entre la capa de aplicación y la capa de transporte de un host.** También se conoce como **interfaz de programación de aplicaciones (API, Application Programming Interface)** entre la aplicación y la red, ya que el socket es la interfaz de programación con la que se construyen las aplicaciones de red. El desarrollador de la aplicación tiene el control sobre todos los elementos del lado de la capa de aplicación del socket, pero apenas tiene control sobre el lado de la capa de transporte del socket. El único control que tiene el desarrollador de la aplicación sobre el lado de la capa de transporte es (1) la elección del protocolo de transporte y (2) quizás la capacidad de fijar unos pocos parámetros de la capa de transporte, como por ejemplo los tamaños máximo del buffer

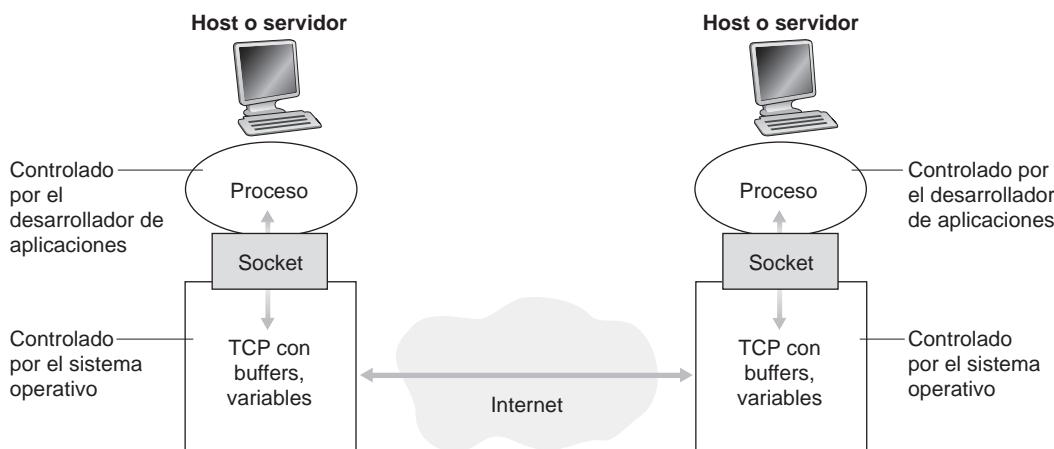


Figura 2.3 ♦ Procesos de aplicación, sockets y protocolo de transporte subyacente.

y de segmento (lo que veremos en el Capítulo 3). Una vez que el desarrollador de la aplicación ha seleccionado un protocolo de transporte (si hay disponibles varios entre los que elegir), la aplicación se construye utilizando los servicios de la capa de transporte proporcionados por dicho protocolo. En las Secciones 2.7 exploraremos los sockets con un cierto grado de detalle.

Direccionamiento de procesos

Para enviar una carta por correo postal a un destino concreto, ese destino necesita disponer de una dirección. De forma similar, para que un proceso que se está ejecutando en un host pueda enviar paquetes a otro proceso que se ejecuta en un host distinto, el proceso receptor necesita disponer de una dirección. Para identificar al proceso receptor, tienen que especificarse dos elementos de información: (1) la dirección del host y (2) un identificador que especifique el proceso de recepción en el host de destino.

En Internet, el host se identifica mediante su dirección IP. En el Capítulo 4 veremos en detalle las direcciones IP. Por el momento, todo lo que necesitamos saber es que una dirección IP es una magnitud de 32 bits que identifica de forma única a un host. Además de conocer la dirección del host al que está destinado un mensaje, el host emisor también debe identificar el proceso receptor (o, para ser más exactos, el socket receptor) que está ejecutándose en el host. Esta información es necesaria porque, en general, un host podría estar ejecutando muchas aplicaciones de red. Para satisfacer este requisito, se utiliza un número de puerto de destino. Las aplicaciones populares tienen asignados números de puerto específicos. Por ejemplo, un servidor web queda identificado por el número de puerto 80. Un proceso servidor de correo (que utilice el protocolo SMTP) se identifica mediante el número de puerto 25. Puede encontrar una lista de números de puerto bien conocidos para todos los protocolos estándar de Internet en <http://www.iana.org>. Examinaremos los números de puerto en detalle en el Capítulo 3.

2.1.3 Servicios de transporte disponibles para las aplicaciones

Recordemos que un socket es la interfaz entre el proceso de la aplicación y el protocolo de la capa de transporte. La aplicación del lado emisor envía los mensajes a través del socket. En el otro lado del socket, el protocolo de la capa de transporte tiene la responsabilidad de llevar los mensajes hasta el socket del proceso receptor.

Muchas redes, incluyendo Internet, proporcionan más de un protocolo de la capa de transporte. Cuando vaya a desarrollar una aplicación, tendrá que elegir uno de los protocolos de la capa de transporte disponibles. ¿Cómo llevar a cabo esta selección? Muy probablemente, tendrá que estudiar los servicios que ofrecen los protocolos de la capa de transporte disponibles y después elegir aquel protocolo que proporcione los servicios que mejor se adapten a las necesidades de la aplicación. La situación es similar a tener que elegir entre viajar en tren o en avión para ir de una ciudad a otra. Tiene que elegir un medio de transporte u otro, y cada uno de ellos ofrece servicios diferentes (por ejemplo, el tren le ofrece partir y llegar al centro de las ciudades, mientras que el avión ofrece un tiempo de viaje más corto).

¿Cuáles son los servicios que puede ofrecer un protocolo de la capa de transporte a las aplicaciones que lo invocan? Podemos clasificar los posibles servicios de forma bastante general según cuatro parámetros: transferencia de datos fiable, tasa de transferencia, temporización y seguridad.

Transferencia de datos fiable

Como se ha explicado en el Capítulo 1, en una red de computadoras pueden perderse paquetes. Por ejemplo, un paquete puede desbordar el buffer de un router, o podría ser descartado por un host o un router después de comprobar que algunos de sus bits están corrompidos. En muchas aplicaciones (como el correo electrónico, la transferencia de archivos, el acceso remoto a hosts, la transferencia

de documentos web y las aplicaciones financieras) la pérdida de datos puede tener consecuencias catastróficas (¡en el último caso, para el banco o para el cliente!). Por tanto, para dar soporte a estas aplicaciones, es preciso hacer algo para garantizar que los datos enviados desde un terminal de la aplicación sean todos ellos entregados correcta y completamente al otro terminal de la aplicación. Si un protocolo proporciona un servicio de entrega de datos garantizado, se dice que proporciona una transferencia de datos fiable. Un servicio importante que un protocolo de la capa de transporte puede potencialmente proporcionar a una aplicación es la transferencia de datos fiable proceso a proceso. Cuando un protocolo de transporte suministra este servicio, el proceso emisor puede pasar sus datos al socket y saber con certidumbre absoluta que los datos llegarán sin errores al proceso receptor.

Si un protocolo de la capa de transporte no proporciona una transferencia de datos fiable, los datos enviados por el proceso emisor pueden no llegar nunca al proceso de recepción. Esto puede ser aceptable para las **aplicaciones tolerantes a pérdidas**; por ejemplo, la mayor parte de las aplicaciones multimedia, como las de audio/vídeo en tiempo real, pueden tolerar que cierta cantidad de datos se pierda. En estas aplicaciones multimedia, la pérdida de datos puede dar lugar a una pequeña interrupción al reproducir el audio/vídeo, lo que no constituye un problema fundamental.

Tasa de transferencia

En el Capítulo 1 hemos presentado el concepto de tasa de transferencia disponible, que en el contexto de una sesión de comunicaciones entre dos procesos a lo largo de una ruta de red, es la tasa a la que el proceso emisor puede suministrar bits al proceso de recepción. Puesto que otras sesiones compartirán el ancho de banda a lo largo de la ruta de red y puesto que esas otras sesiones se iniciarán y terminarán aleatoriamente, la tasa de transferencia disponible puede fluctuar con el tiempo. Estas observaciones nos llevan de forma natural a otro servicio que un protocolo de la capa de transporte podría proporcionar: una tasa de transferencia disponible garantizada a un cierta velocidad especificada. Con un servicio así, la aplicación podría solicitar una tasa de transferencia garantizada de r bits/segundo y el protocolo de transporte podría entonces garantizar que la tasa de transferencia disponible sea siempre al menos de r bits/segundo. Un servicio que ofreciera una tasa de transferencia garantizada como esta resultaría atractivo para muchas aplicaciones. Por ejemplo, si una aplicación de telefonía por Internet codifica voz a 32 kbps, tendrá que enviar datos a la red y tendrá que entregar los datos a la aplicación receptora a esa velocidad. Si el protocolo de transporte no puede proporcionar esa tasa de transferencia, la aplicación tendrá que realizar la codificación a una velocidad menor (y recibir a una tasa de transferencia adecuada como para mantener esa velocidad de codificación más lenta) o bien tendrá que renunciar, puesto que recibir a la mitad de la tasa de transferencia necesaria no tiene ninguna utilidad para esta aplicación de telefonía por Internet. Las aplicaciones con requisitos de tasa de transferencia se conocen como **aplicaciones sensibles al ancho de banda**. Muchas aplicaciones multimedia actuales son sensibles al ancho de banda, pero algunas de ellas pueden emplear técnicas de codificación adaptativa para codificar la voz o el vídeo digitalizados a una velocidad que se adapte a la tasa de transferencia disponible en cada momento.

Mientras que las aplicaciones sensibles al ancho de banda tienen requisitos específicos de tasa de transferencia, las denominadas **aplicaciones elásticas** pueden hacer uso de la tasa de transferencia, mucha o poca, que haya disponible. El correo electrónico, la transferencia de archivos y las transferencias web son todas ellas aplicaciones elásticas. Por supuesto, cuanto mayor sea la tasa de transferencia, mejor. (¡Como reza el dicho, es imposible ser demasiado rico, demasiado guapo o tener demasiada tasa de transferencia!)

Temporización

Un protocolo de la capa de transporte también puede proporcionar garantías de temporización. Al igual que con las tasas de transferencia garantizadas, las garantías de temporización también pueden darse de diversas formas. Un ejemplo de garantía podría ser que cada bit que el emisor empuja

por su socket llegue al socket del receptor en no más de 100 milisegundos. Un servicio así sería atractivo para las aplicaciones interactivas en tiempo real, como la telefonía por Internet, los entornos virtuales, la teleconferencia y los juegos multijugador, todas las cuales requieren restricciones de temporización muy estrictas sobre la entrega de datos para ser efectivas. (Véase el Capítulo 9 y [Gauthier 1999; Ramjee 1994].) Por ejemplo, los retardos largos en la telefonía por Internet tienden a dar lugar a pausas antinaturales en una conversación; en un juego multijugador o en un entorno virtual interactivo, un retardo largo entre la realización de una acción y la visualización de la respuesta del entorno (por ejemplo, de otro jugador que se encuentra en el otro extremo de una conexión extremo a extremo) hace que la aplicación parezca menos realista. En las aplicaciones que no se ejecutan en tiempo real, siempre es preferible un retardo pequeño que uno grande, pero no se aplican restricciones estrictas a los retardos extremo a extremo.

Seguridad

Por último, un protocolo de transporte puede proporcionar a una aplicación uno o más servicios de seguridad. Por ejemplo, en el host emisor, un protocolo de transporte puede cifrar todos los datos transmitidos por el proceso emisor, y en el host receptor puede descifrar los datos antes de entregarlos al proceso receptor. Un servicio así proporcionaría confidencialidad entre los dos procesos, incluso aunque los datos sean observados de alguna manera entre los procesos emisor y receptor. Un protocolo de transporte también puede proporcionar otros servicios de seguridad además del de la confidencialidad, como pueden ser mecanismos para garantizar la integridad de los datos y mecanismos de autenticación en el punto terminal, temas que abordaremos en detalle en el Capítulo 8.

2.1.4 Servicios de transporte proporcionados por Internet

Hasta el momento hemos considerado los servicios de transporte que una red de computadoras podría proporcionar en general. Seamos ahora un poco más específicos y examinemos los tipos de servicio de transporte que Internet proporciona a las aplicaciones. Internet (y, de forma más general, las redes TCP/IP) pone a disposición de las aplicaciones dos protocolos de transporte: UDP y TCP. Cuando usted, como desarrollador de aplicaciones, cree una nueva aplicación de red para Internet, una de las primeras decisiones que tendrá que tomar es si utilizar UDP o TCP. Cada uno de estos protocolos ofrece un conjunto diferente de servicios a las aplicaciones que los invocan. La Figura 2.4 detalla los requisitos de servicio para algunas aplicaciones seleccionadas.

Aplicación	Pérdida de datos	Tasa de transferencia	Sensible al tiempo
Transferencia/descarga de archivos	Sin pérdidas	Elástica	No
Correo electrónico	Sin pérdidas	Elástica	No
Documentos web	Sin pérdidas	Elástica (pocos kbps)	No
Telefonía por Internet/ Videoconferencia	Tolerante a las pérdidas	Audio: unos pocos kbps–1 Mbps Vídeo: 10 kbps–5 Mbps	Sí: décimas de segundo
Flujos de audio/vídeo almacenado	Tolerante a las pérdidas	Como el anterior	Sí: unos pocos segundos
Juegos interactivos	Tolerante a las pérdidas	Unos pocos kbps–10 kbps	Sí: décimas de segundo
Mensajería para teléfono inteligente	Sin pérdidas	Elástica	Sí y no

Figura 2.4 ♦ Requisitos de algunas aplicaciones de red seleccionadas.

Servicios TCP

El modelo de servicio TCP incluye un servicio orientado a la conexión y un servicio de transferencia de datos fiable. Cuando una aplicación invoca TCP como su protocolo de transporte, la aplicación recibe ambos servicios de TCP.

- *Servicio orientado a la conexión.* TCP hace que el cliente y el servidor intercambien entre sí información de control de la capa de transporte *antes* de que empiecen a fluir los mensajes del nivel de aplicación. Este procedimiento denominado de negociación, de reconocimiento o de establecimiento de la conexión, alerta al cliente y al servidor, permitiéndoles prepararse para el intercambio de paquetes. Después de esta fase de negociación, se dice que existe una **conexión TCP** entre los sockets de los dos procesos. La conexión es una conexión full-duplex, en el sentido de que los dos procesos pueden enviar mensajes entre sí a través de la conexión al mismo tiempo. Una vez que la aplicación ha terminado de enviar mensajes, debe **terminar la conexión**. En el Capítulo 3 examinaremos en detalle los servicios orientados a la conexión y veremos cómo se implementan.
- *Servicio de transferencia de datos fiable.* Los procesos que se están comunicando pueden confiar en TCP para entregar todos los datos enviados sin errores y en el orden correcto. Cuando un lado de la aplicación pasa un flujo de bytes a un socket, puede contar con TCP para entregar el mismo flujo de bytes al socket receptor sin pérdida ni duplicación de bytes.

TCP también incluye un mecanismo de control de congestión, que es un servicio para mejorar el funcionamiento general de Internet, más que para el beneficio directo de los procesos que se comunican. Este mecanismo de control de congestión de TCP regula el proceso emisor (cliente o servidor) cuando la red está congestionada entre el emisor y el receptor. Como se explica en el Capítulo 3, el control de congestión de TCP también intenta limitar cada conexión TCP para que utilice una cuota equitativa de ancho de banda de la red.

SEGURIDAD

TCP SEGURO

Ni TCP ni UDP proporcionan ningún mecanismo de cifrado (los datos que el proceso emisor pasa al socket son los mismos datos que viajan a través de la red hasta el proceso de destino). Luego, por ejemplo, si el proceso emisor envía una contraseña en texto legible (es decir, no cifrada) a su socket, esa contraseña viajará a través de todos los enlaces entre el emisor y el receptor, pudiendo ser "husmeada" y descubierta en cualquiera de los enlaces intervinientes. Puesto que la confidencialidad y otras cuestiones de seguridad son críticas para muchas aplicaciones, la comunidad de Internet ha desarrollado una mejora para TCP, denominada **SSL (Secure Sockets Layer, Capa de conectores seguros)**. TCP mejorado con SSL no sólo hace todo lo que hace el protocolo TCP tradicional, sino que también proporciona servicios críticos de seguridad proceso a proceso, entre los que se incluyen mecanismos de cifrado, de integridad de los datos y de autenticación en el punto terminal. Debemos destacar que SSL no es un tercer protocolo de transporte de Internet, al mismo nivel que TCP y UDP, sino que es una mejora de TCP, que se implementa en la capa de aplicación. En concreto, si una aplicación desea utilizar los servicios de SSL, tiene que incluir código SSL (existen clases y librerías enormemente optimizadas) tanto en el lado del cliente como en el del servidor de la aplicación. SSL tiene su propia API de sockets, que es similar a la API de sockets del protocolo TCP tradicional. Cuando una aplicación utiliza SSL, el proceso emisor pasa los datos en texto legible al socket SSL; a continuación, SSL cifra los datos en el host emisor y los pasa al socket TCP. Los datos cifrados viajan a través de Internet hasta el socket TCP del proceso receptor. El socket de recepción pasa los datos cifrados a SSL, que los descifra. Por último, SSL pasa los datos en texto legible a través de su socket al proceso receptor. En el Capítulo 8 se cubre con un cierto detalle SSL.

Servicios UDP

UDP es un protocolo de transporte ligero y simple que proporciona unos servicios mínimos. No está orientado a la conexión, por lo que no tiene lugar un procedimiento de negociación antes de que los dos procesos comiencen a comunicarse. UDP proporciona un servicio de transferencia de datos no fiable; es decir, cuando un proceso envía un mensaje a un socket UDP, el protocolo UDP no ofrece ninguna garantía de que el mensaje vaya a llegar al proceso receptor. Además, los mensajes que sí llegan al proceso receptor pueden hacerlo de manera desordenada.

UDP no incluye tampoco un mecanismo de control de congestión, por lo que el lado emisor de UDP puede introducir datos en la capa inferior (la capa de red) a la velocidad que le parezca. (Sin embargo, tenga en cuenta que la tasa de transferencia extremo a extremo real puede ser menor que esta velocidad, a causa de la capacidad de transmisión limitada de los enlaces intervinientes o a causa de la congestión.)

Servicios no proporcionados por los protocolos de transporte de Internet

Hemos organizado los posibles servicios del protocolo de transporte según cuatro parámetros: transferencia de datos fiable, tasa de transferencia, temporización y seguridad. ¿Cuáles de estos servicios proporcionan TCP y UDP? Ya hemos mencionado que TCP proporciona transferencia de datos fiable extremo a extremo. Y también sabemos que TCP se puede mejorar fácilmente en la capa de aplicación, con SSL, para proporcionar servicios de seguridad. Pero en esta breve descripción de TCP y UDP hemos omitido notoriamente hacer mención de las garantías relativas a la tasa de transferencia o la temporización, servicios que *no* proporcionan los protocolos de transporte de Internet de hoy día. ¿Significa esto que las aplicaciones sensibles al tiempo, como la telefonía por Internet, no se pueden ejecutar actualmente en Internet? Evidentemente, la respuesta es no: Internet lleva muchos años albergando aplicaciones sensibles al tiempo. Estas aplicaciones suelen funcionar bastante bien, porque han sido diseñadas para hacer frente a esta falta de garantías de la mejor forma posible. En el Capítulo 9 veremos algunos de estos trucos de diseño. No obstante, un diseño inteligente tiene sus limitaciones cuando el retardo es excesivo, o cuando la tasa de transferencia extremo a extremo es limitada. En resumen, actualmente Internet puede ofrecer servicios satisfactorios a las aplicaciones sensibles al tiempo, pero no puede proporcionar ninguna garantía de tasa de transferencia ni de temporización.

La Figura 2.5 enumera los protocolos de transporte utilizados por algunas aplicaciones populares de Internet. Podemos ver que aplicaciones como el correo electrónico, el acceso remoto a terminales, la Web y la transferencia de archivos utilizan TCP. Estas aplicaciones han elegido TCP principalmente porque este protocolo ofrece un servicio de transferencia de datos fiable, garantizando que todos los datos llegarán finalmente a su destino. Como las aplicaciones de telefonía por Internet (como Skype) suelen tolerar cierto grado de pérdidas, pero requieren una tasa de transferencia mínima para ser efectivas, los desarrolladores de aplicaciones de telefonía por Internet suelen preferir ejecutarlas sobre UDP, evitando así el mecanismo de control de congestión de TCP y la mayor sobrecarga (bits que no forman parte de la carga útil) que los paquetes de datos tienen en TCP. Pero como muchos cortafuegos están configurados para bloquear el tráfico UDP (o la mayor parte del mismo), las aplicaciones de telefonía por Internet suelen diseñarse para usar TCP como solución alternativa, cuando falla la comunicación a través de UDP.

2.1.5 Protocolos de la capa de aplicación

Acabamos de aprender que los procesos de red se comunican entre sí enviando mensajes a sus sockets. Pero, ¿cómo están estructurados estos mensajes? ¿Cuál es el significado de cada uno de los campos de estos mensajes? ¿Cuándo envían los procesos estos mensajes? Estas preguntas nos llevan al ámbito de los protocolos de la capa de aplicación. Un protocolo de la capa de aplicación define cómo los procesos de una aplicación, que se ejecutan en distintos sistemas terminales, se pasan los mensajes entre sí. En particular, un protocolo de la capa de aplicación define:

Aplicación	Protocolo de la capa de aplicación	Protocolo de transporte subyacente
Correo electrónico	SMTP [RFC 5321]	TCP
Acceso remoto a terminal	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
Transferencia de archivos	FTP [RFC 959]	TCP
Flujos multimedia	HTTP (p. ej. YouTube)	TCP
Telefonía por Internet	SIP [rfc 3261], RTP [RFC 3550] o propietario (p. ej. Skype)	UDP o TCP

Figura 2.5 ♦ Aplicaciones populares de Internet, sus protocolos de la capa de aplicación y sus protocolos de transporte subyacentes.

- Los tipos de mensajes intercambiados; por ejemplo, mensajes de solicitud y mensajes de respuesta.
- La sintaxis de los diversos tipos de mensajes, es decir, los campos de los que consta el mensaje y cómo se delimitan esos campos.
- La semántica de los campos, es decir, el significado de la información contenida en los campos.
- Las reglas para determinar cuándo y cómo un proceso envía mensajes y responde a los mismos.

Algunos protocolos de la capa de aplicación están especificados en documentos RFC y, por tanto, son de dominio público. Por ejemplo, el protocolo de la capa de aplicación para la Web, HTTP (*HyperText Transfer Protocol* [RFC 2616]), está disponible como un RFC. Si quien desarrolla un navegador web sigue las reglas dadas en el RFC que se ocupa de HTTP, el navegador podrá recuperar páginas web de cualquier servidor web que también se ajuste a las reglas de dicho RFC. Existen muchos otros protocolos de la capa de aplicación que son propietarios y que intencionalmente no están disponibles para todo el mundo. Por ejemplo, Skype utiliza protocolos de la capa de aplicación propietarios.

Es importante diferenciar entre aplicaciones de red y protocolos de la capa de aplicación. Un protocolo de la capa de aplicación es únicamente un elemento de una aplicación de red (¡aunque uno muy importante, desde nuestro punto de vista!). Veamos un par de ejemplos. La Web es una aplicación cliente-servidor que permite a los usuarios obtener documentos almacenados en servidores web bajo demanda. La aplicación Web consta de muchos componentes, entre los que se incluyen un estándar para los formatos de documentos (es decir, HTML), navegadores web (como Firefox y Microsoft Internet Explorer), servidores web (por ejemplo, servidores Apache y Microsoft) y un protocolo de la capa de aplicación. El protocolo de la capa de aplicación de la Web, HTTP, define el formato y la secuencia de los mensajes que se pasan entre el navegador web y el servidor web. Por tanto, HTTP es sólo una pieza (aunque una pieza importante) de la aplicación Web. Otro ejemplo sería una aplicación de correo electrónico Internet, la cual también está constituida por muchos componentes, entre los que se incluyen los servidores de correo que albergan los buzones de los usuarios; los clientes de correo (como Microsoft Outlook) que permiten a los usuarios leer y crear mensajes; un estándar para definir la estructura de los mensajes de correo electrónico y protocolos de la capa de aplicación que definen cómo se pasan los mensajes entre los servidores, cómo se pasan los mensajes entre los servidores y los clientes de correo y cómo se interpretan los contenidos de las cabeceras de los mensajes. El principal protocolo de la capa de aplicación para el correo electrónico es SMTP (*Simple Mail Transfer Protocol*, Protocolo simple de transferencia de correo) [RFC 5321].

Por tanto, el protocolo principal de la capa de aplicación para correo electrónico, SMTP, sólo es un componente (aunque un componente importante) de la aplicación de correo electrónico.

2.1.6 Aplicaciones de red analizadas en este libro

Todos los días se desarrollan nuevas aplicaciones de Internet, tanto de dominio público como propietarias. En lugar de abordar un gran número de aplicaciones de Internet a modo de enciclopedia, hemos decidido centrarnos en unas pocas aplicaciones dominantes e importantes. En este capítulo abordaremos cinco aplicaciones relevantes: la Web, el correo electrónico, el servicio de directorio, los flujos de vídeo y las aplicaciones P2P. En primer lugar veremos la Web, no sólo porque es una aplicación enormemente popular, sino porque también su protocolo de la capa de aplicación, HTTP, es sencillo y fácil de comprender. Después veremos el correo electrónico, que fue la primera aplicación de éxito en Internet. El correo electrónico es más complejo que la Web, en el sentido de que no utiliza uno sino varios protocolos de la capa de aplicación. Después del correo electrónico, abordaremos el sistema DNS, que proporciona un servicio de directorio a Internet. La mayoría de los usuarios no interactúan directamente con DNS; en su lugar, invocan indirectamente a DNS a través de otras aplicaciones (entre las que se incluyen las aplicaciones web, de transferencia de archivos y de correo electrónico). DNS ilustra de forma muy conveniente cómo puede implementarse en la capa de aplicación de Internet un elemento de la funcionalidad de red básica (la traducción entre nombres de red y direcciones de red). Después examinaremos las aplicaciones P2P de compartición de archivos y completaremos nuestro estudio de las aplicaciones analizando los flujos de vídeo a la carta, incluyendo la distribución de vídeo almacenado a través de redes de distribución de contenido. En el Capítulo 9 hablaremos más en detalle de las aplicaciones multimedia, incluyendo la de voz sobre IP y la videoconferencia.

2.2 La Web y HTTP

Hasta principios de la década de 1990, Internet era utilizada principalmente por investigadores, profesores y estudiantes universitarios para acceder a hosts remotos; para transferir archivos desde los hosts locales a los hosts remotos, y viceversa, y para recibir y enviar noticias y mensajes de correo electrónico. Aunque estas aplicaciones eran (y continúan siendo) extremadamente útiles, Internet era prácticamente desconocida fuera de las comunidades académica y de investigación. Fue entonces, a principios de la década de 1990, cuando una nueva aplicación importante apareció en escena: la World Wide Web [Berners-Lee 1994]. La Web fue la primera aplicación de Internet que atrajo la atención del público general. Cambió de manera dramática, y continúa cambiando, la forma en que las personas interactúan dentro y fuera de sus entornos de trabajo. Hizo que Internet pasará de ser una de entre muchas redes de datos, a ser prácticamente la única red de datos.

Quizá lo que atrae a la mayoría de los usuarios es que la Web opera *bajo demanda*. Los usuarios reciben lo que desean y cuando lo desean. Es muy diferente a la radio y la televisión, que fuerzan a los usuarios a sintonizar los programas cuando el proveedor de contenido tiene el contenido disponible. Además de estar disponible bajo demanda, la Web posee muchas otras características maravillosas que a todo el mundo le gustan y que todos valoran. Para cualquier persona, es tremadamente fácil publicar información en la Web (todo el mundo puede convertirse en editor con unos costes extremadamente bajos). Los hipervínculos y los motores de búsqueda nos ayudan a navegar a través de un océano de sitios web. Las fotografías y los vídeos estimulan nuestros sentidos. Los formularios, JavaScript, los applets de Java y muchos otros mecanismos nos permiten interactuar con las páginas y sitios. Y la Web y sus protocolos sirven como plataforma para YouTube, para el correo electrónico basado en la Web (como Gmail) y para la mayoría de las aplicaciones móviles de Internet, incluyendo Instagram y Google Maps.

2.2.1 Introducción a HTTP

El corazón de la Web lo forma el **Protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol)**, que es el protocolo de la capa de aplicación de la Web. Está definido en los documentos [RFC 1945] y [RFC 2616]. HTTP se implementa mediante dos programas: un programa cliente y un programa servidor. Ambos programas, que se ejecutan en sistemas terminales diferentes, se comunican entre sí intercambiando mensajes HTTP. HTTP define la estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes. Antes de explicar en detalle HTTP, vamos a hacer un breve repaso de la terminología Web.

Una **página web** (también denominada documento web) consta de objetos. Un objeto es simplemente un archivo (como por ejemplo un archivo HTML, una imagen JPEG, un applet Java o un clip de vídeo) que puede direccionarse mediante un único URL. La mayoría de las páginas web están constituidas por un **archivo base HTML** y varios objetos referenciados. Por ejemplo, si una página web contiene texto HTML y cinco imágenes JPEG, entonces la página web contiene seis objetos: el archivo base HTML y las cinco imágenes. El archivo base HTML hace referencia a los otros objetos contenidos en la página mediante los URL de los objetos. Cada URL tiene dos componentes: el nombre de host del servidor que alberga al objeto y el nombre de la ruta al objeto. Por ejemplo, en el URL

`http://www.unaEscuela.edu/unDepartamento/imagen.gif`

`www.unaEscuela.edu` corresponde a un nombre de host y `/unDepartamento/imagen.gif` es el nombre de una ruta. Puesto que los **navegadores web** (como Internet Explorer y Firefox) implementan el lado del cliente de HTTP, en el contexto de la Web utilizaremos los términos *navegador* y *cliente* de forma indistinta. Los **servidores web**, que implementan el lado del servidor de HTTP, albergan los objetos web, siendo cada uno de ellos direccionable mediante un URL. Entre los servidores web más populares se incluyen Apache y Microsoft Internet Information Server.

HTTP define cómo los clientes web solicitan páginas web a los servidores y cómo estos servidores web transfieren esas páginas a los clientes. Más adelante veremos la interacción entre el cliente y el servidor en detalle, si bien la idea general se ilustra en la Figura 2.6. Cuando un usuario solicita una página web (por ejemplo, haciendo clic en un hipervínculo), el navegador envía al servidor mensajes de solicitud HTTP, pidiendo los objetos contenidos en la página. El servidor recibe las solicitudes y responde con mensajes de respuesta HTTP que contienen los objetos.

HTTP utiliza TCP como su protocolo de transporte subyacente (en lugar de ejecutarse por encima de UDP). El cliente HTTP primero inicia una conexión TCP con el servidor. Una vez que la conexión se ha establecido, los procesos de navegador y de servidor acceden a TCP a través de sus

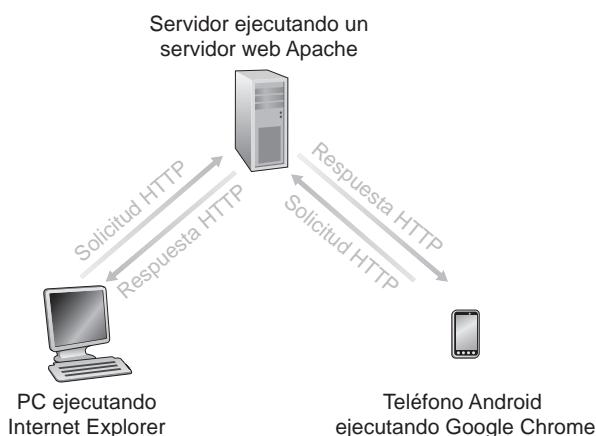


Figura 2.6 ♦ Comportamiento solicitud-respuesta de HTTP.

interfaces de socket. Como se ha descrito en la Sección 2.1, en el lado del cliente la interfaz de socket es la puerta entre el proceso cliente y la conexión TCP; en el lado del servidor, es la puerta entre el proceso servidor y la conexión TCP. El cliente envía mensajes de solicitud HTTP a su interfaz de socket y recibe mensajes de respuesta HTTP procedentes de su interfaz de socket. De forma similar, el servidor HTTP recibe mensajes de solicitud a través de su interfaz de socket y envía mensajes de respuesta a través de la misma. Una vez que el cliente envía un mensaje a su interfaz de socket, el mensaje deja de estar en las manos del cliente y pasa “a las manos” de TCP. Recuerde, de la Sección 2.1, que TCP proporciona un servicio de transferencia de datos fiable a HTTP. Esto implica que cada mensaje de solicitud HTTP enviado por un proceso cliente llegará intacto al servidor; del mismo modo, cada mensaje de respuesta HTTP enviado por el proceso servidor llegará intacto al cliente. Esta es una de las grandes ventajas de una arquitectura en capas: HTTP no tiene que preocuparse por las pérdidas de datos o por los detalles sobre cómo TCP recupera los datos perdidos o los reordena dentro de la red. Ése es el trabajo de TCP y de los protocolos de las capas inferiores de la pila de protocolos.

Es importante observar que el servidor envía los archivos solicitados a los clientes sin almacenar ninguna información acerca del estado del cliente. Si un determinado cliente pide el mismo objeto dos veces en un espacio de tiempo de unos pocos segundos, el servidor no responde diciendo que acaba de servir dicho objeto al cliente; en su lugar, el servidor reenvía el objeto, ya que ha olvidado por completo que ya lo había hecho anteriormente. Dado que un servidor HTTP no mantiene ninguna información acerca de los clientes, se dice que HTTP es un protocolo sin memoria del estado. Debemos destacar también que la Web utiliza la arquitectura de aplicación cliente-servidor, descrita en la Sección 2.1. Un servidor web siempre está activo, con una dirección IP fija, y da servicio a solicitudes procedentes de, potencialmente, millones de navegadores distintos.

2.2.2 Conexiones persistentes y no persistentes

En muchas aplicaciones de Internet, el cliente y el servidor están en comunicación durante un periodo de tiempo amplio, haciendo el cliente una serie de solicitudes y respondiendo el servidor a dichas solicitudes. Dependiendo de la aplicación y de cómo se esté empleando, las solicitudes pueden hacerse una tras otra, periódicamente a intervalos regulares o de forma intermitente. Cuando esta interacción cliente-servidor tiene lugar sobre TCP, el desarrollador de la aplicación tiene que tomar una decisión importante: ¿debería cada par solicitud/respuesta enviarse a través de una conexión TCP separada o deberían enviarse todas las solicitudes y sus correspondientes respuestas a través de la misma conexión TCP? Si se utiliza el primer método, se dice que la aplicación emplea conexiones no persistentes; si se emplea la segunda opción, entonces se habla de conexiones persistentes. Con el fin de profundizar en esta cuestión de diseño, vamos a examinar las ventajas y desventajas de las conexiones persistentes en el contexto de una aplicación específica, en concreto HTTP, que puede utilizar ambos tipos de conexión. Aunque HTTP emplea conexiones persistentes de manera predeterminada, los clientes y servidores HTTP se pueden configurar para emplear en su lugar conexiones no persistentes.

HTTP con conexiones no persistentes

Sigamos los pasos que permiten transferir una página web desde un servidor a un cliente en el caso de conexiones no persistentes. Supongamos que la página consta de un archivo base HTML y de 10 imágenes JPEG, residiendo los 11 objetos en el mismo servidor. Supongamos también que el URL del archivo base HTML es:

`http://www.unaEscuela.edu/unDepartamento/home.index`

Lo que ocurre es lo siguiente:

1. El proceso cliente HTTP inicia una conexión TCP con el servidor `www.unaEscuela.edu` en el puerto número 80, que es el número de puerto predeterminado para HTTP. Asociados con la conexión TCP, habrá un socket en el cliente y un socket en el servidor.
2. El cliente HTTP envía un mensaje de solicitud HTTP al servidor a través de su socket. El mensaje de solicitud incluye el nombre de la ruta `/unDepartamento/home.index`. (Más adelante veremos con más detalle los mensajes HTTP.)
3. El proceso servidor HTTP recibe el mensaje de solicitud a través de su socket, recupera el objeto `/unDepartamento/home.index` de su medio de almacenamiento (RAM o disco), encapsula el objeto en un mensaje de respuesta HTTP y lo envía al cliente a través de su socket.
4. El proceso servidor HTTP indica a TCP que cierre la conexión TCP. (Pero TCP realmente no termina la conexión hasta que está seguro de que el cliente ha recibido el mensaje de respuesta en perfecto estado.)
5. El cliente HTTP recibe el mensaje de respuesta. La conexión TCP termina. El mensaje indica que el objeto encapsulado es un archivo HTML. El cliente extrae el archivo del mensaje de respuesta, examina el archivo HTML y encuentra las referencias a los 10 objetos JPEG.
6. Los cuatro primeros pasos se repiten entonces para cada uno de los objetos JPEG referenciados.

Cuando el navegador recibe la página web, la muestra al usuario. Dos navegadores distintos pueden interpretar (es decir, mostrar al usuario) una página web de formas ligeramente distintas. HTTP no tiene nada que ver con cómo un cliente interpreta una página web. Las especificaciones HTTP ([RFC 1945] y [RFC 2616]) únicamente definen el protocolo de comunicación entre el programa cliente HTTP y el programa servidor HTTP.

Los pasos anteriores ilustran el uso de las conexiones no persistentes, donde cada conexión TCP se cierra después de que el servidor envíe el objeto: la conexión no se mantiene (no persiste) para los restantes objetos. Observe que cada conexión TCP transporta exactamente un mensaje de solicitud y un mensaje de respuesta. Por tanto, en este ejemplo, cuando un usuario solicita la página web, se generan 11 conexiones TCP.

En los pasos descritos anteriormente, hemos sido intencionadamente vagos en lo que respecta a si el cliente obtiene las diez imágenes JPEG a través de diez conexiones TCP en serie o si algunas de dichas imágenes se obtienen a través de conexiones TCP en paralelo. De hecho, los usuarios pueden configurar los navegadores modernos para controlar el grado de paralelismo. En sus modos predeterminados, la mayoría de los navegadores abren entre 5 y 10 conexiones TCP en paralelo y cada una de estas conexiones gestiona una transacción solicitud-respuesta. Si el usuario lo prefiere, el número máximo de conexiones en paralelo puede establecerse en uno, en cuyo caso se establecerán diez conexiones en serie. Como veremos en el siguiente capítulo, el uso de conexiones en paralelo reduce el tiempo de respuesta.

Antes de continuar, vamos a realizar un cálculo aproximado para estimar la cantidad de tiempo que transcurre desde que un cliente solicita el archivo base HTML hasta que recibe dicho archivo completo. Para ello, definimos el **tiempo de ida y vuelta (RTT, Round-Trip Time)**, que es el tiempo que tarda un paquete pequeño en viajar desde el cliente al servidor y volver de nuevo al cliente. El RTT incluye los retardos de propagación de los paquetes, los retardos de cola en los routers y switches intermedios y los retardos de procesamiento de los paquetes (estos retardos se explican en la Sección 1.4). Consideremos ahora lo que ocurre cuando un usuario hace clic en un hipervínculo. Como se muestra en la Figura 2.7, esto hace que el navegador inicie una conexión TCP entre el navegador y el servidor web, lo que implica un proceso de “acuerdo en tres fases” (el cliente envía un pequeño segmento TCP al servidor, el servidor reconoce la recepción y responde con otro pequeño segmento TCP y, por último, el cliente devuelve un mensaje de reconocimiento al servidor). Las dos primeras partes de este proceso de acuerdo en tres fases tardan un periodo de tiempo igual a RTT. Después de completarse las dos primeras fases de la negociación, el cliente envía a la conexión TCP el mensaje de solicitud HTTP combinado con la tercera parte de la negociación (el mensaje de reconocimiento). Una vez que el mensaje de solicitud llega al servidor, este envía el

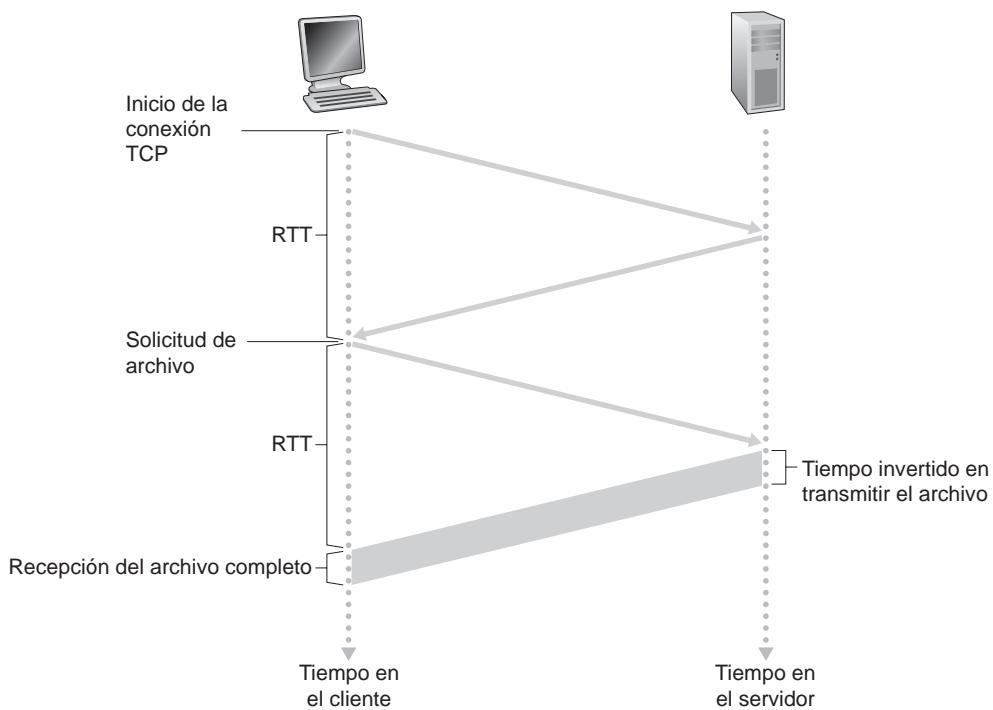


Figura 2.7 ♦ Cálculo aproximado del tiempo necesario para solicitar y recibir un archivo HTML.

archivo HTML a través de la conexión TCP. Este mensaje de solicitud/respuesta HTTP consume otro periodo de tiempo RTT. Luego el tiempo de respuesta total es aproximadamente igual a dos RTT más el tiempo de transmisión del archivo HTML por parte del servidor.

HTTP con conexiones persistentes

Las conexiones no persistentes presentan algunos inconvenientes. En primer lugar, tiene que establecerse y gestionarse una conexión completamente nueva *para cada objeto solicitado*. Para cada una de estas conexiones, deben asignarse buffers TCP y tienen que gestionarse variables TCP tanto en el cliente como en el servidor. Esto puede sobrecargar de forma significativa al servidor web, ya que puede estar sirviendo solicitudes de cientos de clientes distintos simultáneamente. En segundo lugar, como ya hemos explicado, *cada objeto sufre un retardo de entrega de dos RTT: un RTT para establecer la conexión TCP y otro RTT para solicitar y recibir un objeto*.

Con las conexiones persistentes de HTTP 1.1, el servidor deja la conexión TCP abierta después de enviar una respuesta. Las subsiguientes solicitudes y respuestas que tienen lugar entre el mismo cliente y el servidor pueden enviarse a través de la misma conexión. En concreto, *una página web completa* (en el ejemplo anterior, el archivo base HTML y las 10 imágenes) se puede enviar a través de una misma conexión TCP persistente. Además, *varias páginas web que residan en el mismo servidor* pueden enviarse desde el servidor a un mismo cliente a través de una única conexión TCP persistente. *Estas solicitudes de objetos pueden realizarse una tras otra sin esperar a obtener las respuestas a las solicitudes pendientes (pipelining, procesamiento en cadena)*. Normalmente, el servidor HTTP cierra una conexión cuando no se ha utilizado durante cierto tiempo (un intervalo de fin de temporización configurable). *Cuando el servidor recibe las solicitudes una tras otra, envía los objetos uno tras otro. El modo predeterminado de HTTP utiliza conexiones persistentes con procesamiento en cadena*. Más recientemente, HTTP/2 [RFC 7540] amplía la funcionalidad

de HTTP 1.1, permitiendo entrelazar múltiples solicitudes y respuestas en la *misma* conexión, y proporcionando también un mecanismo para priorizar los mensajes de solicitud y respuesta HTTP dentro de dicha conexión. En los problemas de repaso de los Capítulos 2 y 3 compararemos cuantitativamente el rendimiento de las conexiones persistentes y no persistentes. Le animamos también a que consulte [Heidemann 1997; Nielsen 1997; RFC 7540].

2.2.3 Formato de los mensajes HTTP

Las especificaciones HTTP [RFC 1945; RFC 2616]; RFC 7540) incluyen las definiciones de los formatos de los mensajes HTTP. A continuación vamos a estudiar los dos tipos de mensajes HTTP existentes: mensajes de solicitud y mensajes de respuesta.

Mensaje de solicitud HTTP

A continuación se muestra un mensaje de solicitud HTTP típico:

```
GET /unadireccion/pagina.html HTTP/1.1
Host: www.unaEscuela.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Podemos aprender muchas cosas si miramos en detalle este sencillo mensaje de solicitud. En primer lugar, podemos comprobar que el mensaje está escrito en texto ASCII normal, por lo que cualquier persona con conocimientos informáticos puede leerlo. En segundo lugar, vemos que el mensaje consta de cinco líneas, cada una de ellas seguida por un retorno de carro y un salto de línea. La última línea va seguida de un retorno de carro y un salto de línea adicionales. Aunque este mensaje en concreto está formado por cinco líneas, un mensaje de solicitud puede constar de muchas más líneas o tener tan solo una. La primera línea de un mensaje de solicitud HTTP se denomina **línea de solicitud** y las siguientes son las **líneas de cabecera**. La línea de solicitud consta de tres campos: el campo de método, el campo URL y el campo de la versión HTTP. El campo que especifica el **método** puede tomar diferentes valores, entre los que se incluyen **GET, POST, HEAD, PUT y DELETE**. La inmensa mayoría de los mensajes de solicitud HTTP utilizan el método GET. Este método se emplea cuando el navegador solicita un objeto, identificándose dicho objeto en el campo URL. En este ejemplo, el navegador está solicitando el objeto `/unadireccion/pagina.html`. El campo correspondiente a la versión se explica por sí mismo; en este ejemplo, el navegador utiliza la versión HTTP/1.1.

Analicemos ahora las líneas de cabecera de este ejemplo. La línea de cabecera `Host: www.unaescuela.edu` especifica el **host** en el que reside el **objeto**. Podría pensarse que esta línea de cabecera es innecesaria, puesto que ya existe una conexión TCP activa con el host. Pero, como veremos en la Sección 2.2.5, las cachés proxy web necesitan la información proporcionada por la línea de cabecera del host. Al incluir la línea de cabecera `Connection: close`, el navegador está diciendo al servidor que no desea molestarle en trabajar con conexiones persistentes, sino que desea que el servidor cierre la conexión después de enviar el objeto solicitado. La línea de cabecera `User-agent`: especifica el agente de usuario, es decir, el tipo de navegador que está haciendo la solicitud al servidor. En este caso, el agente de usuario es Mozilla/5.0, un navegador Firefox. Esta línea de cabecera resulta útil porque el servidor puede enviar versiones diferentes del mismo objeto a los distintos tipos de agentes de usuario (todas las versiones tienen la misma dirección URL). Por último, la línea de cabecera `Accept-language`: indica que el usuario prefiere recibir una versión en francés del objeto, si tal objeto existe en el servidor; en caso contrario, el servidor debe enviar la versión predeterminada. La línea de cabecera `Accept-language`: sólo es una de las muchas cabeceras de negociación del contenido disponibles en HTTP.

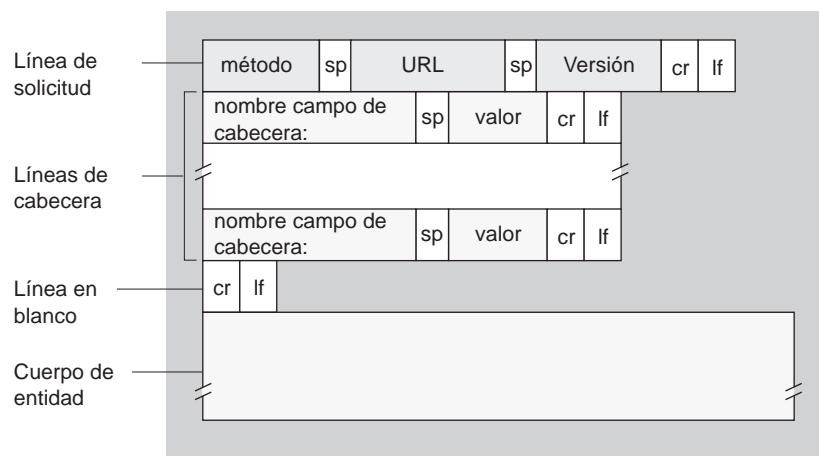


Figura 2.8 ♦ Formato general de un mensaje de solicitud HTTP.

Una vez visto un ejemplo, vamos a estudiar el formato general de un mensaje de solicitud, ilustrado en la Figura 2.8. Podemos comprobar que el formato general es muy similar al usado en el ejemplo anterior. Sin embargo, fíjese en que después de las líneas de cabecera (y el retorno de carro y el salto de línea adicionales) se incluye un “cuerpo de entidad”. Este campo queda vacío cuando se utiliza el método GET, pero no cuando se usa el método POST. A menudo, un cliente HTTP utiliza el método POST cuando el usuario completa un formulario; por ejemplo, cuando especifica términos para realizar una búsqueda utilizando un motor de búsqueda. Con un mensaje POST, el usuario solicita también una página web al servidor, pero el contenido concreto de la misma dependerá de lo que el usuario haya escrito en los campos del formulario. Si el valor del campo de método es POST, entonces el cuerpo de la entidad contendrá lo que el usuario haya introducido en los campos del formulario.

No podemos dejar de mencionar que una solicitud generada con un formulario no necesariamente utiliza el método POST. En su lugar, a menudo los formularios HTML emplean el método GET e incluyen los datos de entrada (especificados en los campos del formulario) en el URL solicitado. Por ejemplo, si un formulario emplea el método GET y tiene dos campos, y las entradas a esos dos campos son monos y bananas, entonces el URL tendrá la estructura `www.unsitio.com/busquedadeanimales?monos&bananas`. Probablemente habrá visto direcciones URL ampliadas de este tipo, al navegar por la Web.

El método HEAD es similar al método GET. Cuando un servidor recibe una solicitud con el método HEAD, responde con un mensaje HTTP, pero excluye el objeto solicitado. Los desarrolladores de aplicaciones a menudo utilizan el método HEAD para labores de depuración. El método PUT suele utilizarse junto con herramientas de publicación web. Esto permite a un usuario cargar un objeto en una ruta específica (directorio) en un servidor web determinado. Las aplicaciones que necesitan cargar objetos en servidores web también emplean el método PUT. El método DELETE permite a un usuario o a una aplicación borrar un objeto de un servidor web.

Mensajes de respuesta HTTP

A continuación se muestra un mensaje de respuesta HTTP típico. Este mensaje podría ser la respuesta al ejemplo de mensaje de solicitud que acabamos de ver.

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
```

```
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(datos datos datos datos datos ...)
```

Examinemos detenidamente este mensaje de respuesta. Tiene tres secciones: una **línea de estado** inicial, seis **líneas de cabecera** y después el **cuerpo de entidad**. El cuerpo de entidad es la parte más importante del mensaje, ya que contiene el objeto solicitado en sí (representado por datos datos datos datos datos ...). La línea de estado contiene tres campos: el que especifica la versión del protocolo, un código de estado y el correspondiente mensaje explicativo del estado. En este ejemplo, la línea de estado indica que el servidor está utilizando HTTP/1.1 y que todo es correcto (OK); es decir, que el servidor ha encontrado y está enviando el objeto solicitado.

Veamos ahora las líneas de cabecera. El servidor utiliza la línea de cabecera **Connection: close** para indicar al cliente que va a cerrar la conexión TCP después de enviar el mensaje. La línea de cabecera **Date:** indica la hora y la fecha en la que se creó la respuesta HTTP y fue enviada por el servidor. Observe que esta línea no especifica la hora en que el objeto fue creado o modificado por última vez; es la hora en la que el servidor recupera el objeto de su sistema de archivos, inserta el objeto en el mensaje de respuesta y lo envía. La línea de cabecera **Server:** indica que el mensaje fue generado por un servidor web Apache; es análoga a la línea de cabecera **User-agent:** del mensaje de solicitud HTTP. La línea de cabecera **Last-Modified:** especifica la hora y la fecha en que el objeto fue creado o modificado por última vez. Esta línea **Last-Modified:**, que enseñaremos en detalle, resulta fundamental para el almacenamiento en caché del objeto, tanto en el cliente local como en los servidores de almacenamiento en caché de la red (también conocidos como servidores proxy). La línea de cabecera **Content-Length:** especifica el número de bytes del objeto que está siendo enviado. La línea **Content-Type:** indica que el objeto incluido en el cuerpo de entidad es texto HTML. (El tipo de objeto está indicado oficialmente por la línea de cabecera **Content-Type:** y no por la extensión del archivo.)

Una vez visto un ejemplo, vamos a pasar a examinar el formato general de un mensaje de respuesta, ilustrado en la Figura 2.9. Este formato general de mensaje de respuesta se corresponde con el del ejemplo anterior. Vamos a comentar algunas cosas más acerca de los códigos de estado y sus descripciones. El código de estado y su frase asociada indican el resultado de la solicitud. Algunos códigos de estado comunes y sus frases asociadas son:

- 200 OK: La solicitud se ha ejecutado con éxito y se ha devuelto la información en el mensaje de respuesta.
- 301 Moved Permanently: El objeto solicitado ha sido movido de forma permanente; el nuevo URL se especifica en la línea de cabecera **Location:** del mensaje de respuesta. El software cliente recuperará automáticamente el nuevo URL.
- 400 Bad Request: Se trata de un código de error genérico que indica que la solicitud no ha sido comprendida por el servidor.
- 404 Not Found: El documento solicitado no existe en este servidor.
- 505 HTTP Version Not Supported: La versión de protocolo HTTP solicitada no es soportada por el servidor.



Nota de video
Uso de Wireshark para investigar el protocolo HTTP.

¿Le gustaría ver un mensaje de respuesta HTTP real? ¡Esto es muy recomendable y además es muy fácil de hacer! En primer lugar, establezca una conexión Telnet con su servidor web favorito. A continuación, escriba un mensaje de solicitud de una línea para obtener algún objeto que esté almacenado en ese servidor. Por ejemplo, si tiene acceso a la línea de comandos (*prompt*), escriba:

```
telnet gaia.cs.umass.edu 80
GET /kurose_ross/interactive/index.php HTTP/1.1
Host: gaia.cs.umass.edu
```

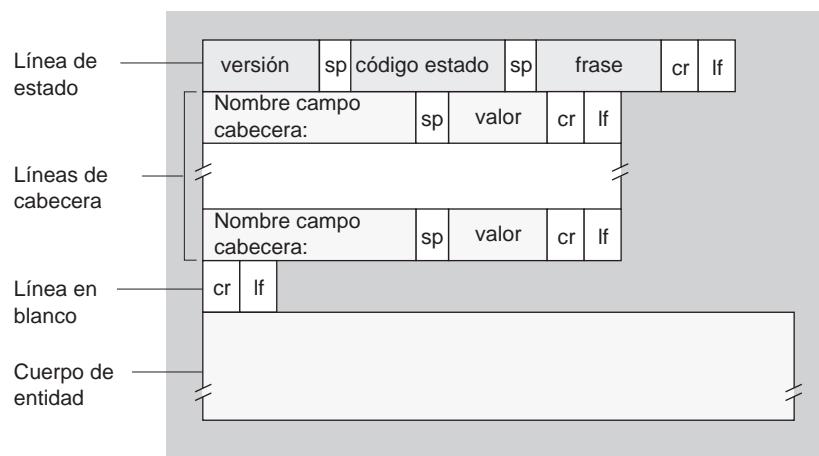


Figura 2.9 ♦ Formato general de un mensaje de respuesta HTTP.

(Pulse dos veces la tecla retorno de carro después de escribir la última línea.) De este modo se abre una conexión TCP en el puerto 80 del host `gaia.cs.umas.edu` y luego se envía el mensaje de solicitud HTTP. Debería ver un mensaje de respuesta que incluya el archivo base HTML de los problemas interactivos de repaso de este libro. Para ver simplemente las líneas del mensaje HTTP y no recibir el objeto, sustituya `GET` por `HEAD`.

En esta sección hemos visto una serie de líneas de cabecera que pueden utilizarse en los mensajes HTTP de solicitud y respuesta. La especificación de HTTP define un gran número de otras líneas de cabecera que pueden ser insertadas por los navegadores, servidores web y servidores de almacenamiento en caché de la red. Hemos cubierto únicamente una pequeña parte de la totalidad de líneas de cabecera disponibles. A continuación veremos unas pocas más y en la Sección 2.2.5 algunas otras, al hablar del almacenamiento en cachés web de la red. Puede leer una exposición enormemente clara y comprensible acerca del protocolo HTTP, incluyendo sus cabeceras y códigos de estado en [Krishnamurty 2001].

¿Cómo decide un navegador qué líneas de cabecera incluir en un mensaje de solicitud? ¿Cómo decide un servidor web qué líneas de cabecera incluir en un mensaje de respuesta? Un navegador generará líneas de cabecera en función del tipo y la versión del navegador (por ejemplo, un navegador HTTP/1.0 no generará ninguna línea de cabecera correspondiente a la versión 1.1), de la configuración del navegador que tenga el usuario (por ejemplo, el idioma preferido) y de si el navegador tiene actualmente en caché una versión del objeto, posiblemente desactualizada. Los servidores web se comportan de forma similar: existen productos, versiones y configuraciones diferentes, que influyen en las líneas de cabecera que se incluirán en los mensajes de respuesta.

2.2.4 Interacción usuario-servidor: cookies

Hemos mencionado anteriormente que un servidor HTTP no tiene memoria del estado de la conexión. Esto simplifica el diseño del servidor y ha permitido a los ingenieros desarrollar servidores web de alto rendimiento que pueden gestionar miles de conexiones TCP simultáneas. Sin embargo, para un sitio web, a menudo es deseable poder identificar a los usuarios, bien porque el servidor desea restringir el acceso a los usuarios o porque desea servir el contenido en función de la identidad del usuario. Para estos propósitos, HTTP utiliza cookies. Las cookies, definidas en [RFC 6265], permiten a los sitios seguir la pista a los usuarios. Actualmente, la mayoría de los sitios web comerciales más importantes utilizan cookies.

Como se muestra en la Figura 2.10, la tecnología de las cookies utiliza cuatro componentes: (1) una línea de cabecera de la cookie en el mensaje de respuesta HTTP; (2) una línea de cabecera de la cookie en el mensaje de solicitud HTTP; (3) el archivo de cookies almacenado en el sistema

terminal del usuario y gestionado por el navegador del usuario; y (4) una base de datos back-end en el sitio web. Basándonos en la Figura 2.10, vamos a ver mediante un ejemplo cómo funcionan las cookies. Suponga que Susana, que accede siempre a la Web utilizando Internet Explorer en el PC de su casa, entra en Amazon.com por primera vez. Supongamos que anteriormente ella había visitado el sitio de eBay. Cuando la solicitud llega al servidor web de Amazon, el servidor genera un número de identificación exclusivo y crea una entrada en su base de datos back-end que está indexada por el número de identificación. El servidor web de Amazon responde entonces al navegador de Susana, incluyendo en la respuesta HTTP una línea de cabecera `Set-cookie:`, que contiene el número de identificación. Por ejemplo, la línea de cabecera podría ser:

`Set-cookie: 1678`

Cuando el navegador de Susana recibe el mensaje de respuesta HTTP, ve la cabecera `Set-cookie:`. Entonces el navegador añade una línea a su archivo especial de cookies. Esta línea incluye el nombre de host del servidor y el número de identificación contenido en la cabecera `Set-cookie:`. Observe que el archivo de cookies ya tiene una entrada para eBay, dado que Susana había visitado dicho sitio en el pasado. A medida que Susana continúa navegando por el sitio de Amazon, cada vez que solicita

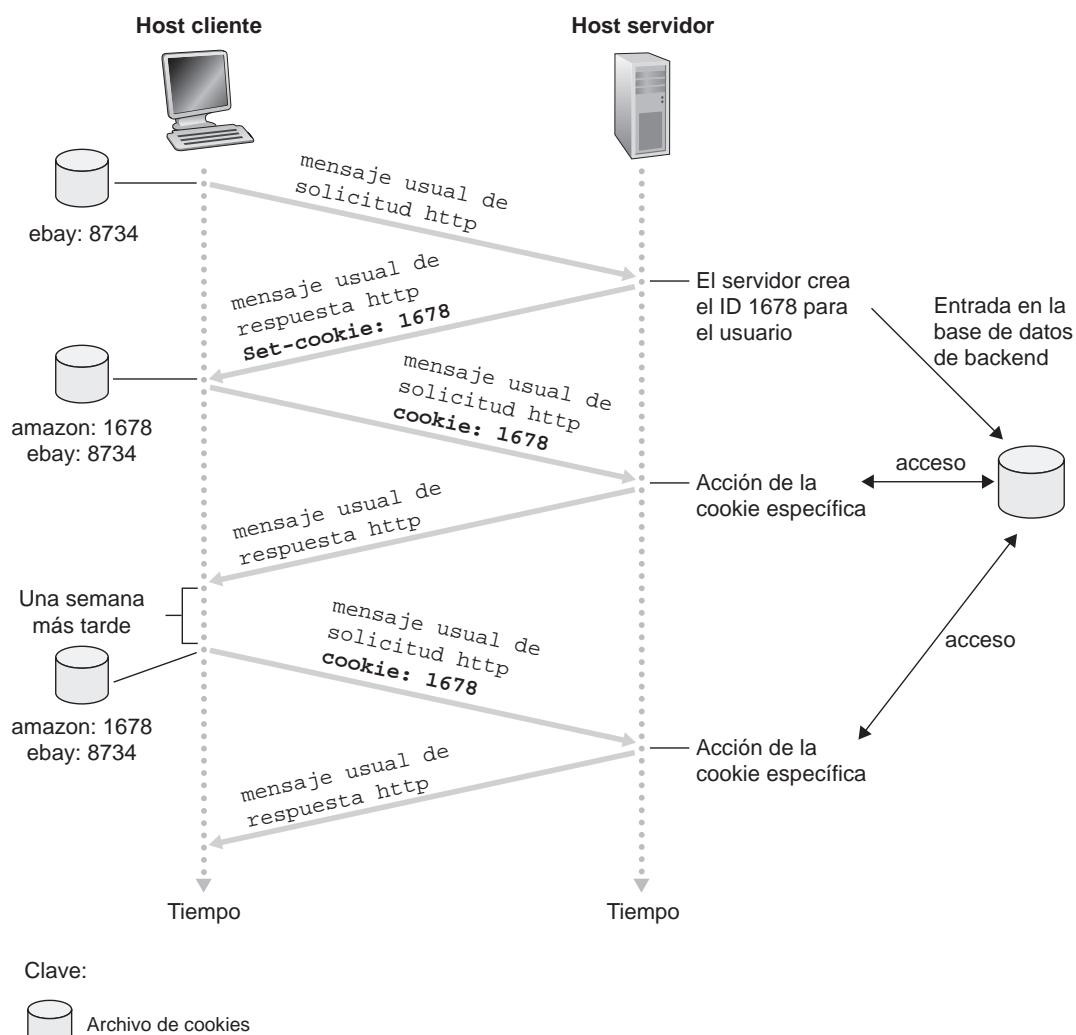


Figura 2.10 ♦ Mantenimiento del estado del usuario mediante cookies.

una página web su navegador consulta su archivo de cookies, extrae su número de identificación para ese sitio y añade a la solicitud HTTP una línea de cabecera de cookie que incluye el número de identificación. Específicamente, cada una de sus solicitudes HTTP al servidor de Amazon incluye la línea de cabecera:

Cookie: 1678

De esta forma, el servidor de Amazon puede seguir la actividad de Susana en ese sitio web. Aunque el sitio web de Amazon no necesariamente conoce el nombre de Susana, ¡sabe exactamente qué páginas ha visitado el usuario número 1678, en qué orden y cuántas veces!. Amazon utiliza cookies para proporcionar su servicio de carro de la compra: Amazon puede mantener una lista de todas las compras previstas por Susana, con el fin de que pueda pagarlas todas juntas al final de la sesión.

Si Susana vuelve al sitio de Amazon, por ejemplo una semana más tarde, su navegador continuará incluyendo la línea de cabecera Cookie: 1678 en los mensajes de solicitud. Amazon también recomienda productos a Susana basándose en las páginas web que ha visitado anteriormente dentro del sitio. Si Susana se registra en Amazon, proporcionando su nombre completo, dirección de correo electrónico, dirección postal y la información de su tarjeta crédito, entonces Amazon podrá incluir esta información en su base de datos, asociando el nombre de Susana con su número de identificación (¡y con todas las páginas que ha visitado dentro del sitio en el pasado!). Así es como Amazon y otros sitios de comercio electrónico proporcionan el servicio de “compra con un clic”: cuando Susana desee comprar un producto en una visita posterior, no tendrá que volver a escribir su nombre, su número de la tarjeta de crédito ni su dirección.

Como puede ver, las cookies pueden utilizarse para identificar a un usuario. La primera vez que un usuario visita un sitio, el usuario puede proporcionar una identificación suya (posiblemente su nombre). En las sesiones posteriores, el navegador pasa al servidor una cabecera de cookie, identificando al usuario ante el servidor. Las cookies pueden por tanto utilizarse para crear una capa de sesión por encima del protocolo HTTP sin memoria del estado de la conexión. Por ejemplo, cuando un usuario inicia una sesión en una aplicación de correo electrónico basada en la Web (como por ejemplo Hotmail), el navegador envía al servidor la información de la cookie, permitiendo al servidor identificar al usuario a lo largo de la sesión que el usuario mantenga con la aplicación.

Aunque las cookies a menudo simplifican a los usuarios la realización de compras por Internet, son controvertidas, porque también pueden considerarse como una invasión de la intimidad del usuario. Como acabamos de ver, empleando una combinación de cookies y de la información sobre cuentas suministrada por el usuario, un sitio web puede obtener mucha información de un usuario y, potencialmente, puede vender esa información a otras empresas. Cookie Central [Cookie Central 2016] proporciona mucha información acerca de la controversia de las cookies.

2.2.5 Almacenamiento en caché web

Una caché web, también denominada servidor proxy, es una entidad de red que satisface solicitudes HTTP en nombre de un servidor web de origen. La caché web dispone de su propio almacenamiento en disco y mantiene en él copias de los objetos solicitados recientemente. Como se muestra en la Figura 2.11, el navegador de un usuario se puede configurar de modo que todas sus solicitudes HTTP se dirijan en primer lugar a la caché web. Por ejemplo, suponga que un navegador está solicitando el objeto `http://www.unaEscuela.edu/campus.gif`. Veamos qué es lo que ocurre:

1. El navegador establece una conexión TCP con la caché web y envía una solicitud HTTP para el objeto a la caché web.
2. La caché web comprueba si tiene una copia del objeto almacenada localmente. Si la tiene, la caché web devuelve el objeto dentro de un mensaje de respuesta HTTP al navegador del cliente.
3. Si la caché web no tiene el objeto, abre una conexión TCP con el servidor de origen, es decir, con `www.unaEscuela.edu`. La caché web envía entonces una solicitud HTTP para obtener

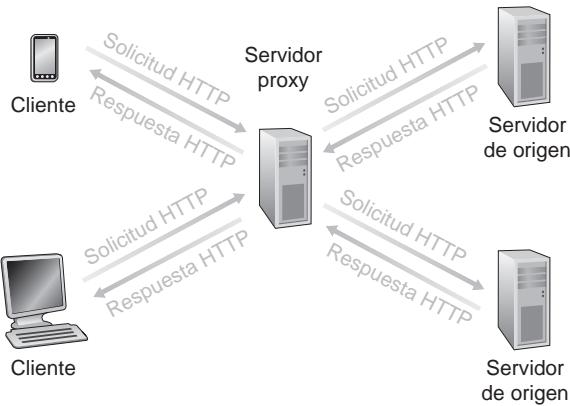


Figura 2.11 ♦ Clientes que solicitan objetos a través de una caché web.

el objeto a través de la conexión TCP caché-servidor. Después de recibir esta solicitud, el servidor de origen envía el objeto dentro de un mensaje de respuesta HTTP a la caché web.

4. Cuando la caché web recibe el objeto, almacena una copia en su dispositivo de almacenamiento local y envía una copia, dentro de un mensaje de respuesta HTTP, al navegador del cliente (a través de la conexión TCP existente entre el navegador del cliente y la caché web).

Observe que una caché es a la vez un servidor y un cliente. Cuando recibe solicitudes de y envía respuestas a un navegador, se comporta como un servidor. Cuando envía solicitudes a y recibe respuestas de un servidor de origen, entonces actúa como un cliente.

Habitualmente es un ISP quien adquiere e instala una caché web. Por ejemplo, una universidad podría instalar una caché en su red del campus y configurar todos los navegadores del campus apuntando a la caché. O un ISP residencial de gran tamaño (como AOL) podría instalar una o más cachés en su red y preconfigurar los navegadores que suministre para que apunten a las cachés instaladas.

El almacenamiento en caché web se ha implantado en Internet por dos razones. La primera razón es que una caché web puede reducir sustancialmente el tiempo de respuesta a la solicitud de un cliente, especialmente si el ancho de banda cuello de botella entre el cliente y el servidor de origen es mucho menor que el ancho de banda cuello de botella entre el cliente y la caché. Si existe una conexión de alta velocidad entre el cliente y la caché, lo que ocurre a menudo, y si la caché tiene el objeto solicitado, entonces ésta podrá suministrar el objeto rápidamente al cliente. La segunda razón es que, como ilustraremos enseguida con un ejemplo, las cachés web pueden reducir sustancialmente el tráfico en el enlace de acceso a Internet de una institución. Reduciendo el tráfico, la institución (por ejemplo, una empresa o una universidad) no tiene que mejorar el ancho de banda tan rápidamente, lo que implica una reducción de costes. Además, las cachés web pueden reducir mucho el tráfico web global en Internet, mejorando en consecuencia el rendimiento de todas las aplicaciones.

Para adquirir un conocimiento profundo de las ventajas de las cachés, vamos a ver un ejemplo en el contexto ilustrado en la Figura 2.12. Esta figura muestra dos redes: la red institucional y el resto de la red pública Internet. La red institucional es una LAN de alta velocidad. Un router de la red institucional y un router de Internet están conectados mediante un enlace a 15 Mbps. Los servidores de origen están conectados a Internet pero se encuentran distribuidos por todo el mundo. Suponga que el tamaño medio del objeto es de 1 Mbits y que la tasa media de solicitudes de los navegadores de la institución a los servidores de origen es de 15 solicitudes por segundo. Suponga que los mensajes de solicitud HTTP son extremadamente pequeños y que por tanto no crean tráfico en las redes ni en el enlace de acceso (desde el router institucional al router de Internet). Suponga también que el tiempo que se tarda en reenviar una solicitud HTTP (contenida en un datagrama IP) desde el router del lado de Internet del enlace de acceso de la Figura 2.12 hasta que

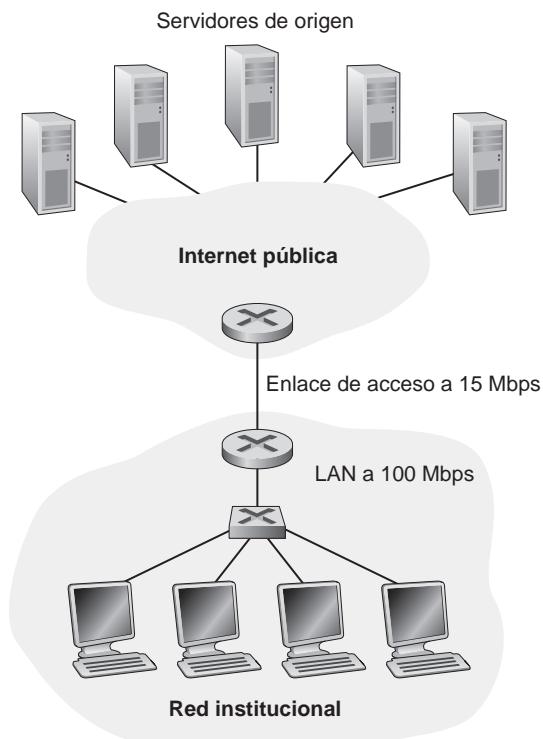


Figura 2.12 ♦ Cuello de botella entre una red institucional e Internet.

se recibe la respuesta (normalmente contenida en muchos datagramas IP) es igual a dos segundos. Informalmente, vamos a denominar a este último retardo “retardo de Internet”.

El tiempo total de respuesta, es decir, el tiempo desde que el navegador solicita un objeto hasta que lo recibe, es la suma del retardo de la LAN, el retardo de acceso (es decir, el retardo entre los dos routers) y el retardo de Internet. Hagamos ahora un burdo cálculo para estimar este retardo. La intensidad de tráfico en la LAN es (véase la Sección 1.4.2):

$$(15 \text{ solicitudes/s}) \cdot (1 \text{ Mbits/solicitud})/(100 \text{ Mbps}) = 0,15$$

mientras que la intensidad de tráfico en el enlace de acceso (desde el router de Internet hasta el router de la institución) es:

$$(15 \text{ solicitudes/s}) \cdot (1 \text{ Mbits/solicitud})/(15 \text{ Mbps}) = 1$$

Una intensidad de tráfico de 0,15 en una LAN normalmente da resultados, como máximo, de decenas de milisegundos de retardo; por tanto, podemos despreciar el retardo de la LAN. Sin embargo, como hemos visto en la Sección 1.4.2, cuando la intensidad de tráfico se aproxima a 1 (como es el caso del enlace de acceso de la Figura 2.12), el retardo en el enlace comienza a aumentar y crece sin límite. Por tanto, el tiempo medio de respuesta para satisfacer las solicitudes es del orden de minutos, si no mayor, lo que es inaceptable para los usuarios de la institución. Evidentemente, es necesario hacer algo.

Una posible solución es incrementar la velocidad de acceso de 15 Mbps a, por ejemplo, 100 Mbps. Esto disminuirá la intensidad de tráfico en el enlace de acceso a 0,15, lo que se traduce en retardos despreciables entre los dos routers. En este caso, el tiempo total de respuesta será de unos dos segundos, es decir, el retardo de Internet. Pero esta solución también implica que la institución debe actualizar su enlace de acceso de 15 Mbps a 100 Mbps, lo que resulta ser una solución cara.

Consideremos ahora la solución alternativa de no actualizar el enlace de acceso sino, en su lugar, instalar una caché web en la red institucional. Esta solución se ilustra en la Figura 2.13. Las tasas de acierto, es decir, la fracción de solicitudes que son satisfechas por una caché, suelen estar comprendidas, en la práctica, entre 0,2 y 0,7. Con propósitos ilustrativos, supongamos que la caché proporciona una tasa de acierto de 0,4 para esta institución. Dado que los clientes y la caché están conectados a la misma LAN de alta velocidad, el 40 por ciento de las solicitudes serán satisfechas casi de forma inmediata, es decir, en 10 milisegundos o menos por la caché. No obstante, el restante 60 por ciento de las solicitudes todavía tendrán que ser satisfechas por los servidores de origen. Pero sólo con el 60 por ciento de los objetos solicitados atravesando el enlace de acceso, la intensidad de tráfico en el mismo se reduce de 1,0 a 0,6. Típicamente, una intensidad de tráfico menor que 0,8 se corresponde con un retardo pequeño, digamos de unas decenas de milisegundos, en un enlace de 15 Mbps. Este retardo es despreciable comparado con los dos segundos del retardo de Internet. Teniendo en cuenta todo esto, el retardo medio será entonces:

$$0,4 \cdot (0,01 \text{ segundos}) + 0,6 \cdot (2,01 \text{ segundos})$$

lo que es algo más de 1,2 segundos. Por tanto, esta segunda solución proporciona incluso un tiempo de respuesta menor que la primera y no requiere que la institución actualice su enlace con Internet. Por supuesto, la institución tiene que comprar e instalar una caché web, pero este coste es bajo, ya que muchas cachés utilizan software de dominio público que se ejecuta en computadoras PC baratas.

Gracias al uso de las **redes de distribución de contenido (CDN, Content Distribution Networks)**, las cachés web están desempeñando un papel cada vez más importante en Internet. Una empresa CDN instala muchas cachés distribuidas geográficamente a través de Internet, localizando la mayor parte del tráfico. Existen redes CDN compartidas (por ejemplo, Akamai y Limelight) y redes CDN dedicadas (como Google y Netflix). Veremos la redes CDN en detalle en la Sección 2.6.

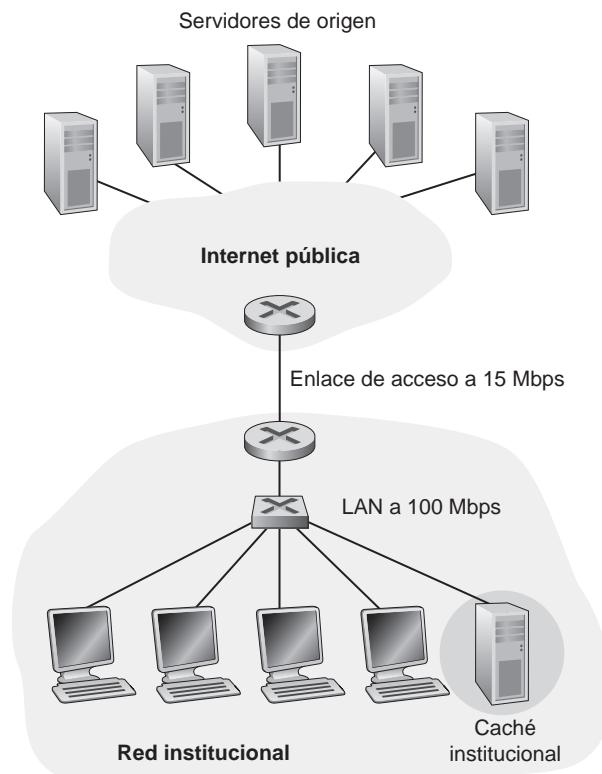


Figura 2.13 ♦ Adición de una caché a una red institucional.

GET condicional

Aunque el almacenamiento en caché puede reducir los tiempos de respuesta percibidos por el usuario, introduce un nuevo problema: la copia de un objeto que reside en la caché puede estar desactualizada. En otras palabras, el objeto almacenado en el servidor web puede haber sido modificado desde que la copia fue almacenada en la caché del cliente. Afortunadamente, HTTP dispone de un mecanismo que permite a la caché verificar que sus objetos están actualizados. Este mecanismo se denomina **GET condicional**. Un mensaje de solicitud HTTP se denomina también mensaje GET condicional si (1) el mensaje de solicitud utiliza el método GET y (2) el mensaje de solicitud incluye una línea de cabecera `If-Modified-Since`:

Para ilustrar cómo opera el GET condicional, veamos un ejemplo. En primer lugar, una caché proxy envía un mensaje de solicitud a un servidor web en nombre de un navegador que realiza una solicitud:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

En segundo lugar, el servidor web envía a la caché un mensaje de respuesta con el objeto solicitado:

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/gif

(datos datos datos datos ...)
```

La caché reenvía el objeto al navegador que lo ha solicitado pero también lo almacena localmente. Y lo que es más importante, la caché también almacena la fecha de la última modificación junto con el objeto. En tercer lugar, una semana después, otro navegador solicita el mismo objeto a través de la caché y el objeto todavía se encuentra almacenado allí. Puesto que este objeto puede haber sido modificado en el servidor web en el transcurso de la semana, la caché realiza una comprobación de actualización ejecutando un GET condicional. Específicamente, la caché envía:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

Observe que el valor de la línea de cabecera `If-modified-since`: es exactamente igual al valor de la línea de cabecera `Last-Modified`: que fue enviada por el servidor hace una semana. Este GET condicional le pide al servidor que envíe el objeto sólo si éste ha sido modificado después de la última fecha especificada. Suponga que el objeto no ha sido modificado desde el 9 de septiembre de 2015 a las 09:23:24. Por último y en cuarto lugar, el servidor web envía un mensaje de respuesta a la caché:

```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)

(cuerpo de entidad vacío)
```

Vemos que en respuesta al GET condicional el servidor web envía un mensaje de respuesta, pero no incluye el objeto solicitado en el mismo. Si incluyera el objeto solicitado sólo conseguiría desperdiciar ancho de banda e incrementar el tiempo de respuesta percibido por el usuario, especialmente si el tamaño del objeto es grande. Observe que este último mensaje de respuesta muestra en

la línea de estado el texto 304 Not Modified (no modificado), lo que indica a la caché que puede reenviar la copia del objeto que tiene en caché al navegador que lo haya solicitado.

Aquí terminamos nuestra exposición acerca de HTTP, el primer protocolo de Internet (un protocolo de la capa de aplicación) que hemos estudiado en detalle. Hemos examinado el formato de los mensajes HTTP y las acciones realizadas por el servidor y el cliente web según se envían y reciben estos mensajes. También hemos visto algo acerca de la infraestructura de las aplicaciones web, incluyendo las cachés, las cookies y las bases de datos back-end, elementos todos ellos que están ligados de alguna manera con el protocolo HTTP.

2.3 Correo electrónico en Internet

El correo electrónico ha existido desde que Internet viera la luz. Era la aplicación más popular cuando Internet estaba en sus comienzos [Segaller 1998] y a lo largo de los años se ha ido convirtiendo en una aplicación cada vez más elaborada y potente. En la actualidad continúa siendo una de las aplicaciones más importantes y utilizadas de Internet.

Al igual que el servicio ordinario de correo postal, el correo electrónico es un medio de comunicación asíncrono (las personas envían y leen los mensajes cuando les conviene, sin tener que coordinarse con las agendas de otras personas). En contraste con el correo postal, el correo electrónico es rápido, fácil de distribuir y barato. Las aplicaciones modernas de correo electrónico disponen de muchas características potentes. Mediante las listas de correo, se pueden enviar mensajes de correo electrónico y correo basura (spam) a miles de destinatarios a un mismo tiempo. A menudo, los mensajes de correo electrónico incluyen adjuntos, hipervínculos, texto en formato HTML y fotografías.

En esta sección vamos a examinar los protocolos de la capa de aplicación que forman la base del correo electrónico en Internet. Pero antes de sumergirnos en una explicación detallada acerca de estos protocolos, vamos a proporcionar una visión de conjunto del sistema de correo de Internet y sus componentes fundamentales.

La Figura 2.14 muestra una visión general del sistema de correo de Internet. A partir de este diagrama, vemos que existen tres componentes principales: **agentes de usuario, servidores de correo y el Protocolo simple de transferencia de correo (SMTP, Simple Mail Transfer Protocol)**. A continuación vamos a describir cada uno de estos componentes en el contexto de un emisor, Alicia, que envía un mensaje de correo electrónico a un destinatario, Benito. Los agentes de usuario permiten a los usuarios leer, responder, reenviar, guardar y componer mensajes. Microsoft Outlook, y Apple Mail son ejemplos de agentes de usuario para correo electrónico. Cuando Alicia termina de componer su mensaje, su agente de usuario envía el mensaje a su servidor de correo, donde el mensaje es colocado en la cola de mensajes salientes del servidor de correo. Cuando Benito quiere leer un mensaje, su agente de usuario recupera el mensaje de su buzón, que se encuentra en el servidor de correo.

Los servidores de correo forman el núcleo de la infraestructura del correo electrónico. Cada destinatario, como por ejemplo Benito, tiene un **buzón de correo** ubicado en uno de los servidores de correo. El buzón de Benito gestiona y mantiene los mensajes que le han sido enviados. Un mensaje típico inicia su viaje en el agente de usuario del emisor, viaja hasta el servidor de correo del emisor y luego hasta el servidor de correo del destinatario, donde es depositado en el buzón del mismo. Cuando Benito quiere acceder a los mensajes contenidos en su buzón, el servidor de correo que lo contiene autentica a Benito (mediante el nombre de usuario y la contraseña). El servidor de correo de Alicia también tiene que ocuparse de los fallos que se producen en el servidor de correo de Benito. Si el servidor de Alicia no puede enviar el mensaje de correo al servidor de Benito, entonces el servidor de Alicia mantiene el mensaje en una **cola de mensajes** e intenta enviarlo más tarde. Normalmente, los reintentos de envío se realizan más o menos cada 30 minutos; si después de varios días no se ha conseguido, el servidor elimina el mensaje y se lo notifica al emisor (Alicia) mediante un mensaje de correo electrónico.

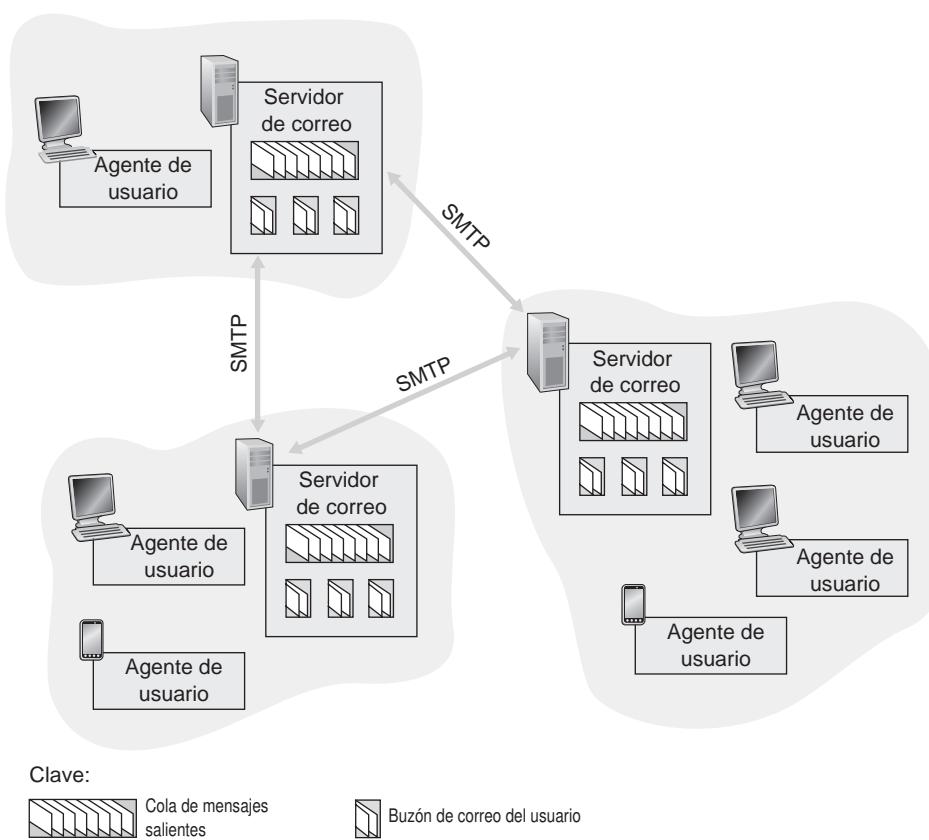


Figura 2.14 ♦ Esquema general del sistema de correo electrónico de Internet.

SMTP es el principal protocolo de la capa de aplicación para el correo electrónico por Internet. Utiliza el servicio de transferencia de datos fiable de TCP para transferir el correo desde el servidor de correo del emisor al servidor de correo del destinatario. Al igual que la mayoría de los protocolos de la capa de aplicación, SMTP tiene dos lados: el lado del cliente, que se ejecuta en el servidor de correo del emisor, y el lado del servidor, que se ejecuta en el servidor de correo del destinatario. Tanto el lado del cliente como el del servidor de SMTP se ejecutan en todos los servidores de correo. Cuando un servidor de correo envía mensajes de correo a otros servidores de correo, actúa como un cliente SMTP. Cuando un servidor de correo recibe correo de otros servidores, actúa como un servidor SMTP.

2.3.1 SMTP

El protocolo SMTP, que está definido en el documento RFC 5321, es el corazón del correo electrónico por Internet. Como hemos mencionado anteriormente, SMTP transfiere mensajes desde los servidores de correo de los emisores a los servidores de correo de los destinatarios. SMTP es mucho más antiguo que HTTP. (El RFC original que se ocupa de SMTP data de 1982 y SMTP ya existía bastante tiempo antes.) Aunque SMTP tienen muchas cualidades maravillosas, como prueba su presencia en Internet, es una tecnología heredada que utiliza algunas funcionalidades arcaicas. Por ejemplo, restringe el cuerpo (no sólo las cabeceras) de todos los mensajes a formato ASCII de 7 bits. Esta restricción tenía sentido a principios de la década de 1980, cuando la capacidad de transmisión era escasa y nadie enviaba mensajes con adjuntos o imágenes de gran tamaño, o archivos de audio o vídeo. Pero actualmente, en la era multimedia, la restricción del formato ASCII de 7 bits causa muchos problemas: requiere que los datos binarios multimedia se codifiquen a

ASCII antes de ser transmitidos a través de SMTP y requiere que el correspondiente mensaje ASCII sea decodificado de vuelta a binario una vez realizado el transporte SMTP. Recuerde, como hemos visto en la Sección 2.2, que HTTP no precisa que los datos multimedia sean codificados a ASCII antes de ser transferidos.

Para ilustrar el funcionamiento básico de SMTP vamos a recurrir a un escenario ya conocido. Suponga que Alicia desea enviar a Benito un sencillo mensaje ASCII.

1. Alicia invoca a su agente de usuario para correo electrónico, proporciona la dirección de correo electrónico de Benito (por ejemplo, benito@unaescuela.edu), compone un mensaje e indica al agente de usuario que lo envíe.
2. El agente de usuario de Alicia envía el mensaje al servidor de correo de ella, donde es colocado en una cola de mensajes.
3. El lado del cliente de SMTP, que se ejecuta en el servidor de correo de Alicia, ve el mensaje en la cola de mensajes. Abre una conexión TCP con un servidor SMTP, que se ejecuta en el servidor de correo de Benito.
4. Después de la fase de negociación inicial de SMTP, el cliente SMTP envía el mensaje de Alicia a través de la conexión TCP.
5. En el servidor de correo de Benito, el lado del servidor de SMTP recibe el mensaje. El servidor de correo de Benito coloca entonces el mensaje en el buzón de Benito.
6. Benito invoca a su agente de usuario para leer el mensaje cuando le apetezca.

En la Figura 2.15 se resume este escenario.

Es importante observar que normalmente **SMTP no utiliza servidores de correo intermedios** para enviar correo, incluso cuando los dos servidores de correo se encuentran en extremos opuestos del mundo. Si el servidor de Alicia está en Hong Kong y el de Benito está en St. Louis, la conexión TCP será una conexión directa entre los servidores de Hong Kong y St. Louis. En particular, si el servidor de correo de Benito está fuera de servicio, el servidor de Alicia conservará el mensaje y lo intentará de nuevo (el mensaje no se deja en un servidor de correo intermedio).

Veamos en detalle cómo transfiere SMTP un mensaje desde un servidor de correo emisor a un servidor de correo receptor. Comprobaremos que el protocolo SMTP presenta muchas similitudes con los protocolos empleados por las personas para las interacciones cara a cara. En primer lugar, el cliente SMTP (que se ejecuta en el host servidor de correo emisor) establece una conexión TCP con el **puerto 25** del servidor SMTP (que se ejecuta en el host servidor de correo receptor). Si el servidor no está operativo, el cliente lo intentará más tarde. Una vez que se ha establecido la conexión, el servidor y el cliente llevan a cabo el proceso de negociación de la capa de aplicación (al igual que las personas, que antes de intercambiar información se presentan, los clientes y servidores SMTP se presentan a sí mismos antes de transferir la información). Durante esta fase de negociación SMTP, el cliente SMTP especifica la dirección de correo electrónico del emisor (la persona que ha generado el mensaje) y la dirección de correo electrónico del destinatario. Una vez que el cliente y el servidor

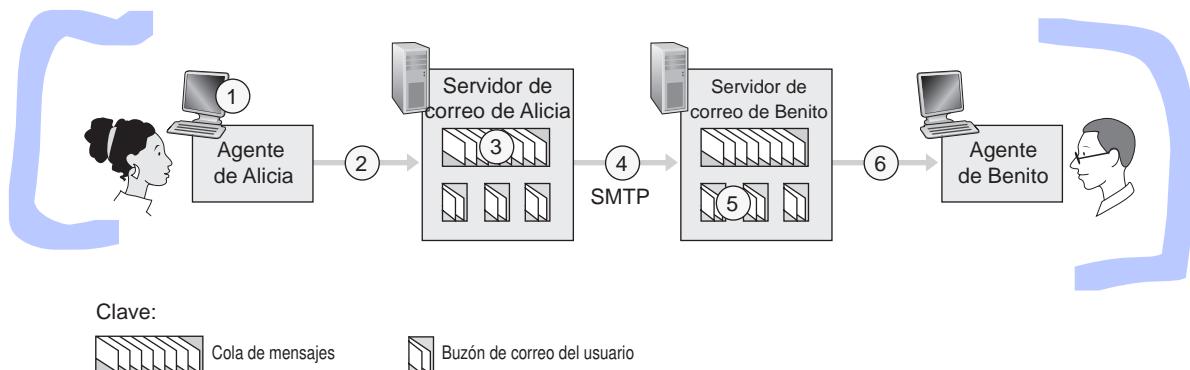


Figura 2.15 ♦ Alicia envía un mensaje a Benito.

SMTP se han presentado a sí mismos, el cliente envía el mensaje. SMTP cuenta con el servicio de transferencia de datos fiable de TCP para transferir el mensaje al servidor sin errores. El cliente repite entonces este proceso a través de la misma conexión TCP si tiene que enviar otros mensajes al servidor; en caso contrario, indica a TCP que cierre la conexión.

Veamos ahora una transcripción de ejemplo de los mensajes intercambiados entre un cliente SMTP (C) y un servidor SMTP (S). El nombre de host del cliente es `crepes.fr` y el nombre de host del servidor es `hamburger.edu`. Las líneas de texto ASCII precedidas por C: son exactamente las líneas que el cliente envía a su socket TCP y las líneas de texto ASCII precedidas por S: son las líneas que el servidor envía a su socket TCP. La siguiente transcripción comienza tan pronto como se establece la conexión TCP.

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250 Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alicia@crepes.fr>
S: 250 alicia@crepes.fr ... Sender ok
C: RCPT TO: <benito@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: ¿Te gusta el ketchup?
C: ¿Y los pepinillos en vinagre?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

En el ejemplo anterior, el cliente envía un mensaje (“¿Te gusta el ketchup? ¿Y los pepinillos en vinagre?”) desde el servidor de correo `crepes.fr` al servidor de correo `hamburger.edu`. Como parte del diálogo, el cliente ejecuta cinco comandos: HELO (una abbreviatura de HELLO), MAIL FROM, RCPT TO, DATA y QUIT. Estos comandos se explican por sí mismos. El cliente también envía una línea que consta únicamente de un punto, que indica el final del mensaje para el servidor. (En la jerga ASCII, cada mensaje termina con CRLF. CRLF, donde CR y LF se corresponden con el retorno de carro y el salto de línea, respectivamente.) El servidor responde a cada comando, teniendo cada una de las respuestas un código de respuesta y una explicación (opcional) en inglés. Hemos mencionado que **SMTP utiliza conexiones persistentes**: si el servidor de correo emisor tiene varios mensajes que enviar al mismo servidor de correo receptor, puede enviar todos los mensajes a través de la misma conexión TCP. Para cada mensaje, el cliente inicia el proceso con un nuevo comando MAIL FROM: `crepes.fr`, designa el final del mensaje con un único punto y ejecuta el comando QUIT sólo después de que todos los mensajes hayan sido enviados.

Es muy recomendable que utilice Telnet para establecer un diálogo directo con un servidor SMTP. Para ello, ejecute:

```
telnet nombreServidor 25
```

donde `nombreServidor` es el nombre de un servidor de correo local. Al hacer esto, simplemente está estableciendo una conexión TCP entre su host local y el servidor de correo. Después de escribir esta línea, debería recibir inmediatamente la respuesta 220 del servidor. A continuación, ejecute los comandos SMTP HELO, MAIL FROM, RCPT TO, DATA, CRLF.CRLF y QUIT en los instantes apropiados. También es extremadamente recomendable que realice la Tarea de programación 2 incluida al final del capítulo. En esta tarea, tendrá que crear un agente de usuario simple que implemente el lado del cliente de SMTP. Esto le permitirá enviar un mensaje de correo electrónico a un destinatario arbitrario a través de un servidor de correo local.

2.3.2 Comparación con HTTP

Vamos ahora a comparar brevemente SMTP con HTTP. Ambos protocolos se emplean para transferir archivos de un host a otro: HTTP transfiere archivos (también denominados objetos) desde un servidor web a un cliente web (normalmente, un navegador); SMTP transfiere archivos (es decir, mensajes de correo electrónico) desde un servidor de correo a otro servidor de correo. Para transferir los archivos, tanto HTTP persistente como SMTP emplean conexiones persistentes. Por tanto, ambos protocolos tienen características comunes. Sin embargo, también presentan algunas diferencias. En primer lugar, **HTTP es principalmente un protocolo *pull*** (protocolo de extracción): alguien carga la información en un servidor web y los usuarios utilizan HTTP para extraer la información del servidor cuando desean. En concreto, la máquina que desea recibir el archivo inicia la conexión TCP. Por el contrario, **SMTP es fundamentalmente un protocolo *push*** (protocolo de inserción): el servidor de correo emisor introduce el archivo en el servidor de correo receptor. En concreto, la máquina que desea enviar el archivo inicia la conexión TCP.

Una segunda diferencia, a la que hemos aludido anteriormente, es que **SMTP requiere que cada mensaje, incluyendo el cuerpo de cada mensaje, esté en el formato ASCII de 7 bits**. Si el mensaje contiene caracteres que no corresponden a dicho formato (como por ejemplo, caracteres acentuados) o contiene datos binarios (como por ejemplo un archivo de imagen), entonces el mensaje tiene que ser codificado en ASCII de 7 bits. **Los datos HTTP no imponen esta restricción**.

Una tercera diferencia importante tiene que ver con cómo se maneja un documento que conste de texto e imágenes (junto con posiblemente otros tipos multimedia). Como hemos visto en la Sección 2.2, **HTTP encapsula cada objeto en su propio mensaje de respuesta HTTP. El correo Internet incluye todos los objetos del mensaje en un mismo mensaje**.

2.3.3 Formatos de los mensajes de correo

Cuando Alicia escribe una carta por correo ordinario a Benito, puede incluir todo tipo de información accesoria en la parte superior de la carta, como la dirección de Benito, su propia dirección de respuesta y la fecha. De forma similar, cuando una persona envía un mensaje de correo electrónico a otra, una cabecera que contiene la información administrativa antecede al cuerpo del mensaje. Esta información se incluye en una serie de líneas de cabecera, que están definidas en el documento RFC 5322. Las líneas de cabecera y el cuerpo del mensaje se separan mediante una línea en blanco (es decir, mediante CRLF). RFC 5322 especifica el formato exacto de las líneas de cabecera, así como sus interpretaciones semánticas. Como con HTTP, cada línea de cabecera contiene texto legible, que consta de una palabra clave seguida de dos puntos y de un valor. Algunas de las palabras clave son obligatorias y otras son opcionales. Toda cabecera tiene que estar formada por una línea de cabecera `From:` y una línea de cabecera `To:`; también puede incluir una línea `Subject:`, así como otras líneas de cabeceraopcionales. Es importante destacar que estas líneas de cabecera son *diferentes* de los comandos SMTP que hemos estudiado en la Sección 2.3.1 (incluso aunque contengan algunas palabra comunes como “`from`” y “`to`”). Los comandos vistos en esa sección forman parte del protocolo de negociación de SMTP; las líneas de cabecera examinadas en esta sección forman parte del propio mensaje de correo.

Una cabecera de mensaje típica sería como la siguiente:

```
From: alicia@crepes.fr
To: benito@hamburger.edu
Subject: Búsqueda del significado de la vida.
```

Después del mensaje de cabecera se incluye una línea en blanco y, a continuación, el cuerpo del mensaje (en ASCII). Debería utilizar Telnet para enviar un mensaje a un servidor de correo que contenga varias líneas de cabecera, incluyendo la línea de cabecera del asunto `Subject:`. Para ello, ejecute el comando `telnet NombreServidor 25`, como se ha explicado en la Sección 2.3.1.

2.3.4 Protocolos de acceso para correo electrónico

Una vez que SMTP envía el mensaje del servidor de correo de Alicia al servidor de correo de Benito, el mensaje se coloca en el buzón de este último. A lo largo de esta exposición, hemos supuesto tácitamente que Benito lee su correo registrándose en el host servidor y utilizando después un lector de correo que se ejecuta en dicho host. Hasta principios de la década de 1990 esta era la forma habitual de hacer las cosas. Pero, actualmente, el acceso al correo electrónico utiliza una arquitectura cliente-servidor; el usuario típico lee el correo electrónico con un cliente que se ejecuta en el sistema terminal del usuario, por ejemplo, en un PC de la oficina, en un portátil o en un smartphone. Ejecutando un cliente de correo en un PC local, los usuarios disponen de un rico conjunto de funcionalidades, entre las que se incluye la posibilidad de visualizar documentos adjuntos y mensajes multimedia.

Puesto que Benito (el destinatario) ejecuta su agente de usuario en su PC local, es natural que considere también incluir un servidor de correo en su PC local. De esta forma, el servidor de correo de Alicia dialogará directamente con el PC de Benito. Sin embargo, hay un problema con este método. Recuerde que un servidor de correo gestiona buzones de correo y ejecuta los lados del cliente y del servidor de SMTP. Si el servidor de correo de Benito residiera en su PC local, entonces el PC de Benito tendría que estar siempre encendido y conectado a Internet para recibir los nuevos correos que pudieran llegar en cualquier momento. Pero esto es impracticable para muchos usuarios de Internet. En su lugar, un usuario típico ejecuta un agente de usuario en el PC local pero accede a su buzón de correo almacenado en un servidor de correo compartido que siempre está encendido. Este servidor es compartido con otros usuarios y, normalmente, mantenido por el ISP del usuario (por ejemplo, una universidad o una empresa).

Ahora vamos a ver qué ruta sigue un mensaje de correo electrónico que Alicia envía a Benito. Acabamos de estudiar que en algún punto a lo largo de la ruta el mensaje de correo electrónico tiene que ser depositado en el servidor de correo de Benito. Esto se podría conseguir fácilmente haciendo que el agente de usuario de Alicia envíe el mensaje directamente al servidor de correo de Benito. Y esto se podría hacer utilizando SMTP (de hecho, SMTP ha sido diseñado para llevar el correo electrónico de un host a otro). Sin embargo, normalmente el agente de usuario del emisor no se comunica directamente con el servidor de correo del destinatario. En su lugar, como se muestra en la Figura 2.16, el agente de usuario de Alicia utiliza SMTP para introducir el mensaje de correo en su propio servidor de correo, y a continuación el servidor de correo de Alicia utiliza SMTP (como un cliente SMTP) para pasar el mensaje al servidor de correo de Benito. ¿Por qué este procedimiento en dos pasos? Fundamentalmente, porque sin la retransmisión a través del servidor de correo de Alicia, el agente de usuario de esta no tiene ninguna forma de acceder a un servidor de correo de destino inalcanzable. Al hacer que Alicia deposite primero el mensaje en su propio servidor de correo, este puede intentar una y otra vez enviar el mensaje al servidor de correo de Benito, por ejemplo, cada 30 minutos, hasta que el servidor de Benito esté de nuevo operativo. (Y si el servidor de correo de Alicia no funciona, entonces ella tiene el recurso de ¡quejarse al administrador del sistema!). El RFC que se ocupa de SMTP define cómo se pueden utilizar los comandos SMTP para transmitir un mensaje a través de varios servidores SMTP.

¡Pero todavía falta una pieza del puzzle! ¿Cómo un destinatario como Benito, que ejecuta un agente de usuario en su PC local, obtiene sus mensajes, que se encuentran en un servidor de correo de su ISP? Tenga en cuenta que el agente de usuario de Benito no puede utilizar SMTP para obtener los mensajes porque es una operación de extracción (pull) mientras que SMTP es un protocolo push (de inserción). Así, el puzzle se completa añadiendo un protocolo especial de acceso al correo que permita transferir los mensajes del servidor de correo de Benito a su PC local. Actualmente existen varios protocolos de acceso a correo electrónico populares, entre los que se incluyen el **Protocolo de oficina de correos versión 3 (POP3, Post Office Protocol—Version 3)**, el **Protocolo de acceso de correo de Internet (IMAP, Internet Mail Access Protocol)** y **HTTP**.

La Figura 2.16 proporciona un resumen de los protocolos que se utilizan para el correo de Internet: SMTP se emplea para transferir correo desde el servidor de correo del emisor al servidor

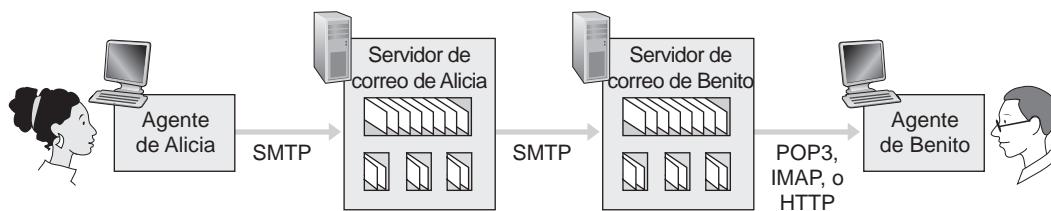


Figura 2.16 ♦ Protocolos de correo electrónico y entidades que los utilizan.

de correo del destinatario; SMTP también se utiliza para transferir correo desde el agente de usuario del emisor al servidor de correo del mismo. Para transferir los mensajes de correo almacenados en el servidor de correo del destinatario al agente de usuario del mismo se emplea un protocolo de acceso a correo, como POP3.

POP3

POP3 es un protocolo de acceso a correo extremadamente simple. Está definido en [RFC 1939], que es un documento corto y bastante claro. Dado que el protocolo es tan simple, su funcionalidad es bastante limitada. POP3 se inicia cuando el agente de usuario (el cliente) abre una conexión TCP en el puerto 110 al servidor de correo (el servidor). Una vez establecida la conexión TCP, POP3 pasa a través de tres fases: autorización, transacción y actualización. Durante la primera fase, la autorización, el agente de usuario envía un nombre de usuario y una contraseña (en texto legible) para autenticar al usuario. Durante la segunda fase, la de transacción, el agente de usuario recupera los mensajes; también durante esta fase, el agente de usuario puede marcar los mensajes para borrado, eliminar las marcas de borrado y obtener estadísticas de correo. La tercera fase, la actualización, tiene lugar después que el cliente haya ejecutado el comando `quit`, terminando la sesión POP3; en este instante, el servidor de correo borra los mensajes que han sido marcados para borrado.

En una transacción POP3, el agente de usuario ejecuta comandos y el servidor devuelve para cada comando una respuesta. Existen dos posibles respuestas: `+OK` (seguida en ocasiones por una serie de datos servidor-cliente), utilizada por el servidor para indicar que el comando anterior era correcto; y `-ERR`, utilizada por el servidor para indicar que había algún error en el comando anterior.

La fase de autorización tiene dos comandos principales: `user <nombreeusuario>` y `pass <contraseña>`. Para ilustrar estos dos comandos, le sugerimos que establezca una conexión Telnet directamente en un servidor POP3, utilizando el puerto 110, y ejecute estos comandos. Suponga que `mailServer` es el nombre de su servidor de correo. Verá algo similar a lo siguiente:

```
telnet mailServer 110
+OK POP3 server ready
user benito
+OK
pass hambre
+OK user successfully logged on
```

Si escribe mal un comando, el servidor POP3 le responderá con un mensaje `-ERR`.

Abordemos ahora la fase de transacción. Un agente de usuario que utilice POP3 suele ser configurado (por el usuario) para “descargar y borrar” o para “descargar y guardar”. La secuencia de comandos que ejecute un agente de usuario POP3 dependerá de en cuál de estos dos modos esté operando. En el modo descargar y borrar, el agente de usuario ejecutará los comandos `list`, `retry` y `dele`. Por ejemplo, suponga que el usuario tiene dos mensajes en su buzón de correo. En el diálogo que proporcionamos a continuación, `C:` (que quiere decir cliente) es el agente de usuario y `S:` (que quiere decir servidor) es el servidor de correo. La transacción será similar a lo siguiente:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: (bla bla ...
S: .....
S: .....bla)
S: .
C: dele 1
C: retr 2
S: (bla bla ...
S: .....
S: .....bla)
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

En primer lugar, el agente de usuario pide al servidor de correo que le informe del tamaño de cada uno de los mensajes almacenados. A continuación, el agente de usuario recupera y borra cada uno de los mensajes del servidor. Observe que después de la fase de autorización, el agente de usuario solo ha utilizado cuatro comandos: `list`, `retr`, `dele` y `quit`. La sintaxis de estos comandos está definida en el documento RFC 1939. Después de procesar el comando `quit`, el servidor POP3 entra en la fase de actualización y borra los mensajes 1 y 2 del buzón.

Un problema en este modo de descarga y borrado es que el destinatario, Benito, puede ser algo nómada y puede desear poder acceder a sus mensajes de correo desde varias máquinas; por ejemplo, desde el PC de la oficina, el PC de su casa y el portátil. El modo de descarga y borrado reparte los mensajes de correo de Benito entre estas tres máquinas; en particular, si Benito lee un mensaje en el PC de la oficina, no podrá volver a leerlo en el portátil en su casa por la noche. En el modo descargar y guardar, el agente de usuario deja los mensajes en el servidor de correo después de que se hayan descargado. En este caso, Benito podrá volver a leer los mensajes desde distintas máquinas; puede acceder a un mensaje en el trabajo y luego también en su casa si lo desea.

Durante una sesión POP3 entre un agente de usuario y el servidor de correo, el servidor POP3 mantiene cierta información de estado; en concreto, mantiene la relación de los mensajes de usuario que han sido marcados para ser borrados. Sin embargo, el servidor POP3 no conserva la información de estado de una sesión POP3 a otra. Esta falta de memoria del estado entre sesiones simplifica enormemente la implementación de un servidor POP3.

IMAP

Con el acceso POP3, una vez que Benito ha descargado sus mensajes en la máquina local, puede crear carpetas de correo y mover los mensajes descargados a las carpetas. A continuación, puede borrar los mensajes, pasárselos a carpetas y realizar búsquedas de mensajes (por nombre del emisor o asunto). Pero este paradigma, es decir, las carpetas y los mensajes guardados en la máquina local, plantea un problema para el usuario nómada, que preferirá mantener una jerarquía de carpetas en un servidor remoto al que pueda acceder desde cualquier computadora. Esto no es posible con POP3 (el protocolo POP3 no proporciona ningún medio al usuario para crear carpetas remotas y asignar mensajes a las mismas).

Para resolver este y otros problemas se inventó el protocolo IMAP, definido en [RFC 3501]. Al igual que POP3, IMAP es un protocolo de acceso a correo. Ofrece muchas más funcionalidades que POP3, pero también es significativamente más complejo (y, por tanto, las implementaciones del lado del cliente y del lado del servidor son bastante más complejas).

Un servidor IMAP asociará cada mensaje con una carpeta; cuando un mensaje llega al servidor, se asocia con la carpeta INBOX (Bandeja de entrada) del destinatario, el cual puede entonces pasar el mensaje a una nueva carpeta creada por el usuario, leer el mensaje, borrarlo, etc. El protocolo IMAP proporciona comandos que permiten a los usuarios crear carpetas y mover los mensajes de una carpeta a otra. IMAP también proporciona comandos que permiten a los usuarios realizar búsquedas en carpetas remotas para localizar mensajes que cumplan unos determinados criterios. Observe que, a diferencia de POP3, un servidor IMAP mantiene información acerca del estado a lo largo de las sesiones IMAP, como por ejemplo, los nombres de las carpetas y los mensajes asociados con cada una de ellas.

Otra importante característica de IMAP es que dispone de comandos que permiten a un agente de usuario obtener partes componentes de los mensajes. Por ejemplo, un agente de usuario puede obtener solo la cabecera del mensaje o solo una parte de un mensaje MIME de varias partes. Esta característica resulta útil cuando se emplea una conexión con un ancho de banda pequeño (por ejemplo, un enlace de módem de baja velocidad) entre el agente de usuario y su servidor de correo. Con una conexión de ancho de banda pequeño, puede que el usuario no quiera descargar todos los mensajes guardados en su buzón y que desee saltarse los mensajes largos que pueda haber; por ejemplo un archivo de audio o de vídeo.

Correo electrónico web

Actualmente, cada vez más usuarios envían y acceden a su correo electrónico a través de sus navegadores web. Hotmail introdujo a mediados de la década de 1990 el acceso basado en la Web; actualmente, también ofrece este servicio Yahoo, Google, así como casi todas las principales universidades y empresas. Con este servicio, el agente de usuario es un navegador web corriente y el usuario se comunica con su buzón remoto a través de HTTP. Cuando un destinatario, como Benito, desea acceder a un mensaje de su buzón, este es enviado desde el servidor de correo de Benito al navegador del mismo utilizando el protocolo HTTP en lugar de los protocolos POP3 o IMAP. Cuando un emisor, como Alicia, desea enviar un mensaje de correo electrónico, este es transmitido desde su navegador a su servidor de correo a través de HTTP en lugar de mediante SMTP. Sin embargo, el servidor de correo de Alicia, continúa enviando mensajes a, y recibiendo mensajes de, otros servidores de correo que emplean SMTP.

2.4 DNS: el servicio de directorio de Internet

Las personas podemos ser identificadas de muchas maneras. Por ejemplo, podemos ser identificadas por el nombre que aparece en nuestro certificado de nacimiento, por nuestro número de la seguridad social o por el número del carnet de conducir. Aunque cada uno de estos datos se puede utilizar para identificar a las personas, dentro de un determinado contexto un identificador puede ser más apropiado que otro. Por ejemplo, las computadoras del IRS (la terrible agencia tributaria de Estados Unidos) prefieren utilizar los números de la seguridad social de longitud fija que el nombre que aparece en el certificado de nacimiento. Por otro lado, las personas prefieren emplear el nombre que figura en los certificados de nacimiento, más fáciles de recordar, a los números de la seguridad social (puede imaginar que alguien le dijera “Hola. Mi nombre es 132-67-9875. Este es mi marido, 178-87-1146.”)

No solo las personas podemos ser identificadas de diversas formas, los hosts de Internet también. Un identificador para los hosts es el **nombre de host**. Los nombres de host, como por ejemplo www.facebook.com, www.google.com, gaia.cs.umass.edu, son mnemónicos y son, por tanto, entendidos por las personas. Sin embargo, los nombres de host proporcionan poca o ninguna información acerca de la ubicación del host dentro de Internet. (Un nombre de host como www.eurecom.fr, que termina con el código de país, .fr, nos informa de que es probable que el host

se encuentre en Francia, pero esto no dice mucho más.) Además, puesto que los nombres de host pueden constar de caracteres alfanuméricos de longitud variable, podrían ser difíciles de procesar por los routers. Por estas razones, los hosts también se identifican mediante **direcciones IP**.

En el Capítulo 4 veremos en detalle las direcciones IP, pero le resultará útil leer ahora una breve exposición sobre las mismas. Una dirección IP consta de cuatro bytes y sigue una rígida estructura jerárquica. El aspecto de una dirección IP es, por ejemplo, 121.7.106.83, donde cada punto separa uno de los bytes expresados en notación decimal con valores comprendidos entre 0 y 255. Una dirección IP es jerárquica porque al explorar la dirección de izquierda a derecha, se obtiene información cada vez más específica acerca del lugar en el que está ubicado el host dentro de Internet (es decir, en qué red de la red de redes). De forma similar, cuando examinamos una dirección postal de abajo a arriba, obtenemos cada vez información más específica acerca del lugar definido por la dirección.

2.4.1 Servicios proporcionados por DNS

Acabamos de ver que hay dos formas de identificar un host, mediante un nombre de host y mediante una dirección IP. Las personas prefieren utilizar como identificador el nombre de host, mientras que los routers prefieren emplear las direcciones IP de longitud fija y que siguen una estructura jerárquica. Para reconciliar estas preferencias necesitamos un servicio de directorio que traduzca los nombres de host en direcciones IP. Esta es la tarea principal que lleva a cabo el **Sistema de nombres de dominio (DNS, Domain Name System)** de Internet. DNS es (1) una base de datos distribuida implementada en una jerarquía de servidores DNS y (2) un protocolo de la capa de aplicación que permite a los hosts consultar la base de datos distribuida. Los servidores DNS suelen ser máquinas UNIX que ejecutan el software BIND (*Berkeley Internet Name Domain, Dominio de nombres de Internet de Berkeley*) [BIND 2016]. El protocolo DNS se ejecuta sobre UDP y utiliza el puerto 53.

Otros protocolos de la capa de aplicación, como HTTP y SMTP, emplean habitualmente DNS para traducir los nombres de host suministrados por el usuario en direcciones IP. Por ejemplo, veamos qué ocurre cuando un navegador (es decir, un cliente HTTP), que se ejecuta en un determinado host de usuario, solicita el URL `www.unaescuela.edu/index.html`. Para que el host del usuario pueda enviar un mensaje de solicitud HTTP al servidor web `www.unaescuela.edu`, el host del usuario debe obtener en primer lugar la dirección IP de `www.unaescuela.edu`. Esto se hace del siguiente modo:

1. La propia máquina cliente ejecuta el lado del cliente de la aplicación DNS.
2. El navegador extrae el nombre de host, `www.unaescuela.edu`, del URL y pasa el nombre de host al lado del cliente de la aplicación DNS.
3. El cliente DNS envía una consulta que contiene el nombre de host a un servidor DNS.
4. El cliente DNS recibe finalmente una respuesta, que incluye la dirección IP correspondiente al nombre del host.
5. Una vez que el navegador recibe la dirección IP del servidor DNS, puede iniciar una conexión TCP con el proceso servidor HTTP localizado en el puerto 80 en esa dirección IP.

Podemos ver a partir de este ejemplo que DNS añade un retardo adicional, en ocasiones, sustancial, a las aplicaciones de Internet que le utilizan. Afortunadamente, como veremos más adelante, la dirección IP deseada a menudo está almacenada en caché en un servidor DNS “próximo”, lo que ayuda a reducir el tráfico de red DNS, así como el retardo medio del servicio DNS.

DNS proporciona algunos otros servicios importantes además de la traducción de los nombres de host en direcciones IP:

- **Alias de host.** Un host con un nombre complicado puede tener uno o más alias. Por ejemplo, un nombre de host como `relay1.west-coast.enterprise.com` podría tener, digamos, dos alias como `enterprise.com` y `www.enterprise.com`. En este caso, el nombre de host

`relay1.west-coast.enterprise.com` se dice que es el **nombre de host canónico**. Los alias de nombres de host, cuando existen, normalmente son más mnemónicos que los nombres canónicos. Una aplicación puede invocar DNS para obtener el nombre de host canónico para un determinado alias, así como la dirección IP del host.

- **Alias del servidor de correo.** Por razones obvias, es enormemente deseable que las direcciones de correo electrónico sean mnemónicas. Por ejemplo, si Benito tiene una cuenta de Yahoo Mail, su dirección de correo puede ser tan simple como `benito@yahoo.mail`. Sin embargo, el nombre de host del servidor de correo de Yahoo es más complicado y mucho menos mnemónico que simplemente `yahoo.com` (por ejemplo, el nombre canónico podría ser algo parecido a `relay1.west-coast.yahoo.com`). Una aplicación de correo puede invocar al servicio DNS para obtener el nombre de host canónico para un determinado alias, así como la dirección IP del host. De hecho, el registro MX (véase más adelante) permite al servidor de correo y al servidor web de una empresa tener nombres de host (con alias) iguales; por ejemplo, tanto el servidor web como el servidor de correo de una empresa se pueden llamar `enterprise.com`.
- **Distribución de carga.** DNS también se emplea para realizar la distribución de carga entre servidores replicados, como los servidores web replicados. Los sitios con una gran carga de trabajo, como `cnn.com`, están replicados en varios servidores, ejecutándose cada servidor en un sistema terminal distinto y teniendo cada uno una dirección IP diferente. Para los servidores web replicados hay asociado un *conjunto* de direcciones IP con un único nombre de host canónico. La base de datos DNS contiene este conjunto de direcciones IP. Cuando los clientes realizan una consulta DNS sobre un nombre asignado a un conjunto de direcciones, el servidor responde con el conjunto completo de direcciones IP, pero rota el orden de las direcciones en cada respuesta. Dado que normalmente un cliente envía su mensaje de solicitud HTTP a la dirección IP que aparece en primer lugar dentro del conjunto, la rotación DNS distribuye el tráfico entre los servidores replicados. La rotación DNS también se emplea para el correo electrónico de modo que múltiples servidores de correo pueden tener el mismo alias. Además, las empresas de distribución de contenido como Akamai han utilizado DNS de formas muy sofisticadas [Dilley 2002] para proporcionar la distribución de contenido web (véase la Sección 2.6.3).

DNS está especificado en los documentos RFC 1034 y RFC 1035, y actualizado en varios RFC adicionales. Es un sistema complejo y aquí solo hemos visto los aspectos básicos de su funcionamiento. El lector interesado puede consultar estos RFC y el libro de Abitz y Liu [Abitz 1993]; también puede consultar el documento retrospectivo [Mockapetris 1988], que proporciona una descripción sobre qué es DNS y el por qué de DNS; asimismo puede ver [Mockapetris 2005].

2.4.2 Cómo funciona DNS

Ahora vamos a presentar a alto nivel cómo funciona DNS. Nos centraremos en el servicio de traducción nombre de host-dirección IP.

Suponga que una determinada aplicación (como por ejemplo un navegador web o un lector de correo), que se ejecuta en el host de un usuario, necesita traducir un nombre de host en una dirección IP. La aplicación invocará al lado del cliente de DNS, especificando el nombre de host del que necesita la correspondiente traducción. (En muchos sistemas basados en UNIX, `gethostbyname()` es la llamada a función que una aplicación utiliza para llevar a cabo la traducción.) Entonces, la aplicación DNS en el host del usuario entra en funcionamiento, enviando un mensaje de consulta a la red. Todos los mensajes de consulta y de respuesta DNS se envían dentro de datagramas UDP al puerto 53. Transcurrido un cierto retardo, del orden de milisegundos a segundos, el servicio DNS del host del usuario recibe un mensaje de respuesta DNS que proporciona la traducción deseada, la cual se pasa entonces a la aplicación que lo ha invocado. Por tanto, desde la perspectiva de dicha aplicación que se ejecuta en el host del usuario, DNS es una caja negra que proporciona un servicio de traducción simple y directo. Pero, de hecho, la caja negra que implementa el servicio es compleja,



EN LA PRÁCTICA

DNS: FUNCIONES CRÍTICAS DE RED MEDIANTE EL PARADIGMA CLIENTE-SERVIDOR

Como HTTP, FTP y SMTP, el protocolo DNS es un protocolo de la capa de aplicación puesto que (1) se ejecuta entre sistemas terminales que están en comunicación utilizando el paradigma cliente-servidor y (2) se basa en un protocolo de transporte subyacente extremo a extremo para transferir los mensajes DNS entre los sistemas terminales en comunicación. Sin embargo, desde otro punto de vista, la función de DNS es bastante diferente a la de las aplicaciones web de transferencia de archivos y de correo electrónico. A diferencia de estas aplicaciones, DNS no es una aplicación con la que el usuario interactúe directamente; en su lugar, DNS lleva a cabo una de las funciones básicas en Internet: traducir los nombres de hosts en sus direcciones IP subyacentes para las aplicaciones de usuario y otras aplicaciones software de Internet. Hemos mencionado en la Sección 1.2 que gran parte de la complejidad de la arquitectura de Internet se encuentra en las “fronteras” de la red. DNS, que implementa el importante proceso de traducción de nombres en direcciones, utilizando clientes y servidores ubicados en la frontera de la red, es otro ejemplo más de dicha filosofía de diseño.

constando de un gran número de servidores DNS distribuidos por todo el mundo, así como de un protocolo de la capa de aplicación que especifica cómo los servidores DNS y los hosts que realizan las consultas se comunican.

Un diseño simple de DNS podría ser un servidor DNS que contuviera todas las correspondencias entre nombres y direcciones. En este diseño centralizado, los clientes simplemente dirigirían todas las consultas a un mismo servidor DNS y este respondería directamente a las consultas de los clientes. Aunque la simplicidad de este diseño es atractiva, es inapropiado para la red Internet de hoy día a causa de la enorme (y creciente) cantidad de hosts. Entre los problemas con un diseño centralizado podemos citar los siguientes:

- **Un único punto de fallo.** Si el servidor DNS falla, entonces ¡también falla toda la red Internet!
- **Volumen de tráfico.** Un único servidor DNS tendría que gestionar todas las consultas DNS (de todas las solicitudes HTTP y mensajes de correo electrónico generados en cientos de millones de hosts).
- **Base de datos centralizada distante.** Un único servidor DNS no puede “estar cerca” de todos los clientes que efectúan consultas. Si colocamos ese único servidor DNS en la ciudad de Nueva York, entonces todas las consultas desde Australia deberían viajar hasta el otro extremo del globo, quizás a través de enlaces lentos y congestionados. Esto podría llevar por tanto a retardos significativos.
- **Mantenimiento.** Ese único servidor DNS tendría que mantener registros de todos los hosts de Internet. No solo sería una enorme base de datos centralizada, sino que tendría que ser actualizada con frecuencia con el fin de incluir todos los hosts nuevos.

En resumen, una base de datos centralizada almacenada en un único servidor DNS simplemente *no podría escalarse*. En consecuencia, el sistema DNS utiliza un diseño distribuido. De hecho, DNS es un estupendo ejemplo de cómo se puede implementar una base de datos distribuida en Internet.

Una base de datos jerárquica y distribuida

Para abordar el problema del escalado, DNS utiliza un gran número de servidores, organizados de forma jerárquica y distribuidos alrededor de todo el mundo. Ningún servidor DNS dispone de todas las correspondencias de todos los hosts de Internet. En su lugar, las correspondencias están repartidas por los servidores DNS. En una primera aproximación, podemos decir que existen tres clases de

servidores DNS: los servidores DNS raíz, los servidores DNS de dominio de nivel superior (TLD, *Top-Level Domain*) y los servidores DNS autoritativos, organizados en una jerarquía como la mostrada en la Figura 2.17. Con el fin de comprender cómo estas tres clases de servidores interactúan, suponga que un cliente DNS desea determinar la dirección IP correspondiente al nombre de host `www.amazon.com`. En una primera aproximación tienen lugar los siguientes sucesos: primero, el cliente contacta con uno de los servidores raíz, el cual devuelve las direcciones IP para los servidores TLD del dominio de nivel superior `.com`. A continuación, el cliente contacta con uno de estos servidores TLD, que devuelve la dirección IP de un servidor autoritativo para `amazon.com`. Por último, el cliente contacta con uno de los servidores autoritativos de `amazon.com`, el cual devuelve la dirección IP correspondiente al nombre de host `www.amazon.com`. Enseguida examinaremos este proceso de búsqueda DNS con más detalle. Pero en primer lugar, echemos un vistazo a estas tres clases de servidores DNS:

- **Servidores DNS raíz.** Existen unos 400 servidores de nombres raíz distribuidos por todo el mundo. La Figura 2.18 muestra los países que disponen de estos servidores, con los países que tienen más de diez servidores sombreados en gris oscuro. Trece organizaciones diferentes gestionan estos servidores de nombres raíz. Puede encontrar una lista completa de los servidores de nombres raíz, junto con las organizaciones que los gestionan y sus direcciones IP en [Root Servers 2016]. Los servidores de nombres raíz proporcionan las direcciones IP de los servidores TLD.
- **Servidores de dominio de nivel superior (TLD).** Para todos los dominios de nivel superior, como son `.com`, `.org`, `.net`, `.edu` y `.gov`, y todos los dominios de nivel superior correspondientes a los distintos países, como por ejemplo, `.uk`, `.fr`, `.ca` y `.jp`, existe un servidor TLD (o agrupación de servidores). La empresa Verisign Global Registry Services mantiene los servidores TLD para el dominio de nivel superior `.com` y la empresa Educause mantiene los servidores TLD para el dominio de nivel superior `.edu`. La infraestructura de red que da soporte a un TLD puede ser muy grande y compleja; consulte [Osterweil 2012] para ver una introducción de la red de Verisign. Consulte [TLD list 2016] para ver una lista de todos los dominios de nivel superior. Los servidores TLD proporcionan las direcciones IP para los servidores DNS autoritativos.
- **Servidores DNS autoritativos.** Todas las organizaciones que tienen hosts accesibles públicamente (como son los servidores web y los servidores de correo) a través de Internet deben proporcionar registros DNS accesibles públicamente que establezcan la correspondencia entre los nombres de dichos hosts y sus direcciones IP. Un servidor DNS autoritativo de una organización alberga estos registros DNS. Una organización puede elegir implementar su propio servidor DNS autoritativo para almacenar estos registros; alternativamente, la organización puede pagar por tener esos registros almacenados en un servidor DNS autoritativo de algún proveedor de servicios. La mayoría de las universidades y de las empresas de gran tamaño implementan y mantienen sus propios servidores DNS autoritativos principal y secundario (*backup*).

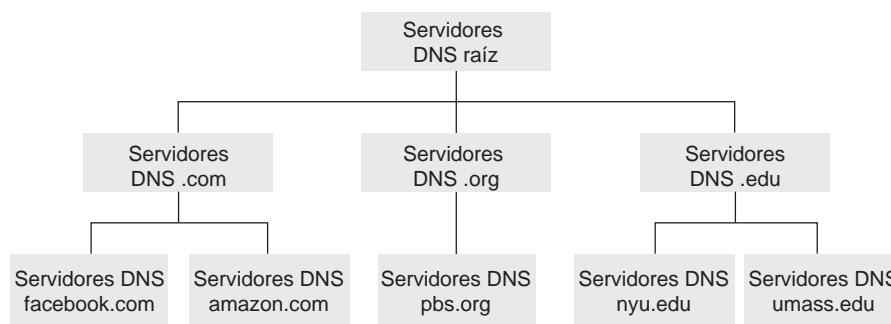


Figura 2.17 ♦ Parte de la jerarquía de los servidores DNS.



Figura 2.18 ♦ Servidores DNS raíz en 2016.

Todos los servidores DNS raíz, TLD y autoritativos pertenecen a la jerarquía de servidores DNS, como se muestra en la Figura 2.17. Existe otro tipo importante de servidor DNS conocido como **servidor DNS local**. Un servidor DNS local no pertenece estrictamente a la jerarquía de servidores, pero no obstante es importante dentro de la arquitectura DNS. Cada ISP (como por ejemplo un ISP residencial o un ISP institucional) tiene un servidor DNS local (también denominado servidor de nombres predeterminado). Cuando un host se conecta a un ISP, este proporciona al host las direcciones IP de uno o más de sus servidores DNS locales (normalmente a través de DHCP, lo que veremos en el Capítulo 4). Usted puede determinar fácilmente la dirección IP de su servidor DNS local accediendo a las ventanas de estado de la red en Windows o UNIX. Generalmente, un servidor DNS local de un host se encuentra “cerca” de ese host. En el caso de un ISP institucional, el servidor DNS local puede encontrarse en la misma LAN que el host; en el caso de un ISP residencial, estará separado del host no más de unos pocos routers. Cuando un host realiza una consulta DNS, esta se envía al servidor DNS local, que actúa como proxy, reenviando la consulta a la jerarquía de servidores DNS, como veremos más detalladamente a continuación.

Veamos un ejemplo sencillo. Suponga que el host `cse.nyu.edu` desea conocer la dirección de `gaia.cs.umass.edu`. Suponga también que el servidor DNS local de la Universidad de Nueva York para `cse.nyu.edu` se denomina `dns.nyu.edu` y que un servidor DNS autoritativo para `gaia.cs.umass.edu` se llama `dns.umass.edu`. Como se muestra en la Figura 2.19, el host `cse.nyu.edu` envía en primer lugar un mensaje de consulta DNS a su servidor DNS local, `dns.nyu.edu`. El mensaje de consulta contiene el nombre de host que debe ser traducido, `gaia.cs.umass.edu`. El servidor DNS local reenvía la consulta a un servidor DNS raíz, el cual toma el sufijo `edu` y devuelve al servidor DNS local una lista de las direcciones IP de los servidores TLD responsables del dominio `edu`. El servidor DNS local reenvía a continuación el mensaje de consulta a uno de estos servidores TLD. El servidor TLD toma nota del sufijo `umass.edu` y responde con la dirección IP del servidor DNS autoritativo correspondiente a la Universidad de Massachusetts, es decir, `dns.umass.edu`. Por último, el servidor DNS local reenvía la consulta directamente a `dns.umass.edu`, que responde con la dirección IP de `gaia.cs.umass.edu`. Observe que en este ejemplo, para obtener la dirección correspondiente a un nombre de host, se han enviado ocho mensajes DNS: ¡cuatro mensajes de consulta y cuatro mensajes de respuesta! Pronto veremos cómo el almacenamiento en caché DNS reduce este tráfico de consultas.

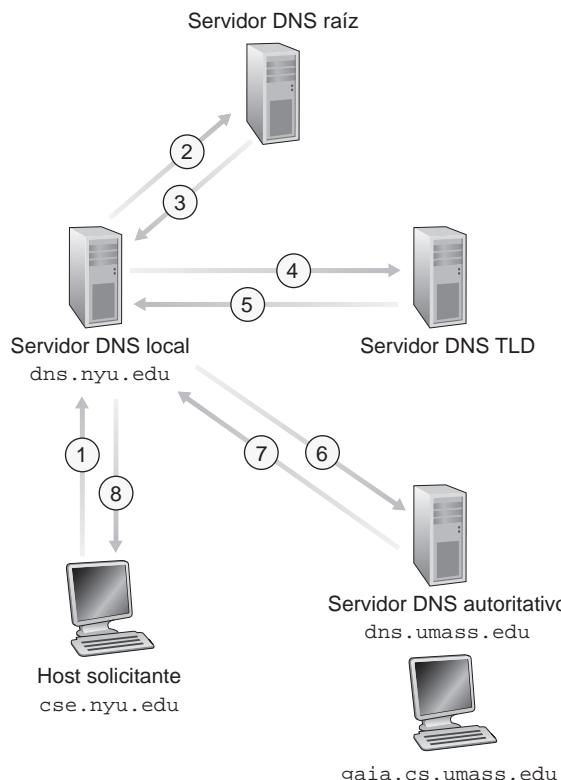


Figura 2.19 ♦ Interacción de los distintos servidores DNS.

En el ejemplo anterior suponímos que el servidor TLD conoce el servidor DNS autoritativo correspondiente al nombre de host. En general esto no es así. En lugar de ello, el servidor TLD puede saber únicamente de un servidor DNS intermedio, el cual a su vez sabe cuál es el servidor DNS autoritativo para el nombre de host. Por ejemplo, suponga de nuevo que la Universidad de Massachusetts tiene un servidor DNS para la universidad, denominado `dns.umass.edu`. Suponga también que cada uno de los departamentos de la Universidad de Massachusetts tiene su propio servidor DNS y que cada servidor DNS departamental es un servidor autoritativo para todos los hosts del departamento. En este caso, cuando el servidor DNS intermedio `dns.umass.edu` recibe una consulta para un host cuyo nombre termina en `cs.umass.edu`, devuelve a `dns.nyu.edu` la dirección IP de `dns.cs.umass.edu`, que es autoritativo para todos los nombres de host que terminan con `cs.umass.edu`. El servidor DNS local `dns.nyu.edu` envía entonces la consulta al servidor DNS autoritativo, que devuelve la correspondencia deseada al servidor DNS local, el cual a su vez devuelve la correspondencia al host que ha hecho la solicitud. En este caso, ¡se han enviado un total de 10 mensajes!

El ejemplo mostrado en la Figura 2.19 utiliza tanto **consultas recursivas** como **consultas iterativas**. La consulta enviada desde `cse.nyu.edu` a `dns.nyu.edu` es una consulta recursiva, ya que la consulta solicita a `dns.nyu.edu` que obtenga por sí mismo la correspondencia. Pero las tres consultas siguientes son iterativas puesto que todas las respuestas son devueltas directamente a `dns.nyu.edu`. En teoría, cualquier consulta DNS puede ser iterativa o recursiva. Por ejemplo, la Figura 2.20 muestra una cadena de consultas DNS para las que todas las consultas son recursivas. **[En la práctica, las consultas normalmente siguen el patrón mostrado en la Figura 2.19; la consulta procedente del host que hace la solicitud al servidor DNS local es recursiva y las restantes consultas son iterativas.]**

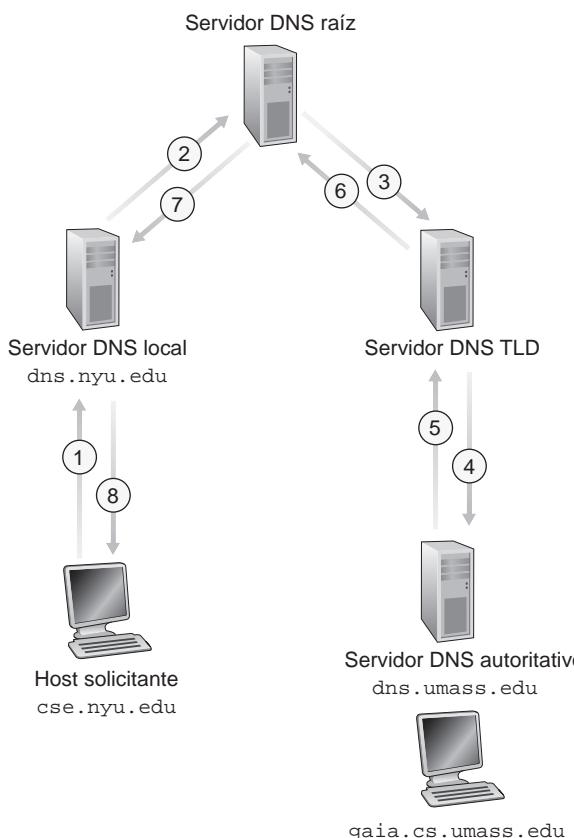


Figura 2.20 ♦ Consultas recursivas en DNS.

Almacenamiento en caché DNS

Hasta el momento hemos ignorado el **almacenamiento en caché DNS**, una funcionalidad extremadamente importante del sistema DNS. En realidad, DNS explota exhaustivamente el almacenamiento en caché para mejorar el comportamiento de los retardos y reducir el número de mensajes DNS que van de un lado a otro por Internet. La idea que se esconde detrás del almacenamiento en caché DNS es muy simple. En una cadena de consultas, cuando un servidor DNS recibe una respuesta DNS (que contiene, por ejemplo, una correspondencia entre un nombre de host y una dirección IP), puede almacenar esta información en su memoria local. Por ejemplo, en la Figura 2.19, cada vez que el servidor DNS local dns.nyu.edu recibe una respuesta de algún servidor DNS, puede almacenar en caché cualquier información contenida en esa respuesta. Si una relación nombre de host/dirección IP se almacena en la caché de un servidor DNS y llegan otras consultas a ese mismo servidor DNS preguntando por el mismo nombre de host, el servidor DNS puede proporcionar la dirección IP deseada, incluso aunque no sea autoritativo para el nombre de host. Dado que los hosts y las correspondencias entre nombres de host y direcciones IP no son permanentes, los servidores DNS descartan la información almacenada en caché pasado un cierto periodo de tiempo (normalmente, unos dos días).

Por ejemplo, suponga que un host `apricot.nyu.edu` consulta a `dns.nyu.edu` solicitándole la dirección IP correspondiente al nombre de host `cnn.com`. Suponga también que unas pocas horas más tarde, otro host de la Universidad de Nueva York, digamos `kiwi.nyu.edu`, también hace una consulta a `dns.nyu.edu` especificando el mismo nombre de host. Gracias al almacenamiento en caché, el servidor DNS local podrá devolver de forma inmediata la dirección IP de `cnn.com` al segundo host que la ha solicitado sin tener que consultar a ningún

otro servidor DNS. Un servidor DNS local también puede almacenar en caché las direcciones IP de los servidores TLD, permitiendo así a los servidores DNS locales saltarse a los servidores DNS raíz en una cadena de consultas. De hecho, gracias al almacenamiento en caché, todas las consultas DNS, excepto una pequeña fracción de las mismas, se saltan a los servidores raíz.

2.4.3 Registros y mensajes DNS

Los servidores DNS que implementan conjuntamente la base de datos distribuida DNS **almacenan los registros de recursos (RR)**, incluyendo los que proporcionan las correspondencias entre nombre de host y dirección IP. Cada mensaje de respuesta DNS transporta uno o más registros de recursos. En esta y en la subsección siguiente, proporcionamos una breve introducción a los recursos y mensajes DNS; puede encontrar información detallada en [Abitz 1993] o en los RFC sobre DNS [RFC 1034; RFC 1035].

Un registro de recurso está formado por los siguientes cuatro campos:

(Nombre, Valor, Tipo, TTL)

TTL es el tiempo de vida del registro de recurso; determina cuándo un recurso debería ser eliminado de una caché. En los registros de ejemplo dados a continuación, hemos ignorado el campo TTL. El significado de Nombre y Valor depende del campo Tipo:

- Si Tipo=A, entonces Nombre es un nombre de host y Valor es la dirección IP correspondiente a dicho nombre. Por tanto, un registro de tipo A proporciona la correspondencia estándar nombre de host-dirección IP. Por ejemplo, (relay1.bar.foo.com, 145.37.93.126, A) es un registro de tipo A.
- Si Tipo = NS, entonces Nombre es un dominio (como foo.com) y Valor es el nombre de host de un servidor DNS autoritativo que sabe cómo obtener las direcciones IP de los hosts del dominio. Este registro se utiliza para enrutar las consultas DNS a lo largo de la cadena de consultas. Por ejemplo, (foo.com, dns.foo.com, NS) es un registro de tipo NS.
- Si Tipo = CNAME, entonces Valor es un nombre de host canónico correspondiente al alias especificado por Nombre. Este registro puede proporcionar a los hosts que hacen consultas el nombre canónico correspondiente a un nombre de host. Por ejemplo, (foo.com, relay1.bar.foo.com) es un registro CNAME.
- Si Tipo = MX, entonces Valor es el nombre canónico de un servidor de correo que tiene un alias dado por Nombre. Por ejemplo, (foo.com, mail.bar.foo.com, MX) es un registro MX. Los registros MX permiten a los nombres de host de los servidores de correo tener alias simples. Observe que utilizando el registro MX, una empresa puede tener el mismo alias para su servidor de correo y para uno de sus otros servidores (como por ejemplo, el servidor web). Para obtener el nombre canónico del servidor de correo, un cliente DNS consultaría un registro MX y para conocer el nombre canónico del otro servidor, consultaría el registro CNAME.

Si un servidor DNS es autoritativo para un determinado nombre de host, entonces el servidor DNS contendrá un registro de tipo A para el nombre de host. (Incluso aunque el servidor DNS no sea autoritativo, puede contener un registro de tipo A en su caché.) Si un servidor no es autoritativo para un nombre de host, entonces el servidor contendrá un registro de tipo NS para el dominio que incluye el nombre de host; también contendrá un registro de tipo A que proporcione la dirección IP del servidor DNS en el campo Valor del registro NS. Por ejemplo, suponga un servidor TLD edu que no es autoritativo para el host gaia.cs.umass.edu. Por tanto, este servidor contendrá un registro para un dominio que incluye el host gaia.cs.umass.edu, por ejemplo, (umass.edu, dns.umass.edu, NS). El servidor TLD edu también contendrá un registro de tipo A, que establece la correspondencia entre el servidor DNS dns.umass.edu y una dirección IP, como en (dns.umass.edu, 128.119.40.111, A).

Mensajes DNS

Anteriormente en esta sección hemos hecho referencia a los mensajes de respuesta y consultas DNS. Únicamente existen estas dos clases de mensajes DNS. Además, tanto los mensajes de respuesta como de consulta utilizan el mismo formato, que se muestra en la Figura 2.21. La semántica en los distintos campos de un mensaje DNS es la siguiente:

- Los primeros 12 bytes constituyen la *sección de cabecera*, la cual contiene una serie de campos. El primero de estos campos es un número de 16 bits que identifica la consulta. Este identificador se copia en el mensaje de respuesta a la consulta, lo que permite al cliente establecer las correspondencias correctas entre las respuestas recibidas y las consultas enviadas. En el campo Indicadores se incluyen una serie de indicadores. Un indicador consulta/respondida de 1 bit informa de si el mensaje es una consulta (0) o una respuesta (1). Un indicador autoritativo de 1 bit se activa en un mensaje de respuesta cuando un servidor DNS es un servidor autoritativo para un nombre solicitado. El indicador recursión-deseada, también de 1 bit, se activa cuando un cliente (host o servidor DNS) desea que el servidor DNS realice una recursión cuando no disponga del registro. En un mensaje de respuesta, el campo de recursión-disponible de 1 bit se activa si el servidor DNS soporta la recursión. En la cabecera también se incluyen cuatro campos “número de”, que indican el número de apariciones de los cuatro tipos de secciones de datos que siguen a la cabecera.
- La *sección cuestiones* contiene información acerca de la consulta que se va a realizar. Esta sección incluye (1) un campo de nombre que contiene el nombre que se va a consultar y (2) un campo de tipo que especifica el tipo de cuestión que se plantea acerca del nombre; por ejemplo, la dirección del host asociada con un nombre (tipo A) o el servidor de correo para un nombre (tipo MX).
- En una respuesta de un servidor DNS, la *sección respuestas* contiene los registros del recurso para el nombre que fue consultado originalmente. Recuerde que en cada registro de recurso existe un parámetro *Tipo* (por ejemplo, A, NS, CNAME y MX), un parámetro *Valor* y el parámetro *TTL*. Una respuesta puede devolver varios registros de recursos, ya que un nombre de host puede tener asociadas varias direcciones IP (por ejemplo, para los servidores web replicados, como hemos visto anteriormente en esta sección).
- La *sección autoridad* contiene registros de otros servidores autoritativos.
- La *sección información adicional* contiene otros registros útiles. Por ejemplo, el campo de respuesta en un mensaje de respuesta a una consulta MX contiene un registro de recurso que

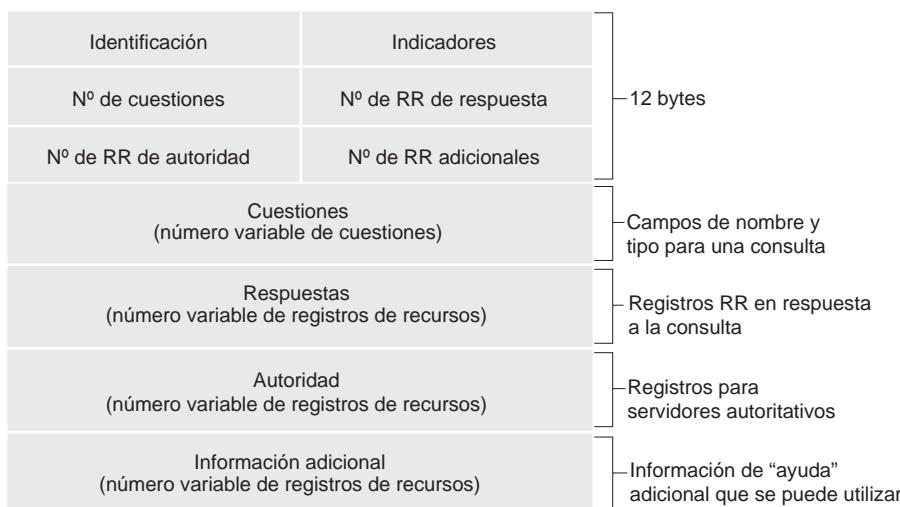


Figura 2.21 ♦ Formato de los mensajes DNS.

proporciona el nombre de host canónico de un servidor de correo. Esta sección de información adicional contiene un registro de tipo A que proporciona la dirección IP para el nombre de host canónico del servidor de correo.

¿Le gustaría enviar un mensaje de consulta DNS directamente desde el host en el que está trabajando a algún servidor DNS? Esto podría hacerse fácilmente con el **programa nslookup**, que está disponible en la mayoría de las plataformas Windows y UNIX. Por ejemplo, en un host Windows basta con abrir la aplicación Símbolo del sistema (*prompt*) e invocar al programa nslookup escribiendo simplemente “nslookup”. A continuación, podrá enviar una consulta DNS a cualquier servidor DNS (raíz, TLD o autoritativo). Después de recibir el mensaje de respuesta del servidor DNS, nslookup mostrará los registros incluidos en la respuesta (en un formato legible para las personas). Como alternativa a la ejecución de nslookup desde su propio host, puede visitar uno de los muchos sitios web que permiten utilizar nslookup de forma remota (basta con escribir “nslookup” en un motor de búsqueda para acceder a uno de estos sitios). La práctica de laboratorio DNS Wireshark disponible al final de este capítulo le permitirá explorar DNS en detalle.

Inserción de registros en la base de datos DNS

La exposición anterior se ha centrado en cómo se recuperan los registros de la base de datos DNS. Es posible que ahora se esté preguntando cómo se introducen los registros en dicha base de datos. Veamos cómo se hace esto en el contexto de un ejemplo concreto. Suponga que hemos creado una nueva empresa llamada Network Utopia. Lo primero que seguramente deseará hacer es registrar el nombre de dominio `networkutopia.com` en un registro. Un **registrador** es una entidad comercial que verifica la unicidad del nombre de dominio, lo añade a la base de datos DNS (como veremos más adelante) y percibe unas pequeñas tasas por sus servicios. Antes de 1999, un único registrador, Network Solutions, tenía el monopolio sobre el registro de nombres para los dominios com, net y org. Pero actualmente, existen muchas entidades registradoras que compiten por los clientes. La ICANN (*Internet Corporation for Assigned Names and Numbers*, Corporación de Internet para nombres y números asignados) acredita a las distintas entidades registradoras. En la dirección `http://www.internic.net` puede encontrar una lista completa de estas entidades.

Al registrar el nombre de dominio `networkutopia.com` en alguna entidad registradora, también tendrá que proporcionarle los nombres y direcciones IP de sus servidores DNS autoritativos principal y secundario. Suponga que en nuestro ejemplo estos datos son: `dns1.networkutopia.com`, `dns2.networkutopia.com`, `212.212.212.1` y `212.212.212.2`. Para cada uno de estos dos servidores DNS autoritativos, el registrador se asegura de que se introduzca un registro de tipo NS y un registro de tipo A en los servidores TLD com. Específicamente, para el servidor autoritativo principal de `networkutopia.com`, la entidad registradora deberá insertar los siguientes dos registros de recursos en el sistema DNS:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

También tendrá que asegurarse de que el registro de recurso de tipo A para su servidor web `www.networkutopia.com` y el registro de recurso de tipo MX para su servidor de correo `mail.networkutopia.com` se han introducido en sus servidores DNS autoritativos. (Hasta hace poco, los contenidos de los servidores DNS se configuraban estáticamente; por ejemplo, a partir de un archivo de configuración creado por un administrador del sistema. Sin embargo, recientemente se ha añadido una opción de actualización (UPDATE) al protocolo DNS que permite añadir o borrar dinámicamente los datos de la base de datos mediante mensajes DNS. En los documentos [RFC 2136] y [RFC 3007] se especifican las actualizaciones dinámicas de DNS.)

Una vez que haya completado estos pasos, los usuarios podrán visitar su sitio web y enviar correos electrónicos a los empleados de su empresa. Vamos a terminar esta exposición sobre DNS verificando que esta afirmación es cierta. Esta verificación también le ayudará a consolidar lo que

SEGURIDAD

VULNERABILIDADES DNS

Hemos visto que DNS es un componente fundamental de la infraestructura de Internet y que muchos servicios importantes, entre los que incluyen las aplicaciones web y de correo electrónico, no podrían funcionar sin él. Por tanto, lo natural es preguntarse: ¿Cómo puede ser atacado DNS? ¿Es DNS un blanco fácil, que espera a ser puesto fuera de servicio, arrastrando con él a la mayoría de las aplicaciones de Internet?

El primer tipo de ataque que nos viene a la mente es un ataque DDoS por de ancho de banda (véase la Sección 1.6) contra los servidores DNS. Por ejemplo, un atacante podría intentar enviar a cada uno de los servidores DNS raíz una gran cantidad de paquetes, tantos que la mayor parte de las consultas DNS legítimas nunca fueran contestadas. Un ataque DDoS a gran escala contra servidores DNS raíz tuvo lugar realmente el 21 de octubre de 2002. En ese ataque, los atacantes utilizaron una botnet para enviar enormes cargas de mensajes ping ICMP a las direcciones IP de 13 servidores DNS raíz. (Los mensajes ICMP se estudian en la Sección 5.6. Por el momento, nos basta con saber que los paquetes ICMP son tipos especiales de datagramas IP.) Afortunadamente, este ataque a gran escala causó unos daños mínimos, sin tener apenas impacto sobre la experiencia en Internet de los usuarios. Los atacantes dirigieron con éxito gran cantidad de paquetes a los servidores raíz, pero muchos de estos servidores estaban protegidos mediante mecanismos de filtrado de paquetes configurados para bloquear siempre todos los mensajes ping ICMP dirigidos a los mismos. Estos servidores protegidos estaban resguardados y funcionaron normalmente. Además, la mayoría de los servidores DNS locales tenían almacenadas en caché las direcciones IP de los servidores de dominio de nivel superior, permitiendo el procesamiento de consultas ignorando normalmente a los servidores DNS raíz.

Un ataque DDoS potencialmente más efectivo contra servidores DNS consistiría en enviar una gran cantidad de consultas DNS a los servidores de dominio de alto nivel; por ejemplo, a todos aquellos servidores que administren el dominio .com. Sería bastante complicado filtrar las consultas DNS dirigidas a servidores DNS; y los servidores de dominio de nivel superior no pueden ser puenteados tan fácilmente como los servidores raíz. Pero la severidad de un ataque así podría ser parcialmente mitigada por el almacenamiento en caché de los servidores DNS locales.

El sistema DNS también podría ser atacado, en teoría, de otras maneras. En un ataque por intermediación (*man-in-the-middle*), el atacante intercepta las consultas procedentes de los hosts y devuelve respuestas falsas. En el ataque por envenenamiento DNS, el atacante envía respuestas falsas a un servidor DNS, engañándole y haciendo que almacene en su caché registros falsos. Cualquiera de estos ataques podría utilizarse, por ejemplo, para redirigir a un usuario web inadvertido al sitio web del atacante. Sin embargo, estos ataques son difíciles de implementar, ya que requieren interceptar los paquetes o engañar a los servidores [Skoudis 2006].

En resumen, DNS ha demostrado ser sorprendentemente robusto frente a los ataques. Hasta el momento, no ha habido ningún ataque que haya conseguido interrumpir con éxito el servicio DNS.

ha aprendido sobre DNS. Suponga que Alicia se encuentra en Australia y desea ver la página web www.networkutopia.com. Como hemos explicado anteriormente, su host enviará en primer lugar una consulta DNS a su servidor DNS local, el cual a su vez se pondrá en contacto con un servidor TLD .com. (El servidor DNS local también tendrá que comunicarse con un servidor DNS raíz si no tiene en caché la dirección de un servidor TLD .com.) El servidor TLD contendrá los registros de recursos de tipo NS y de tipo A enumerados anteriormente, ya que la entidad registradora los habrá almacenado en todos los servidores TLD .com. El servidor TLD .com envía entonces una respuesta al servidor DNS local de Alicia, conteniendo dicha respuesta los dos registros de recursos. A continuación, el servidor DNS local transmite una consulta DNS a 212.212.212.1, solicitando

el registro de tipo A correspondiente a www.networkkutopia.com. Este registro proporciona la dirección IP del servidor web deseado, por ejemplo, 212.212.71.4, que el servidor DNS local pasa al host de Alicia. En este momento, el navegador de Alicia puede iniciar una conexión TCP con el host 212.212.71.4 y enviar una solicitud HTTP a través de la misma. Como ha podido ver, son muchas las cosas que suceden entre bastidores cuando navegamos por la Web.

2.5 Distribución de archivos P2P

Las aplicaciones descritas en este capítulo hasta el momento, incluyendo las aplicaciones web, de correo electrónico y DNS, emplean todas ellas arquitecturas cliente-servidor que dependen en gran medida de que exista una infraestructura de servidores siempre activos. Recuerde de la Sección 2.1.1 que con una arquitectura P2P no se depende más que mínimamente (o no se depende en absoluto) de que exista esa infraestructura de servidores siempre activos. En su lugar, una serie de parejas de hosts conectados de forma intermitente, denominados pares o *peers*, se comunican directamente entre sí. Los pares no son propiedad de un proveedor de servicios, sino que son computadoras de escritorio o portátiles controlados por los usuarios.

En esta sección vamos a ver una aplicación P2P muy corriente, la distribución de un archivo de gran tamaño desde un único servidor a muchos otros hosts (denominados pares). El archivo podría ser una nueva versión del sistema operativo Linux, un parche software para una aplicación o un sistema operativo existentes, un archivo de audio MP3 o un archivo de vídeo MPEG. En la distribución de archivos cliente-servidor, el servidor debe enviar una copia del archivo a cada uno de los pares, provocando una enorme sobrecarga en el servidor y consumiendo una gran cantidad de su ancho de banda. En la distribución de archivos P2P, cada par puede redistribuir cualquier parte del archivo que ha recibido a cualesquiera otros pares, ayudando de este modo al servidor a llevar a cabo el proceso de distribución. En 2016, el protocolo de distribución de archivos P2P más popular es BitTorrent. Originalmente, fue desarrollado por Bram Cohen, pero ahora existen muchos clientes BitTorrent independientes distintos que cumplen con el protocolo BitTorrent, al igual que existen diversos navegadores web que cumplen el protocolo HTTP. En esta subsección examinaremos en primer lugar la característica de auto-escalabilidad de las arquitecturas P2P en el contexto de la distribución de archivos. Describiremos en detalle BitTorrent, destacando sus funcionalidades y características más importantes.

Escalabilidad de las arquitecturas P2P

Con el fin de comparar las arquitecturas cliente-servidor con las arquitecturas P2P y para ilustrar la auto-escalabilidad inherente de P2P, ahora vamos a considerar un modelo cuantitativo simple para la distribución de un archivo a un conjunto fijo de pares en ambos tipos de arquitectura. Como se muestra en la Figura 2.22, el servidor y los pares están conectados a Internet mediante enlaces de acceso. Sea u_s la velocidad de carga del enlace de acceso del servidor, u_i la velocidad de carga del enlace de acceso del par i y d_i la velocidad de descarga del enlace de acceso del par i . Sea F el tamaño en bits del archivo que se va a distribuir y N el número de pares que desean obtener una copia del archivo. El **tiempo de distribución** es el tiempo que tardan los N pares en obtener una copia del archivo. En el análisis del tiempo de distribución que proporcionamos a continuación, para ambas arquitecturas, cliente-servidor y P2P, hemos hecho una simplificación (pero generalmente precisa [Akella 2003]): suponer que el núcleo de Internet tiene el ancho de banda suficiente, lo que implica que todos los cuellos de botella se encuentran en el acceso a red. También hemos supuesto que el servidor y los clientes no están participando en ninguna otra aplicación de red, de modo que los anchos de banda para carga y descarga están dedicados completamente a distribuir el archivo.

En primer lugar, vamos a determinar el tiempo de distribución para la arquitectura cliente-servidor, el cual denotaremos como D_{cs} . En esta arquitectura, ninguno de los pares ayudan a distribuir el archivo. Tenemos que hacer las dos observaciones siguientes:

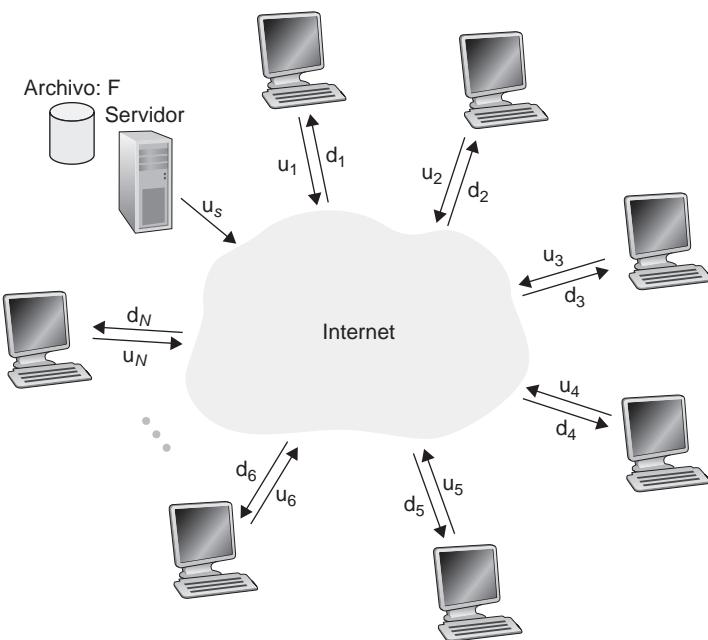


Figura 2.22 ♦ Problema de distribución de un archivo.

- El servidor debe transmitir una copia del archivo a cada uno de los N pares. Por tanto, el servidor tiene que transmitir NF bits. Puesto que la velocidad de carga del servidor es u_s , el tiempo para distribuir el archivo tiene que ser como mínimo NF/u_s .
- Sea d_{\min} la velocidad de descarga del par cuya velocidad de descarga sea menor; es decir, $d_{\min} = \min\{d_1, d_2, \dots, d_N\}$. El par con la menor velocidad de descarga no puede obtener los F bits del archivo en menos de F/d_{\min} segundos. Por tanto, el tiempo mínimo de distribución es, al menos igual a F/d_{\min} .

Teniendo en cuenta estas dos observaciones, se obtiene:

$$D_{cs} \geq \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{\min}} \right\}$$

Esto proporciona un límite inferior al tiempo de distribución mínimo para la arquitectura cliente-servidor. En los problemas de repaso se le pedirá que demuestre que el servidor puede planificar sus transmisiones de manera que el límite inferior sea alcanzado realmente. Por tanto, tomemos este límite inferior como el tiempo de distribución real, es decir,

$$D_{cs} = \max \left\{ \frac{NF}{u_s}, \frac{F}{d_{\min}} \right\} \quad (2.1)$$

A partir de la Ecuación 2.1, se ve que para N lo suficientemente grande, el tiempo de distribución en una arquitectura cliente-servidor está dada por NF/u_s . Por tanto, el tiempo de distribución aumenta linealmente con el número de pares N . Así, por ejemplo, si el número de pares se multiplica por mil en una semana, pasando de mil a un millón, el tiempo necesario para distribuir el archivo a todos los pares se verá multiplicado por 1.000.

Hagamos ahora un análisis similar para la arquitectura P2P, donde cada par puede ayudar al servidor a distribuir el archivo. En particular, cuando un par recibe datos del archivo, puede utilizar su propia capacidad de carga para redistribuir los datos a otros pares. Calcular el tiempo

de distribución para la arquitectura P2P es algo más complicado que para la arquitectura cliente-servidor, ya que el tiempo de distribución depende de cómo cada par implicado distribuya partes del archivo a los demás pares. No obstante, puede obtenerse una expresión simple que permite calcular el tiempo mínimo de distribución [Kumar 2006]. Para este fin, debemos tener en cuenta las siguientes observaciones:

- Al comenzar el proceso de distribución, el archivo solo lo tiene el servidor. Para que este archivo llegue a la comunidad de pares, el servidor tiene que enviar cada bit del archivo al menos una vez por su enlace de acceso. Por tanto, el tiempo mínimo de distribución es, como mínimo, F/u_s . (A diferencia de lo que ocurre en el esquema cliente-servidor, un bit enviado por el servidor puede no tener que ser enviado de nuevo por el mismo, ya que los pares pueden redistribuirlo entre ellos.)
- Al igual que en la arquitectura cliente-servidor, el par con la menor velocidad de descarga no puede obtener los F bits del archivo en menos de F/d_{\min} segundos. Por tanto, el tiempo mínimo de distribución es al menos igual a F/d_{\min} .
- Por último, observe que la capacidad total de carga del sistema como un todo es igual a la velocidad de carga del servidor más las velocidades de carga de cada par, es decir, $u_{total} = u_s + u_1 + \dots + u_N$. El sistema tiene que suministrar (cargar) F bits en cada uno de los N peers, suministrando en total NF bits. Esto no se puede hacer a una velocidad mayor que u_{total} ; por tanto, el tiempo mínimo de distribución es también mayor o igual que $NF/(u_s + u_1 + \dots + u_N)$.

Teniendo en cuenta estas tres observaciones, obtenemos el tiempo mínimo de distribución para la arquitectura P2P, D_{P2P}

$$D_{P2P} \geq \max \left\{ \frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right\} \quad (2.2)$$

La Ecuación 2.2 proporciona un límite inferior para el tiempo mínimo de distribución en una arquitectura P2P. Si suponemos que cada peer puede redistribuir un bit tan pronto como lo recibe, entonces existe un esquema de redistribución que permite alcanzar este límite inferior [Kumar 2006]. (Demostraremos un caso especial de este resultado en los problemas de repaso.) En realidad, cuando se redistribuyen fragmentos del archivo en lugar de bits individuales, la Ecuación 2.2 sirve como una buena aproximación del tiempo mínimo de distribución real. Por tanto, vamos a tomar el límite inferior dado por la Ecuación 2.2 como el tiempo mínimo de distribución real, es decir,

$$D_{P2P} = \max \left\{ \frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right\} \quad (2.3)$$

La Figura 2.23 compara el tiempo mínimo de distribución de las arquitecturas cliente-servidor y P2P, suponiendo que todos los pares tienen la misma velocidad de carga u . En la figura, hemos establecido que $F/u = 1$ hora, $u_s = 10u$ y $d_{\min} \geq u_s$. Por tanto, un par puede transmitir el archivo completo en una hora, la velocidad de transmisión del servidor es 10 veces la velocidad de carga del par y (para simplificar) las velocidades de descarga de los pares son lo suficientemente grandes como para no tener ningún efecto. A partir de la Figura 2.23, podemos ver que para la arquitectura cliente-servidor el tiempo de distribución aumenta linealmente y sin límite a medida que el número de pares aumenta. Sin embargo, en una arquitectura P2P, el tiempo mínimo de distribución no solo siempre es menor que el tiempo de distribución en la arquitectura cliente-servidor; también es menor que una hora para *cualquier* número N de pares. Por tanto, las aplicaciones que emplean arquitectura P2P pueden auto-escalarse. Esta escalabilidad es una consecuencia directa de que los pares actúan a la vez como redistribuidores y consumidores de bits.

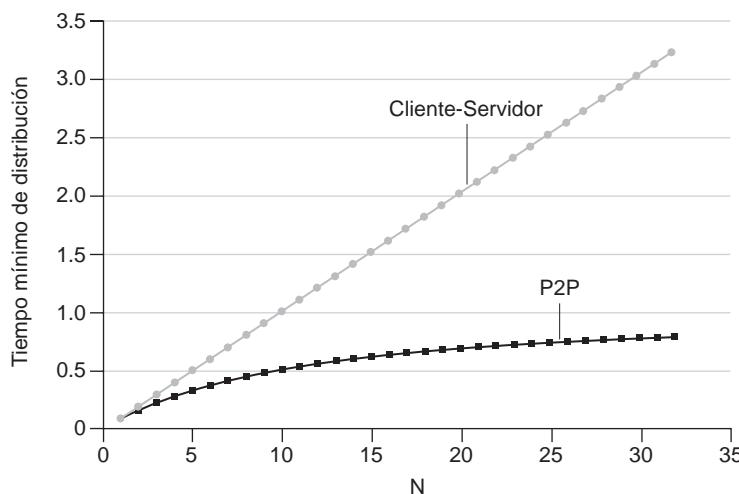


Figura 2.23 ♦ Tiempo de distribución para las arquitecturas P2P y cliente-servidor.

BitTorrent

BitTorrent es un popular protocolo P2P para la distribución de archivos [Chao 2011]. En la jerga de BitTorrent, la colección de todos los pares que participan en la distribución de un determinado archivo se conoce como *torrent* (torrente). Los peers de un torrente descargan *fragmentos* del mismo tamaño del archivo de uno a otro, siendo el tamaño típico de un fragmento de 256 KBytes. Cuando un par se une por primera vez a un torrente, no tiene fragmentos del archivo. A lo largo del tiempo va acumulando cada vez más fragmentos. A la vez que descarga fragmentos, actualiza fragmentos en otros pares. Una vez que un par ha adquirido el archivo completo, puede (egoístamente) abandonar el torrente, o (de forma altruista) permanecer en el mismo y continuar suministrando fragmentos a otros pares. Además, cualquier par puede abandonar el torrente en cualquier instante con solo un subconjunto de fragmentos, y volver a unirse más tarde.

Veamos ahora más de cerca cómo opera BitTorrent. Puesto que BitTorrent es un sistema y protocolo bastante complejo, sólo vamos a describir sus mecanismos más importantes, vamos a dejar al margen algunos detalles con el fin de poder ver claramente cómo funciona. Cada torrente tiene un nodo de infraestructura denominado *tracker* (rastreador). Cuando un par se une a un torrente, se registra mediante el *tracker* y, periódicamente, informa de que todavía se encuentra en el torrente. De esta manera, el *tracker* sigue la pista a los pares que están participando en el torrente. Un determinado torrente puede tener en un instante dado un número de pares participantes tan bajo como diez o tan alto como mil.

Como se muestra en la Figura 2.24, cuando un nuevo par, Alicia, se une al torrente, el *tracker* selecciona aleatoriamente un subconjunto de pares (digamos por ejemplo 50, con el fin de concretar) del conjunto de peers participantes y envía las direcciones IP de estos 50 peers a Alicia. Teniendo en su poder esta lista de pares, Alicia intenta establecer conexiones TCP concurrentes con todos los pares incluidos en dicha lista. Denominaremos a todos los pares con los que Alicia consigue establecer con éxito una conexión TCP “pares vecinos”. (En la Figura 2.24 vemos que Alicia solo tiene tres pares vecinos. Normalmente, podría tener muchos más.) A medida que pasa el tiempo, algunos de estos pares pueden abandonar la conexión y otros (aparte de los 50 iniciales) pueden intentar establecer conexiones TCP con Alicia. Por tanto, los pares vecinos de un determinado par irán variando con el tiempo.

En un determinado instante de tiempo, cada par tendrá un subconjunto de fragmentos del archivo, disponiendo los distintos pares de subconjuntos diferentes. Periódicamente, Alicia preguntará a cada uno de sus vecinos (a través de las conexiones TCP) por la lista de fragmentos de la que disponen.

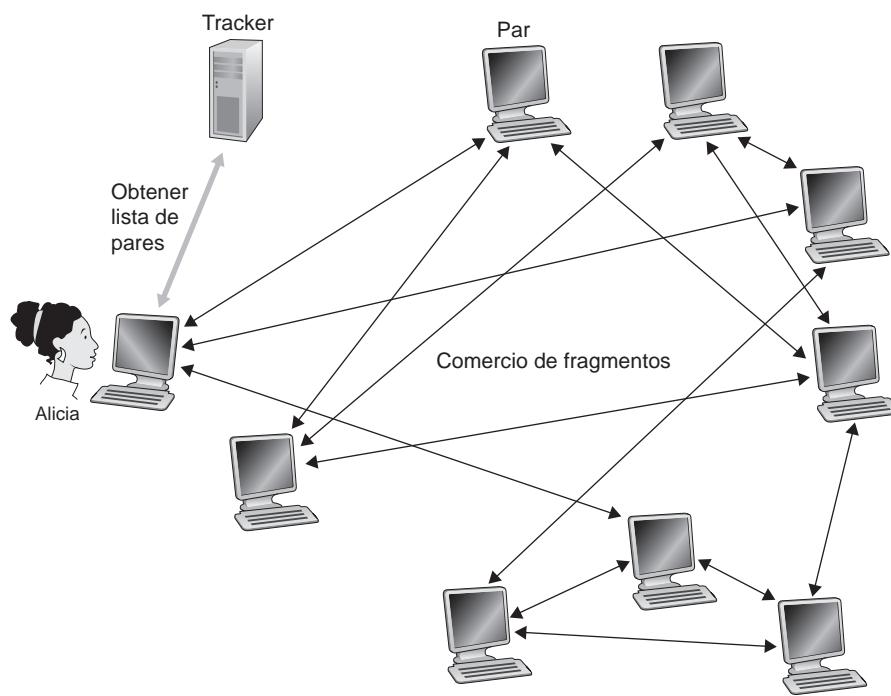


Figura 2.24 ♦ Distribución de archivos con BitTorrent.

Si Alicia tiene L vecinos diferentes obtendrá L listas de fragmentos. Con esta información, Alicia solicitará (a través de las conexiones TCP) los fragmentos que ella no tiene.

De esta manera, en un determinado instante, Alicia tendrá un subconjunto de fragmentos y sabrá qué fragmentos tienen sus vecinos. Con esta información, Alicia tendrá que tomar dos importantes decisiones. En primer lugar, qué fragmentos debe solicitar primero a sus vecinos. Y en segundo lugar, a cuáles de sus vecinos debe enviar ella los fragmentos solicitados. Para decidir qué fragmentos solicitar, Alicia utiliza una técnica conocida como **primero el menos común**. La idea es determinar, de entre los fragmentos que ella no tiene, los fragmentos menos comunes entre sus vecinos (es decir, los fragmentos de los que existe el menor número de copias repetidas repartidas entre los vecinos) y solicitar entonces en primer lugar esos fragmentos menos comunes. De esta manera, dichos fragmentos se redistribuirán más rápidamente, consiguiendo que el número de copias de cada fragmento sea aproximadamente igual dentro del torrente.

Para determinar a qué solicitudes debe ella responder, BitTorrent utiliza un algoritmo de intercambio inteligente. La idea básica es que Alicia dé prioridad a los vecinos que actualmente están suministrando sus datos *a mayor velocidad*. Específicamente, para cada uno de los vecinos, Alicia mide de forma continua la velocidad a la que recibe bits y determina cuáles son los cuatro pares que le envían bits a mayor velocidad. Entonces, ella, de forma recíproca, envía fragmentos a esos mismos cuatro pares. Cada 10 segundos, vuelve a calcular las velocidades y, posiblemente, tendrá que modificar el conjunto formado por los cuatro pares. En la jerga de BitTorrent, se dice que estos cuatro pares están **no filtrados** (*unchoked*). Además, y lo que es más importante, cada 30 segundos Alicia elige de forma aleatoria un vecino adicional y le envía fragmentos. Supongamos que el par elegido aleatoriamente es el de Benito. En la jerga de BitTorrent, se dice que Benito está **no filtrado de forma optimista**. Dado que Alicia está enviando datos a Benito, ella puede convertirse en uno de los cuatro suministradores principales de Benito, en cuyo caso Benito comenzaría a enviar datos a Alicia. Si la velocidad a la que Benito envía datos a Alicia es lo suficientemente alta, Benito podría entonces, a su vez, convertirse en uno de los cuatro suministradores principales de Alicia. En otras palabras, cada 30 segundos Alicia elegirá aleatoriamente un nuevo socio de intercambio e iniciará

las transacciones con él. Si los dos pares están satisfechos con el intercambio, se incluirán en sus respectivas listas de los cuatro principales y continuarán realizando intercambios hasta que uno de los pares encuentre un socio mejor. El efecto es que los pares capaces de suministrar datos a velocidades compatibles tienden a emparejarse. La selección aleatoria de vecinos también permite a los nuevos pares obtener fragmentos, con el fin de tener algo que intercambiar. Todos los demás pares vecinos excepto estos cinco (los cuatro pares “principales” más el par de prueba) están “filtrados”, es decir, no reciben fragmentos de Alicia. BitTorrent dispone de una serie de interesantes mecanismos que no vamos a ver aquí, entre los que se incluyen la gestión de piezas (minifragmentos), el procesamiento en cadena, la selección aleatoria del primer fragmento, el modo *endgame* y el *anti-snubbing* [Cohen 2003].

El mecanismo de incentivos para intercambio que acabamos de describir a menudo se denomina *tit-for-tat* (toma y daca, una estrategia de la teoría de juegos) [Cohen 2003]. Se ha demostrado que este esquema de incentivos puede soslayarse maliciosamente [Liogkas 2006; Locher 2006; Piatek 2007]. No obstante, el ecosistema BitTorrent ha tenido un éxito bárbaro, con millones de pares simultáneos compartiendo activamente archivos en cientos de miles de torrentes. Si BitTorrent se hubiera diseñado sin la estrategia *tit-for-tat* (o una variante), y aunque todo el resto de características fueran las mismas, es posible que BitTorrent no existiera actualmente, ya que la mayor parte de los usuarios hubieran pretendido aprovecharse de los demás [Saroiu 2002].

Vamos a terminar esta exposición sobre P2P mencionando brevemente otra aplicación de P2P, las tablas hash distribuidas (DHT, *Distributed Hash Table*). Una tabla hash distribuida es una base de datos simple, diatribuyéndose los registros de la base de datos a través de los pares de un sistema P2P. Las DHT han sido ampliamente implementadas (por ejemplo, en BitTorrent) y han sido objeto de exhaustivas investigaciones. Proporcionamos una introducción a las mismas en una nota de vídeo disponible en el sitio web de acompañamiento.



Nota de vídeo

Introducción a las tablas hash distribuidas

2.6 Flujos de vídeo y redes de distribución de contenido

La distribución de flujos pregrabados de vídeo representa ya la mayor parte del tráfico en los ISP residenciales de Norteamérica. En concreto, sólo los servicios de Netflix y YouTube consumieron un asombroso 37% y 16%, respectivamente, del tráfico de los ISP residenciales en 2015 [Sandvine 2015]. En esta sección proporcionaremos un resumen del modo en que se implementan en la Internet de hoy en día los servicios más populares de flujos de vídeo. Como veremos, se implementan utilizando protocolos de nivel de aplicación y servidores que funcionan, en cierto modo, como una caché. En el Capítulo 9, dedicado a las redes multimedia, examinaremos más en detalle el tema del vídeo por Internet, así como otros servicios Internet de carácter multimedia.

2.6.1 Vídeo por Internet

En las aplicaciones con flujos de vídeo almacenado, el medio subyacente es un vídeo pregrabado, como por ejemplo una película, un programa de televisión, un evento deportivo en diferido o un vídeo pregrabado generado por el usuario (como los que se pueden ver comúnmente en YouTube). Estos vídeos pregrabados se almacenan en servidores, y los usuarios envían solicitudes a los servidores para ver los vídeos *a la carta*. Muchas empresas de Internet proporcionan hoy en día flujos de vídeo, como por ejemplo Netflix, YouTube (Google), Amazon y Youku.

Pero antes de entrar a analizar los flujos de vídeo, conviene primero echar un rápido vistazo al propio medio subyacente: el vídeo. Un vídeo es una secuencia de imágenes, que normalmente se visualizan a velocidad constante, por ejemplo de 24 o 30 fotogramas por segundo. Una imagen codificada digitalmente y no comprimida, consta de una matriz de píxeles, codificándose cada píxel mediante una serie de bits que representan la luminancia y el color. Una característica importante del vídeo es que se puede comprimir, sacrificando algo de calidad de imagen a cambio de reducir la tasa

de transmisión de bits. Los algoritmos comerciales de compresión existentes hoy en día permiten comprimir un vídeo prácticamente a cualquier tasa de bits que se desee. Por supuesto, cuanto mayor sea la tasa de bits, mayor será la calidad de la imagen y mejor será la experiencia global del usuario, en lo que a visualización se refiere.

Desde la perspectiva de las redes, quizá la característica más destacable del vídeo sea su alta tasa de bits. El vídeo comprimido para Internet suele requerir entre 100 kbps (para vídeo de baja calidad) y más de 3 Mbps (para flujos de películas de alta resolución); los flujos de vídeo en formato 4K prevén una tasa de bits superior a 10 Mbps. Esto se traduce en una enorme cantidad de tráfico y de almacenamiento, particularmente para vídeo de alta gama. Por ejemplo, un único vídeo a 2 Mbps con una duración de 67 minutos consumirá 1 gigabyte de almacenamiento y de tráfico. La medida de rendimiento más importante para los flujos de vídeo es, con mucho, la tasa de transferencia media de extremo a extremo. Para poder garantizar una reproducción continua, la red debe proporcionar a la aplicación de flujos de vídeo una tasa de transferencia media que sea igual o superior a la tasa de bits del vídeo comprimido.

También podemos usar la compresión para crear múltiples versiones del mismo vídeo, cada una con un nivel diferente de calidad. Por ejemplo, podemos usar la compresión para crear tres versiones del mismo vídeo, con tasas de 300 kbps, 1 Mbps y 3 Mbps. Los usuarios puede entonces decidir qué versión quieren ver, en función de su ancho de banda actualmente disponible. Los usuarios con conexiones Internet de alta velocidad podrían seleccionar la versión a 3 Mbps, mientras que los usuarios que vayan a ver el vídeo a través de un teléfono inteligente con tecnología 3G podrían seleccionar la versión a 300 kbps.

2.6.2 Flujos de vídeo HTTP y tecnología DASH

En los flujos multimedia HTTP, el vídeo simplemente se almacena en un servidor HTTP como un archivo normal, con un URL específico. Cuando un usuario quiere ver el vídeo, el cliente establece una conexión TCP con el servidor y emite una solicitud GET HTTP para dicho URL. El servidor envía entonces el archivo de vídeo dentro de un mensaje de respuesta HTTP, con la máxima velocidad que permitan los protocolos de red subyacentes y las condiciones existentes de tráfico. En el lado del cliente, los bytes se acumulan en un buffer de la aplicación cliente. En cuanto el número de bytes en el buffer sobrepasa un umbral predeterminado, la aplicación cliente comienza la reproducción: para ser concretos, la aplicación de flujos de vídeo extrae periódicamente fotogramas de vídeo del buffer de la aplicación cliente, descomprime los fotogramas y los muestra en la pantalla del usuario. De ese modo, la aplicación de flujos de vídeo muestra el vídeo al mismo tiempo que recibe y almacena en el buffer otros fotogramas, correspondientes a secuencias posteriores del vídeo.

Aunque los flujos HTTP, tal como se describen en el párrafo anterior, se han implantado ampliamente en la práctica (por ejemplo por parte de YouTube, desde su nacimiento), presentan una carencia fundamental: todos los clientes reciben la misma versión codificada del vídeo, a pesar de las grandes variaciones existentes en cuanto al ancho de banda disponible para cada cliente, e incluso en cuanto al ancho de banda disponible para un cliente concreto a lo largo del tiempo. Esto condujo al desarrollo de un nuevo tipo de flujos de vídeo basados en HTTP, una tecnología a la que se suele denominar **DASH**, (*Dynamic Adaptive Streaming over HTTP*, **Flujos dinámicos adaptativos sobre HTTP**). En DASH, el vídeo se codifica en varias versiones diferentes, teniendo cada versión una tasa de bits distinta y, por tanto, un nivel de calidad diferente. El cliente solicita dinámicamente segmentos de vídeo de unos pocos segundos de duración. Cuando el ancho de banda disponible es grande, el cliente selecciona de forma natural los segmentos de una versión de alta tasa de bits; y cuando el ancho de banda disponible es pequeño, selecciona de forma natural los segmentos de una versión con menor tasa de bits. El cliente selecciona los segmentos de uno en uno mediante mensajes de solicitud GET HTTP [Akhshabi 2011].

DASH permite que los clientes con diferentes tasas de acceso a Internet reciban flujos de vídeo con tasas de codificación diferentes. Los clientes con conexiones 3G a baja velocidad pueden recibir una versión con baja tasa de bits (y baja calidad), mientras que los clientes con conexiones de fibra

pueden recibir una versión de alta calidad. DASH también permite a un cliente adaptarse al ancho de banda disponible, si el ancho de banda extremo a extremo varía durante la sesión. Esta característica es particularmente importante para los usuarios móviles, que suelen experimentar fluctuaciones del ancho de banda disponible a medida que se desplazan con respecto a las estaciones base.

Con DASH, cada versión del vídeo se almacena en un servidor HTTP, teniendo cada versión un URL distinto. El servidor HTTP dispone también de un **archivo de manifiesto**, que indica el URL de cada versión, junto con su correspondiente tasa de bits. El cliente solicita primero el archivo de manifiesto y determina cuáles son las diferentes versiones disponibles. Después solicita los segmentos de uno en uno, especificando un URL y un rango de bytes mediante un mensaje de solicitud GET HTTP para cada segmento. Mientras está descargando los segmentos, el cliente también mide el ancho de banda de recepción y ejecuta un algoritmo de determinación de la tasa de bits, para seleccionar el segmento que debe solicitar a continuación. Naturalmente, si el cliente tiene una gran cantidad de vídeo almacenado en el buffer y si el ancho de banda de recepción medido es grande, seleccionará un segmento correspondiente a una versión con alta tasa de bits. Igualmente, si tiene poca cantidad de vídeo almacenado en el buffer y el ancho de banda de recepción medido es pequeño, seleccionará un segmento correspondiente a una versión con baja tasa de bits. DASH permite, de ese modo, que el cliente cambie libremente entre distintos niveles de calidad.

2.6.3 Redes de distribución de contenido

Actualmente, muchas empresas de vídeo a través de Internet están distribuyendo a la carta flujos de vídeo de múltiples Mbps a millones de usuarios diariamente. YouTube, por ejemplo, con una librería de cientos de millones de vídeos, distribuye a diario centenares de millones de flujos de vídeo a usuarios repartidos por todo el mundo. Enviar todo este tráfico a ubicaciones de todo el mundo, al mismo tiempo que se proporciona una reproducción continua y una alta interactividad, constituye claramente un auténtico desafío.

Para una empresa de vídeo a través de Internet, quizás la solución más directa para proporcionar un servicio de flujos de vídeo sea construir un único centro de datos masivo, almacenar todos los vídeos en ese centro de datos y enviar los flujos de vídeo directamente desde el centro de datos a clientes repartidos por todo el mundo. Pero esta solución tiene tres problemas principales. En primer lugar, si el cliente está lejos del centro de datos, los paquetes que viajan desde el servidor al cliente atravesarán muchos enlaces de comunicaciones y probablemente atraviesen muchos ISP, estando algunos de los ISP posiblemente ubicados en continentes distintos. Si uno de esos enlaces proporciona una tasa de transferencia inferior a la velocidad a la que se consume el vídeo, la tasa de transferencia extremo a extremo también será inferior a la tasa de consumo, lo que provocará molestas congelaciones de la imagen de cara al usuario. (Recuerde del Capítulo 1 que la tasa de transferencia extremo a extremo de un flujo de datos está determinada por la tasa de transferencia del enlace que actúe como cuello de botella.) La probabilidad de que esto suceda se incrementa a medida que aumenta el número de enlaces que componen la ruta extremo a extremo. Una segunda desventaja es que un vídeo muy popular será probablemente enviado muchas veces a través de los mismos enlaces de comunicaciones. Esto no solo hace que se desperdicie ancho de banda de la red, sino que la propia empresa de vídeo a través de Internet estará pagando a su ISP proveedor (conectado al centro de datos) por enviar los *mismos* bytes una y otra vez a Internet. Un tercer problema de esta solución es que un único centro de datos representa un punto único de fallo, si se caen el centro de datos o sus enlaces de conexión con Internet, la empresa no podrá distribuir *ningún* flujo de vídeo.

Para poder afrontar el desafío de distribuir cantidades masivas de datos de vídeo a usuarios dispersos por todo el mundo, casi todas las principales empresas de flujos de vídeo utilizan **redes de distribución de contenido (CDN, Content Distribution Network)**. Una CDN gestiona servidores situados en múltiples ubicaciones geográficamente distribuidas, almacena copias de los vídeos (y de otros tipos de contenido web, como documentos, imágenes y audio) en sus servidores y trata de dirigir cada solicitud de usuario a una ubicación de la CDN que proporcione la mejor experiencia de usuario posible. La CDN puede ser una **CDN privada**, es decir, propiedad del

propio proveedor de contenido; por ejemplo, la CDN de Google distribuye vídeos de YouTube y otros tipos de contenido. Alternativamente, la CDN puede ser una **CDN comercial** que distribuya contenido por cuenta de múltiples proveedores de contenido; Akamai, Limelight y Level-3, por ejemplo, operan redes CDN comerciales. Un resumen muy legible de las redes CDN modernas es [Leighton 2009; Nygren 2010].

Las redes CDN adoptan normalmente una de dos posibles filosofías de colocación de los servidores [Huang 2008]:

- **Introducción profunda.** Una de las filosofías, de la que Akamai fue pionera, consiste en *introducirse en profundidad* en las redes de acceso de los proveedores de servicios Internet (ISP), implantando clústeres de servidores en proveedores ISP de acceso por todo el mundo. (Las redes de acceso se describen en la Sección 1.3.) Akamai ha adoptado esta solución con clústeres de servidores en aproximadamente 1.700 ubicaciones. El objetivo es acercarse a los usuarios finales, mejorando así el retardo percibido por el usuario y la tasa de transferencia por el procedimiento de reducir el número de enlaces y de routers existentes entre el usuario final y el servidor CDN del que recibe el contenido. Debido a este diseño altamente distribuido, la tarea de mantener y gestionar los clústeres se convierte en todo un desafío.
- **Atraer a los ISP.** Una segunda filosofía de diseño, adoptada por Limelight y muchas otras empresas de redes CDN, consiste en *atraer a los ISP* construyendo grandes clústeres en un número más pequeño (por ejemplo, unas decenas) de lugares. En lugar de introducirse en los ISP de acceso, estas CDN suelen colocar sus clústeres en puntos de intercambio Internet (IXP, *Internet Exchange Point*, véase la Sección 1.3). Comparada con la filosofía de diseño basada en la introducción profunda, el diseño basado en atraer a los ISP suele tener menores costes de mantenimiento y gestión, posiblemente a cambio de un mayor retardo y una menor tasa de transferencia para los usuarios finales.

Una vez implantados los clústeres, la CDN replica el contenido entre todos ellos. La red CDN puede no siempre almacenar una copia de cada vídeo en cada clúster, ya que algunos vídeos solo se visualizan en raras ocasiones o solo son populares en ciertos países. De hecho, muchas CDN no copian activamente los vídeos en sus clústeres, sino que usan una estrategia simple: si un cliente solicita un vídeo de un clúster que no lo tiene almacenado, el clúster extrae el vídeo (de un repositorio central o de otro clúster) y almacena una copia localmente, al mismo tiempo que envía el flujo de vídeo al cliente. De forma similar a lo que ocurre con las cachés web (véase la Sección 2.2.5), cuando el espacio de almacenamiento del clúster se llena, el clúster elimina los vídeos que no son solicitados frecuentemente.

Funcionamiento de una red CDN

Habiendo identificado las dos soluciones principales de implantación de una CDN, profundicemos en el modo en que una de estas redes opera. Cuando se indica al navegador del host de un usuario que extraiga un vídeo concreto (identificado mediante un URL), la CDN debe interceptar la solicitud para (1) determinar un clúster de servidores de la CDN que resulte adecuado para ese cliente en ese preciso instante y (2) redirigir la solicitud del cliente a un servidor situado en dicho clúster. En breve veremos cómo puede la CDN determinar un clúster adecuado. Pero antes, examinemos la mecánica del proceso de interceptación y redirección de una solicitud.

La mayoría de las CDN aprovechan DNS para interceptar y redirigir las solicitudes; un análisis interesante de esa utilización de DNS es [Vixie 2009]. Consideremos un ejemplo simple para ilustrar el modo en que DNS suele participar. Suponga que un proveedor de contenido, NetCinema, utiliza a una empresa proveedora de servicios CDN, KingCDN, para distribuir sus vídeos a sus clientes. En las páginas web de NetCinema, cada uno de los vídeos tiene asignado un URL que incluye la cadena “video” y un identificador único del propio vídeo; por ejemplo, a *Transformers 7* se le podría asignar <http://video.netcinema.com/6Y7B23V>. Entonces se sucederán seis pasos, como se muestra en la Figura 2.25:

CASO DE ESTUDIO

INFRAESTRUCTURA DE RED DE GOOGLE

Para dar soporte a su amplia variedad de servicios en la nube —incluyendo las búsquedas, Gmail, calendarios, vídeos YouTube, mapas, documentos y redes sociales—, Google ha implantado una amplia red privada y una infraestructura CDN. La infraestructura CDN de Google tiene tres niveles de clústeres de servidores:

- Catorce “mega-centros de datos” (ocho en Norteamérica, cuatro en Europa y dos en Asia [Google Locations 2016]), teniendo cada centro de datos del orden de 100.000 servidores. Estos mega-centros de datos se encargan de servir contenido dinámico (y a menudo personalizado), incluyendo resultados de búsqueda y mensajes Gmail.
- Unos 50 clústeres en IXP dispersos por todo el mundo, consistiendo cada clúster en unos 100-500 servidores [Adhikari 2011a]. Estos clústeres son responsables de servir contenido estático, incluyendo vídeos de YouTube [Adhikari 2011a].
- Varios cientos de clústeres de “introducción profunda”, ubicados dentro de proveedores ISP de acceso. En este caso, los clústeres suelen estar compuestos por decenas de servidores, situados en un mismo bastidor. Estos servidores de introducción profunda se encargan de la división TCP (véase la Sección 3.7) y de servir contenido estático [Chen 2011], incluyendo las partes estáticas de las páginas web donde se insertan los resultados de búsqueda.

Todos estos centros de datos y clústeres de servidores están conectados en red mediante la propia red privada de Google. Cuando un usuario hace una búsqueda, a menudo la búsqueda se envía primero a través del ISP local hasta una caché cercana de introducción profunda, de donde se extrae el contenido estático; mientras se proporciona el contenido estático al cliente, la caché cercana reenvía también la consulta a través de la red privada de Google hasta uno de los mega-centros de datos, de donde se extraen los resultados de búsqueda personalizados. Para un vídeo de YouTube, el propio vídeo puede provenir de una de las cachés en los IXP, mientras que parte de la página web que rodea al vídeo puede provenir de la caché cercana de introducción profunda y los anuncios que rodean al vídeo vienen de los centros de datos. Resumiendo: salvo por lo que se refiere al ISP local, los servicios en la nube de Google son proporcionados, en buena medida, por una infraestructura de red que es independiente de la Internet pública.

1. El usuario visita la página web en NetCinema.
2. Cuando el usuario hace clic sobre el vínculo <http://video.netcinema.com/6Y7B23V>, el host del usuario envía una solicitud DNS preguntando por video.netcinema.com.
3. El servidor DNS local del usuario (al que llamaremos LDNS) retransmite la solicitud DNS a un servidor DNS autoritativo de NetCinema, que observa la cadena “video” en el nombre de host video.netcinema.com. Para “transferir” la consulta DNS a KingCDN, lo que hace el servidor DNS autoritativo de NetCinema es, en vez de devolver una dirección IP, enviar al LDNS un nombre de host perteneciente al dominio de KingCDN, como por ejemplo a1105.kingcdn.com.
4. A partir de ese punto, la consulta DNS entra en la infraestructura DNS privada de KingCDN. El LDNS del usuario envía entonces una segunda consulta, preguntando ahora por a1105.kingcdn.com, y el sistema DNS de KingCDN termina por devolver al LDNS las direcciones IP de un servidor de contenido de KingCDN. Es por tanto aquí, dentro del sistema DNS de KingCDN, donde se especifica el servidor CDN desde el cual recibirá el cliente su contenido.
5. El LDNS reenvía al host del usuario la dirección IP del nodo CDN encargado de servir el contenido.

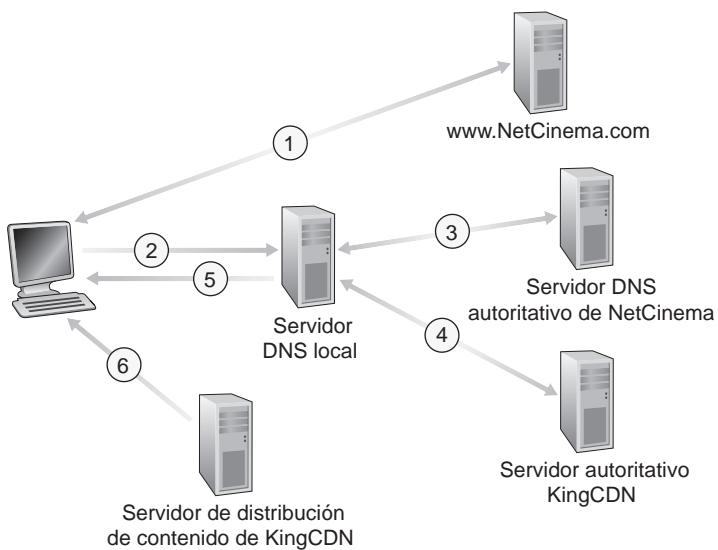


Figura 2.25 ♦ DNS redirige una solicitud de usuario hacia un servidor CDN.

6. Una vez que el cliente recibe la dirección IP de un servidor de contenido de KingCDN, establece una conexión TCP directa con el servidor situado en dicha dirección IP y transmite una solicitud GET HTTP para el vídeo deseado. Si se utiliza DASH, el servidor enviará primero al cliente un archivo de manifiesto con una lista de URL, uno para cada versión del vídeo, y el cliente seleccionará dinámicamente segmentos de las distintas versiones.

Estrategias de selección de clústeres

Uno de los fundamentos de cualquier implantación de una red CDN es la **estrategia de selección de clústeres**, es decir, el mecanismo para dirigir a los clientes dinámicamente hacia un clúster de servidores o un centro de datos pertenecientes a la CDN. Como acabamos de ver, la CDN determina la dirección IP del servidor LDNS del cliente a partir de la búsqueda DNS realizada por el cliente. Después de determinar esta dirección IP, la red CDN necesita seleccionar un clúster apropiado, dependiendo de dicha dirección. Las redes CDN emplean, generalmente, estrategias propietarias de selección de clústeres. Vamos a ver brevemente algunas soluciones, cada una de las cuales tiene sus ventajas y sus desventajas.

Una estrategia sencilla consiste en asignar al cliente al clúster **geográficamente más próximo**. Usando bases de datos comerciales de geolocalización (como Quova [Quova 2016] y MaxMind [MaxMind 2016]), la dirección IP de cada LDNS se hace corresponder con una ubicación geográfica. Cuando se recibe una solicitud DNS de un LDNS concreto, la CDN selecciona el clúster geográficamente más próximo, es decir, el clúster situado a menos kilómetros “a vuelo de pájaro” del LDNS. Una solución de este estilo puede funcionar razonablemente bien para una gran parte de los clientes [Agarwal 2009]. Sin embargo, para algunos clientes la solución puede proporcionar un mal rendimiento, porque el clúster geográficamente más cercano no es necesariamente el clúster más próximo en términos de la longitud o el número de saltos de la ruta de red. Además, un problema inherente a todas las soluciones basadas en DNS es que algunos usuarios finales están configurados para usar servidores LDNS remotos [Shaikh 2001; Mao 2002], en cuyo caso la ubicación del LDNS puede estar lejos de la del cliente. Además, esta estrategia tan simple no tiene en cuenta la variación a lo largo del tiempo del retardo y del ancho de banda disponible de las rutas en Internet, asignando siempre el mismo clúster a cada cliente concreto.

Para determinar el mejor clúster para un cliente basándose en las condiciones *actuales* de tráfico, las redes CDN pueden, alternativamente, realizar periódicamente **medidas en tiempo real** del retardo y del comportamiento de pérdidas entre sus clústeres y los clientes. Por ejemplo, una CDN puede hacer que todos sus clústeres envíen periódicamente mensajes de sondeo (por ejemplo, mensajes ping o consultas DNS) a todos los LDNS de todo el mundo. Una desventaja de esta técnica es que muchos LDNS están configurados para no responder a tales mensajes de sondeo.

2.6.4 Casos de estudio: Netflix, YouTube y Kankan

Terminamos nuestra exposición sobre los flujos de vídeo almacenados echando un vistazo a tres implantaciones a gran escala de enorme éxito: Netflix, YouTube y Kankan. Veremos que cada uno de estos sistemas adopta una solución muy diferente, aunque todos ellos emplean muchos de los principios subyacentes expuestos en esta sección.

Netflix

Netflix, que generó el 37% del tráfico de bajada en los ISP residenciales de Norteamérica en 2015, se ha convertido en el principal proveedor de servicios para películas y series de TV en línea en los Estados Unidos [Sandvine 2015]. Como vamos a ver, la distribución de vídeo de Netflix tiene dos componentes principales: la nube de Amazon y su propia infraestructura CDN privada.

Netflix dispone de un sitio web que se encarga de gestionar numerosas funciones, incluyendo los registros e inicios de sesión de los usuarios, la facturación, el catálogo de películas que puede hojearse o en el que se pueden realizar búsquedas y un sistema de recomendación de películas. Como se muestra en la Figura 2.26, este sitio web (y sus bases de datos *back-end* asociadas) se ejecuta enteramente en servidores de Amazon, dentro de la nube de Amazon. Además, la nube de Amazon se encarga de las siguientes funciones críticas:

- **Ingesta de contenidos.** Antes de que Netflix pueda distribuir una película a sus usuarios, debe primero realizar la ingestión y procesar la película. Netflix recibe versiones maestras de estudio de las películas y las carga en hosts situados en la nube de Amazon.

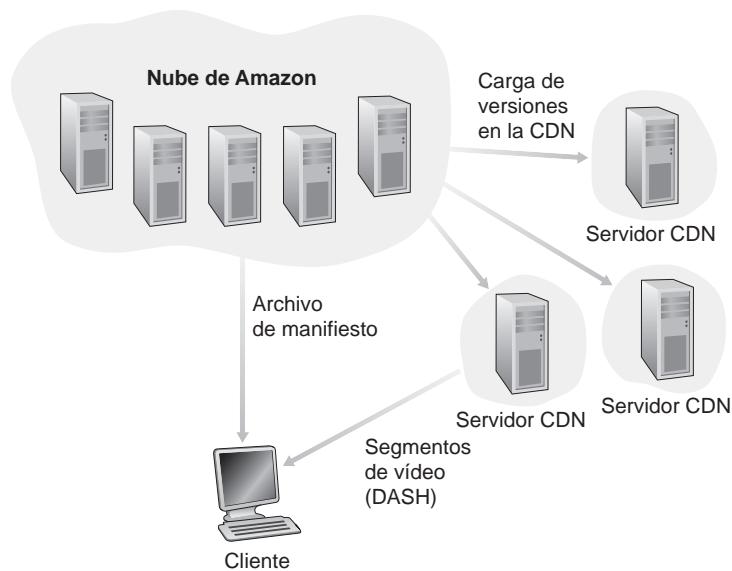


Figura 2.26 ♦ Plataforma de flujos de video de Netflix.

- **Procesamiento del contenido.** Las máquinas de la nube de Amazon generan muchos formatos distintos para cada película, adecuados para una amplia variedad de clientes de reproducción de vídeo que se ejecutan en computadoras de sobremesa, teléfonos inteligentes y consolas de juegos conectadas a televisiones. Para cada uno de estos formatos se crea una versión diferente, con múltiples tasas de bits, lo que permite un envío adaptativo de los flujos multimedia a través de HTTP, usando DASH.
- **Carga de las versiones en su CDN.** Una vez generadas todas las versiones de una película, los hosts de la nube de Amazon cargan esas versiones en la CDN de Netflix.

Cuando Netflix inauguró su servicio de distribución de flujos de vídeo en 2007, empleaba tres empresas proveedoras de servicios de red CDN para distribuir su contenido de vídeo. Desde entonces, Netflix ha creado su propia CDN privada, desde la cual distribuye ahora todos sus flujos de vídeo. (Sin embargo, Netflix sigue usando Akamai para distribuir sus páginas web.) Para crear su propia CDN, Netflix ha instalado bastidores de servidores tanto en IXP como dentro de los propios ISP residenciales. Netflix dispone actualmente de bastidores de servidores en más de 50 ubicaciones de IXP; en [Netflix Open Connect 2016] puede ver una lista actualizada de IXP que albergan bastidores de Netflix. También hay centenares de ubicaciones de ISP que albergan bastidores de Netflix; véase también [Netflix Open Connect 2016], donde Netflix proporciona a los potenciales ISP asociados instrucciones sobre cómo instalar un bastidor Netflix (gratuito) para sus redes. Cada servidor del bastidor dispone de varios puertos Ethernet a 10 Gbps y de más de 100 terabytes de almacenamiento. El número de servidores de un bastidor es variable: las instalaciones de los IXP suelen tener decenas de servidores y contienen toda la biblioteca de flujos de vídeo de Netflix, incluyendo las múltiples versiones de los vídeos que hacen falta para soportar DASH; los ISP locales pueden disponer de un único servidor y almacenar solo los vídeos más populares. Netflix no utiliza un sistema de caché que se rellena bajo demanda (*pull-caching*) para cargar el contenido en sus servidores CDN ubicados en los IXP e ISP. En lugar de ello, Netflix realiza la distribución cargando activamente los vídeos en sus servidores CDN durante las horas menor tráfico. Para aquellas ubicaciones que no pueden almacenar la biblioteca completa, Netflix carga solo los vídeos más populares, que se determinan diariamente. El diseño de CDN de Netflix se describe con un cierto grado de detalle en los vídeos de Youtube [Netflix Video 1] y [Netflix Video 2].

Habiendo descrito los componentes de la arquitectura Netflix, examinemos más detalladamente la interacción entre el cliente y los distintos servidores implicados en la distribución de las películas. Como hemos dicho anteriormente, las páginas web a través de las que se explora la videoteca de Netflix se sirven desde servidores situados en la nube de Amazon. Cuando un usuario selecciona una película para reproducirla, el software de Netflix, ejecutándose en la nube de Amazon, determina primero cuáles de sus servidores CDN disponen de una copia de la película. A continuación, el software determina cuál de entre los servidores que disponen de la película es el “mejor” para esa solicitud del cliente. Si el cliente está utilizando un ISP residencial que tiene instalado un bastidor de servidores de la CDN de Netflix, y si ese bastidor dispone de una copia de la película solicitada, entonces suele seleccionarse un servidor de ese bastidor. Si no, lo que se suele seleccionar es un servidor de algún IXP cercano.

Una vez que Netflix ha determinado el servidor CDN que tiene que distribuir el contenido, envía al cliente la dirección IP de ese servidor concreto, junto con un archivo de manifiesto, que contiene los URL de las diferentes versiones de la película solicitada. A continuación, el cliente y ese servidor CDN interaccionan directamente, usando una versión propietaria de DASH. Específicamente, como se describe en la Sección 2.6.2, el cliente usa la cabecera de rango de bytes de los mensajes de solicitud GET HTTP para solicitar segmentos de las diferentes versiones de la película. Netflix usa segmentos de una duración aproximada de cuatro segundos Adhikari 2012]. Mientras se descargan los segmentos, el cliente mide la tasa de transferencia de recepción y ejecuta un algoritmo de determinación de la velocidad para identificar la calidad del segmento que debe solicitar a continuación.

Netflix utiliza muchos de los principios básicos que hemos expuesto anteriormente en esta sección, incluyendo los flujos adaptativos y la distribución a través de una CDN. Sin embargo, como Netflix emplea su propia CDN privada, que solo distribuye vídeo (y no páginas web), Netflix ha sido capaz de simplificar y adaptar su diseño de CDN. En particular, Netflix no necesita usar la redirección DNS, explicada en la Sección 2.6.3, para conectar un cliente concreto con un servidor CDN; en lugar de ello, el software de Netflix (que se ejecuta en la nube de Amazon) instruye directamente al cliente para que utilice un servidor CDN concreto. Además, la CDN de Netflix carga el contenido de la caché en sus servidores en momentos planificados (*push-caching*), durante las horas de menor tráfico, en lugar de cargarlo dinámicamente a medida que se producen fallos de localización en caché.

You Tube

Con 300 horas de vídeo cargadas en YouTube cada minuto y varios miles de millones de reproducciones diarias de vídeo [YouTube 2016], YouTube es, sin lugar a dudas, el mayor sitio de compartición de vídeos del mundo. YouTube comenzó a prestar servicio en abril de 2005 y fue adquirido por Google en noviembre de 2006. Aunque el diseño y los protocolos de Google/YouTube son propietarios, podemos hacernos una idea básica de cómo opera YouTube gracias a diversos trabajos de medida independientes [Zink 2009; Torres 2011; Adhikari 2011a]. Al igual que Netflix, YouTube hace un amplio uso de la tecnología CDN para distribuir sus vídeos [Torres 2011]. De forma similar a Netflix, Google utiliza su propia CDN privada para distribuir los vídeos de YouTube y ha instalado clústeres de servidores en muchos cientos de ubicaciones IXP e ISP distintas. Google distribuye los vídeos de YouTube desde estas ubicaciones y directamente desde sus inmensos centros de datos [Adhikari 2011a]. A diferencia de Netflix, sin embargo, Google emplea *pull-caching* y un mecanismo de redirección DNS, como se describe en la Sección 2.6.3. La mayoría de las veces, la estrategia de selección de clústeres de Google dirige al cliente hacia el clúster que tenga un menor RTT entre el clúster y el cliente; sin embargo, para equilibrar la carga entre los clústeres, en ocasiones se dirige al cliente (a través de DNS) a un clúster más distante [Torres 2011].

YouTube emplea flujos HTTP, ofreciendo a menudo un pequeño número de versiones distintas de cada vídeo, cada una con diferente tasa de bits y, correspondientemente, un diferente nivel de calidad. YouTube no utiliza flujos adaptativos (como DASH), sino que exige al cliente que seleccione una versión de forma manual. Para ahorrar ancho de banda y recursos de servidor, que se desperdiciarían en caso de que el usuario efectúe un reposicionamiento o termine la reproducción anticipadamente, YouTube emplea la solicitud de rango de bytes de HTTP para limitar el flujo de datos transmitidos, después de precargar una cierta cantidad predeterminada de vídeo.

Cada día se cargan en YouTube varios millones de vídeos. No solo se distribuyen a través de HTTP los flujos de vídeo de YouTube de los servidores a los clientes, sino que los usuarios que cargan vídeos en YouTube desde el cliente hacia el servidor también los cargan a través de HTTP. YouTube procesa cada vídeo que recibe, convirtiéndolo a formato de vídeo de YouTube y creando múltiples versiones con diferentes tasas de bits. Este procesamiento se realiza enteramente en los centros de datos de Google (véase el caso de estudio sobre la infraestructura de red de Google en la Sección 2.6.3).

Kankan

Acabamos de ver cómo una serie de servidores dedicados, operados por redes CDN privadas, se encargan de distribuir a los clientes los vídeos de Netflix y YouTube. Ambas empresas tienen que pagar no solo por el hardware del servidor, sino también por el ancho de banda que los servidores usan para distribuir los vídeos. Dada la escala de estos servicios y la cantidad de ancho de banda que consumen, ese tipo de implantación de una CDN puede ser costoso.

Concluiremos esta sección describiendo un enfoque completamente distinto para la provisión a gran escala de vídeos a la carta a través de Internet - un enfoque que permite al proveedor del servicio reducir significativamente sus costes de infraestructura y de ancho de banda. Como el lector estará suponiendo, este enfoque utiliza distribución P2P en lugar de (o además de) distribución cliente-servidor. Desde 2011, Kankan (cuyo propietario y operador es Xunlei) ha estado implantando con gran éxito su sistema P2P de distribución de vídeo, que cuenta con decenas de millones de usuarios cada mes [Zhang 2015].

A alto nivel, los flujos de vídeo P2P son muy similares a la descarga de archivos con BitTorrent. Cuando uno de los participantes quiere ver un vídeo, contacta con un *tracker* para descubrir otros homólogos en el sistema que dispongan de una copia de ese vídeo. El homólogo solicitante pide entonces segmentos de vídeo en paralelo a todos los homólogos que dispongan de él. Sin embargo, a diferencia de lo que sucede con la descarga en BitTorrent, las solicitudes se realizan preferentemente para segmentos que haya que reproducir en un futuro próximo, con el fin de garantizar una reproducción continua [Dhungel 2012].

Recientemente, Kankan ha efectuado la migración a un sistema de flujos de vídeo híbrido CDN-P2P [Zhang 2015]. Específicamente, Kankan tiene ahora implantados unos pocos cientos de servidores en China y carga de forma activa contenido de vídeo en esos servidores. Esta CDN de Kankan juega un papel principal durante la etapa inicial de transmisión de los flujos de vídeo. En la mayoría de los casos, el cliente solicita el principio del contenido a los servidores de la CDN y en paralelo pide contenido a los homólogos. Cuando el tráfico P2P total es suficiente para la reproducción de vídeo, el cliente deja de descargar de la CDN y descarga sólo de los homólogos. Pero si el tráfico de descarga de flujos de vídeo P2P pasa a ser insuficiente, el cliente restablece las conexiones con la CDN y vuelve al modo de flujos de vídeo híbrido CDN-P2P. De esta manera, Kankan puede garantizar retardos iniciales de arranque cortos, al mismo tiempo que minimiza la utilización de ancho de banda y de una costosa infraestructura de servidores.

2.7 Programación de sockets: creación de aplicaciones de red

Ahora que hemos examinado una serie de importantes aplicaciones de red, vamos a ver cómo se escriben en la práctica los programas de aplicaciones de redes. Recuerde de la Sección 2.1 que muchas aplicaciones de red están compuestas por una pareja de programas (un programa cliente y un programa servidor) que residen en dos sistemas terminales distintos. Cuando se ejecutan estos dos programas, se crean un proceso cliente y un proceso servidor, y estos dos procesos se comunican entre sí leyendo y escribiendo en sockets. Cuando se crea una aplicación de red, la tarea principal del desarrollador es escribir el código para los programas cliente y servidor.

Existen dos tipos de aplicaciones de red. Uno de ellos es una implementación de un estándar de protocolo definido en, por ejemplo, un RFC o algún otro documento relativo a estándares. Para este tipo de implementaciones, los programas cliente y servidor deben adaptarse a las reglas dictadas por ese RFC. Por ejemplo, el programa cliente podría ser una implementación del lado del cliente del protocolo HTTP, descrito en la Sección 2.2 y definido explícitamente en el documento RFC 2616; de forma similar, el programa servidor podría ser una implementación del protocolo de servidor HTTP, que también está definido explícitamente en el documento RFC 2616. Si un desarrollador escribe código para el programa cliente y otro desarrollador independiente escribe código para el programa servidor y ambos desarrolladores siguen cuidadosamente las reglas marcadas en el RFC, entonces los dos programas serán capaces de interoperar. Ciertamente, muchas de las aplicaciones de red actuales implican la comunicación entre programas cliente y servidor que han sido creados por desarrolladores independientes (por ejemplo, un navegador Google Chrome comunicándose con un servidor web Apache, o un cliente BitTorrent comunicándose con un tracker BitTorrent).

El otro tipo de aplicación de red son las aplicaciones propietarias. En este caso, el protocolo de la capa de aplicación utilizado por los programas cliente y servidor *no* tiene que cumplir necesariamente ninguna recomendación RFC existente. Un único desarrollador (o un equipo de desarrollo) crea tanto el programa cliente como el programa servidor, y ese desarrollador tiene el control completo sobre aquello que se incluye en el código. Pero como el código no implementa ningún protocolo abierto, otros desarrolladores independientes no podrán desarrollar código que interopere con esa aplicación.

En esta sección vamos a examinar los problemas fundamentales del desarrollo de aplicaciones propietarias cliente-servidor y echaremos un vistazo al código que implementa una aplicación cliente-servidor muy sencilla. Durante la fase de desarrollo, una de las primeras decisiones que el desarrollador debe tomar es si la aplicación se ejecutará sobre TCP o sobre UDP. Recuerde que TCP está orientado a la conexión y proporciona un canal fiable de flujo de bytes a través del cual se transmiten los datos entre los dos sistemas terminales. Por su parte, UDP es un protocolo sin conexión, que envía paquetes de datos independientes de un sistema terminal a otro, sin ningún tipo de garantía acerca de la entrega. Recuerde también que cuando un programa cliente o servidor implementa un protocolo definido por un RFC, debe utilizar el número de puerto bien conocido asociado con el protocolo; asimismo, al desarrollar una aplicación propietaria, el desarrollador debe evitar el uso de dichos números de puerto bien conocidos. (Los números de puerto se han explicado brevemente en la Sección 2.1 y los veremos en detalle en el Capítulo 3).

Vamos a presentar la programación de sockets en UDP y TCP mediante una aplicación UDP simple y una aplicación TCP simple. Mostraremos estas sencillas aplicaciones en Python 3. Podríamos haber escrito el código en Java, C o C++, pero hemos elegido Python fundamentalmente porque Python expone de forma clara los conceptos clave de los sockets. Con Python se usan pocas líneas de código, y cada una de ellas se puede explicar a un programador novato sin dificultad, por lo que no debe preocuparse si no está familiarizado con Python. Podrá seguir fácilmente el código si tiene experiencia en programación en Java, C o C++.

Si está interesado en la programación cliente-servidor con Java, le animamos a que consulte el sitio web de acompañamiento del libro; de hecho, allí podrá encontrar todos los ejemplos de esta sección (y las prácticas de laboratorio asociadas) en Java. Para aquellos lectores que estén interesados en la programación cliente-servidor en C, hay disponibles algunas buenas referencias [Donahoo 2001; Stevens 1997; Frost 1994; Kurose 1996]; los ejemplos en Python que proporcionamos a continuación tienen un estilo y aspecto similares a C.

2.7.1 Programación de sockets con UDP

En esta subsección vamos a escribir programas cliente-servidor simples que utilizan UDP. En la siguiente sección, escribiremos programas similares que emplean TCP.

Recuerde de la Sección 2.1 que los procesos que se ejecutan en máquinas diferentes se comunican entre sí enviando mensajes a través de sockets. Dijimos que cada proceso era análogo a una vivienda y que el socket del proceso era análogo a una puerta. La aplicación reside en un lado de la puerta de la vivienda; el protocolo de la capa de transporte reside en el otro lado de la puerta, en el mundo exterior. El desarrollador de la aplicación dispone de control sobre todo lo que está situado en el lado de la capa de aplicación del socket; sin embargo, el control que tiene sobre el lado de la capa de transporte es muy pequeño.

Veamos ahora la interacción existente entre dos procesos que están comunicándose que usan sockets UDP. Si se usa UDP, antes de que un proceso emisor pueda colocar un paquete de datos en la puerta del socket, tiene que asociar en primer lugar una dirección de destino al paquete. Una vez que el paquete atraviesa el socket del emisor, Internet utilizará la dirección de destino para enrutar dicho paquete hacia el socket del proceso receptor, a través de Internet. Cuando el paquete llega al socket de recepción, el proceso receptor recuperará el paquete a través del socket y a continuación inspeccionará el contenido del mismo y tomará las acciones apropiadas.

Así que puede que se esté preguntando ahora: ¿qué es lo que se introduce en la dirección de destino asociada al paquete? Como cabría esperar, la dirección IP del host de destino es parte de esa dirección de destino. Al incluir la dirección IP de destino en el paquete, los routers de Internet serán capaces de enrutar el paquete hasta el host de destino. Pero, dado que un host puede estar ejecutando muchos procesos de aplicaciones de red, cada uno de ellos con uno o más sockets, también es necesario identificar el socket concreto dentro del host de destino. Cuando se crea un socket, se le asigna un identificador, al que se denomina **número de puerto**. Por tanto, como cabría esperar, la dirección de destino del paquete también incluye el número de puerto del socket. En resumen, el proceso emisor asocia con el paquete una dirección de destino que está compuesta de la dirección IP del host de destino y del número de puerto del socket de destino. Además, como veremos enseguida, también se asocia al paquete la dirección de origen del emisor —compuesta por la dirección IP del host de origen y por el número de puerto del socket de origen—. Sin embargo, la asociación de la dirección de origen al paquete *no* suele ser realizada por el código de aplicación UDP; en lugar de ello, lo realiza automáticamente el sistema operativo subyacente.

Vamos a utilizar la siguiente aplicación cliente-servidor simple para demostrar cómo programar un socket tanto para UDP como para TCP:

1. El cliente lee una línea de caracteres (datos) de su teclado y envía los datos al servidor.
2. El servidor recibe los datos y convierte los caracteres a mayúsculas.
3. El servidor envía los datos modificados al cliente.
4. El cliente recibe los datos modificados y muestra la línea en su pantalla.

La Figura 2.27 muestra la actividad principal relativa a los sockets del cliente y del servidor que se comunican a través del servicio de transporte UDP.

A continuación proporcionamos la pareja de programas cliente-servidor para una implementación UDP de esta sencilla aplicación. Realizaremos un análisis detallado línea a línea de cada

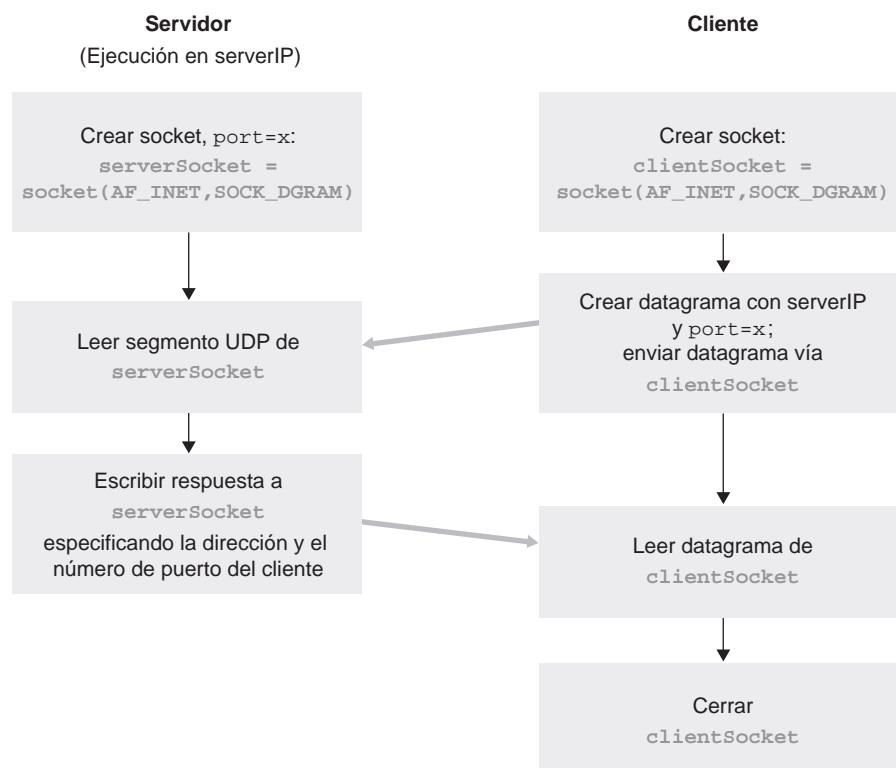


Figura 2.27 ♦ Aplicación cliente-servidor usando UDP.

uno de los programas. Comenzaremos con el cliente UDP, que enviará un mensaje del nivel de aplicación simple al servidor. Con el fin de que el servidor sea capaz de recibir y responder al mensaje del cliente, este debe estar listo y ejecutándose; es decir, debe estar ejecutándose como un proceso antes de que cliente envíe su mensaje.

El nombre del programa cliente es `UDPCliente.py` y el nombre del programa servidor es `UDPServidor.py`. Con el fin de poner el énfasis en las cuestiones fundamentales, hemos proporcionado de manera intencionada código que funciona correctamente pero que es mínimo. Un “código realmente bueno” tendría unas pocas más líneas auxiliares, en concreto aquellas destinadas al tratamiento de errores. Para esta aplicación, hemos seleccionado de forma arbitraria el número de puerto de servidor 12000.

UDPCliente.py

He aquí el código para el lado del cliente de la aplicación:

```
from socket import *
serverName = 'hostname'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Escriba una frase en minúsculas:')
clientSocket.sendto(message.encode(),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(modifiedMessage.decode())
clientSocket.close()
```

Veamos ahora las distintas líneas de código del programa `UDPClient.py`.

```
from socket import *
```

El módulo `socket` constituye la base de todas las comunicaciones de red en Python. Incluyendo esta línea, podemos crear sockets dentro de nuestro programa.

```
serverName = 'hostname'
serverPort = 12000
```

La primera línea define la variable `serverName` como la cadena ‘`hostname`’. Aquí, se proporciona una cadena de caracteres que contiene la dirección IP del servidor (como por ejemplo, “128.138.32.126”) o el nombre de host del servidor (por ejemplo, “`cis.poly.edu`”). Si utilizamos el nombre de host, entonces se llevará a cabo automáticamente una búsqueda DNS para obtener la dirección IP.) La segunda línea asigna el valor 12000 a la variable entera `serverPort`.

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Esta línea crea el socket de cliente denominado `clientSocket`. El primer parámetro indica la familia de direcciones; en particular, `AF_INET` indica que la red subyacente está utilizando IPv4. (No se preocupe en este momento por esto, en el Capítulo 4 abordaremos el tema de IPv4.) El segundo parámetro especifica que el socket es de tipo `SOCK_DGRAM`, lo que significa que se trata de un socket UDP (en lugar de un socket TCP). Observe que no se especifica el número de puerto del socket del cliente al crearlo; en lugar de ello, dejamos al sistema operativo que lo haga por nosotros. Una vez que hemos creado la puerta del proceso del cliente, querremos crear un mensaje para enviarlo a través de la puerta.

```
message = raw_input('Escriba una frase en minúsculas:')
```

`raw_input()` es una función incorporada de Python. Cuando este comando se ejecuta, se solicita al usuario que se encuentra en el cliente que introduzca un texto en minúsculas con la frase “Escriba

una frase en minúsculas:”. El usuario utiliza entonces su teclado para escribir una línea, la cual se guarda en la variable `message`. Ahora que ya tenemos un socket y un mensaje, desearemos enviar el mensaje a través del socket hacia el host de destino.

```
clientSocket.sendto(message.encode(), (serverName, serverPort))
```

En la línea anterior, en primer lugar convertimos el mensaje de tipo cadena a tipo byte, ya que necesitamos enviar bytes por el socket; esto se hace mediante el método `encode()`. El método `sendto()` asocia la dirección de destino (`serverName, serverPort`) al mensaje y envía el paquete resultante por el socket de proceso, `clientSocket`. (Como hemos mencionado anteriormente, la dirección de origen también se asocia al paquete, aunque esto se realiza de forma automática en lugar de explícitamente a través del código.) ¡Enviar un mensaje de un cliente al servidor a través de un socket UDP es así de sencillo! Una vez enviado el paquete, el cliente espera recibir los datos procedentes del servidor.

```
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
```

Con esta línea, cuando un paquete procedente de Internet llega al socket del cliente, los datos del paquete se colocan en la variable `modifiedMessage` (mensaje modificado) y la dirección de origen del paquete se almacena en la variable `serverAddress`. Esta variable contiene tanto la dirección IP del servidor como el número del puerto del mismo. El programa `UDPClient` realmente no necesita esta información de dirección del servidor, puesto que ya la conoce, pero no obstante esta línea de Python proporciona dicha dirección. El método `recvfrom` también especifica el tamaño de buffer de 2048 como entrada. (Este tamaño de buffer es adecuado para prácticamente todos los propósitos.)

```
print(modifiedMessage.decode())
```

Esta línea muestra el mensaje modificado (`modifiedMessage`) en la pantalla del usuario, después de convertir el mensaje en bytes a un mensaje de tipo cadena, que tiene que ser la línea original que escribió el usuario, pero ahora escrito en letras mayúsculas.

```
clientSocket.close()
```

Esta línea cierra el socket y el proceso termina.

UDPServidor.py

Echemos ahora un vistazo al lado del servidor de la aplicación:

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print("El servidor está listo para recibir")
while True:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.decode().upper()
    serverSocket.sendto(modifiedMessage.encode(), clientAddress)
```

Observe que el principio de `UDPServidor` es similar a `UDPClient`. También importa el módulo `socket`, asigna el valor 12000 a la variable entera `serverPort` y crea también un socket de tipo `SOCK_DGRAM` (un socket UDP). La primera línea de código que es significativamente diferente de `UDPClient` es:

```
serverSocket.bind(('', serverPort))
```

La línea anterior asocia (es decir, asigna) el número de puerto 12000 al socket del servidor. Así, en UDPServidor, el código (escrito por el desarrollador de la aplicación) asigna explícitamente un número de puerto al socket. De este modo, cuando alguien envía un paquete al puerto 12000 en la dirección IP del servidor, dicho paquete será dirigido a este socket. A continuación, UDPServidor entra en un bucle `while`; este bucle permite a UDPServidor recibir y procesar paquetes de los clientes de manera indefinida. En el bucle `while`, UDPServidor espera a que llegue un paquete.

```
message, clientAddress = serverSocket.recvfrom(2048)
```

Esta línea de código es similar a la que hemos visto en UDPCliente. Cuando un paquete llega al socket del servidor, los datos del paquete se almacenan en la variable `message` y la dirección de origen del paquete se coloca en la variable `clientAddress`. La variable `clientAddress` contiene tanto la dirección IP del cliente como el número de puerto del cliente. Aquí, UDPServidor *hará uso* de esta información de dirección, puesto que proporciona una dirección de retorno, de forma similar a la dirección del remitente en una carta postal ordinaria. Con esta información de la dirección de origen, el servidor ahora sabe dónde dirigir su respuesta.

```
modifiedMessage = message.decode().upper()
```

Esta línea es la más importante de nuestra sencilla aplicación, ya que toma la línea enviada por el cliente y, después de convertir el mensaje en una cadena, utiliza el método `upper()` para pasarlo a mayúsculas.

```
serverSocket.sendto(modifiedMessage.encode(),clientAddress)
```

Esta última línea asocia la dirección del cliente (dirección IP y número de puerto) al mensaje escrito en mayúsculas (después de convertir la cadena a bytes) y envía el paquete resultante al socket del servidor. (Como hemos mencionado anteriormente, la dirección del servidor también se asocia con el paquete, aunque esto se hace de forma automática en lugar de explícitamente mediante el código.) A continuación, se suministrará el paquete a esa dirección de cliente a través de Internet. Una vez que el servidor envía el paquete, permanece en el bucle `while`, esperando la llegada de otro paquete UDP (de cualquier cliente que se esté ejecutando en cualquier host).

Para probar ambos programas, ejecute `UDPCliente.py` en un host y `UDPServidor.py` en otro host. Asegúrese de incluir el nombre de host o la dirección apropiados del servidor en `UDPCliente.py`. A continuación, ejecute `UDPServidor.py`, el programa del servidor compilado, en el host servidor. De este modo se crea un proceso en el servidor que está a la espera hasta que es contactado por algún cliente. Después, ejecute `UDPCliente.py`, el programa del cliente compilado, en el cliente. Esto crea un proceso en el cliente. Por último, utilice la aplicación en el cliente, escriba una frase seguida de un retorno de carro.

Para crear su propia aplicación cliente-servidor UDP, puede empezar modificando ligeramente los programas de cliente o de servidor. Por ejemplo, en lugar de pasar todas las letras a mayúsculas, el servidor podría contar el número de veces que aparece la letra `s` y devolver dicho número. O puede modificar el cliente de modo que después de recibir la frase en mayúsculas, el usuario pueda continuar enviando más frases al servidor.

2.7.2 Programación de sockets con TCP

A diferencia de UDP, TCP es un protocolo orientado a la conexión. Esto significa que antes de que el cliente y el servidor puedan empezar a enviarse datos entre sí, tienen que seguir un proceso de acuerdo en tres fases y establecer una conexión TCP. Un extremo de la conexión TCP se conecta al socket del cliente y el otro extremo se conecta a un socket de servidor. Cuando creamos la conexión TCP, asociamos con ella la dirección del socket de cliente (dirección IP y número de puerto) y la dirección del socket de servidor (dirección IP y número de puerto). Una vez establecida la conexión

TCP, cuando un lado desea enviar datos al otro lado, basta con colocar los datos en la conexión TCP a través de su socket. Esto es distinto al caso de UDP, en el que el servidor tiene que tener asociada al paquete una dirección de destino antes de colocarlo en el socket.

Ahora, examinemos más en detalle la interacción entre los programas cliente y servidor en TCP. Al cliente le corresponde iniciar el contacto con el servidor. Para que este puede reaccionar al contacto inicial del cliente, tendrá que estar preparado, lo que implica dos cosas. En primer lugar, como en el caso de UDP, el servidor TCP tiene que estar ejecutándose como proceso antes de que el cliente trate de iniciar el contacto. En segundo lugar, el programa servidor debe disponer de algún tipo de puerta especial (o, más precisamente, un socket especial) que acepte algún contacto inicial procedente de un proceso cliente que se esté ejecutando en un host arbitrario. Utilizando nuestra analogía de la vivienda/puerta para un proceso/socket, en ocasiones nos referiremos a este contacto inicial del cliente diciendo que es equivalente a “llamar a la puerta de entrada”.

Con el proceso servidor ejecutándose, el proceso cliente puede iniciar una conexión TCP con el servidor. Esto se hace en el programa cliente creando un socket TCP. Cuando el cliente crea su socket TCP, especifica la dirección del socket de acogida (*wellcoming socket*) en el servidor, es decir, la dirección IP del host servidor y el número de puerto del socket. Una vez creado el socket en el programa cliente, el cliente inicia un proceso de acuerdo en tres fases y establece una conexión TCP con el servidor. El proceso de acuerdo en tres fases, que tiene lugar en la capa de transporte, es completamente transparente para los programas cliente y servidor.

Durante el proceso de acuerdo en tres fases, el proceso cliente llama a la puerta de entrada del proceso servidor. Cuando el servidor “escucha” la llamada, crea una nueva puerta (o de forma más precisa, un *nuevo socket*) que estará dedicado a ese cliente concreto. En el ejemplo que sigue, nuestra puerta de entrada es un objeto socket TCP que denominamos `serverSocket`; el socket que acabamos de crear dedicado al cliente que hace la conexión se denomina `connectionSocket`. Los estudiantes que se topan por primera vez con los sockets TCP confunden en ocasiones el socket de acogida (que es el punto inicial de contacto para todos los clientes que esperan para comunicarse con el servidor) con cada socket de conexión de nueva creación del lado del servidor que se crea posteriormente para comunicarse con cada cliente.

Desde la perspectiva de la aplicación, el socket del cliente y el socket de conexión del servidor están conectados directamente a través de un conducto. Como se muestra en la Figura 2.28, el proceso cliente puede enviar bytes arbitrarios a través de su socket, y TCP garantiza que

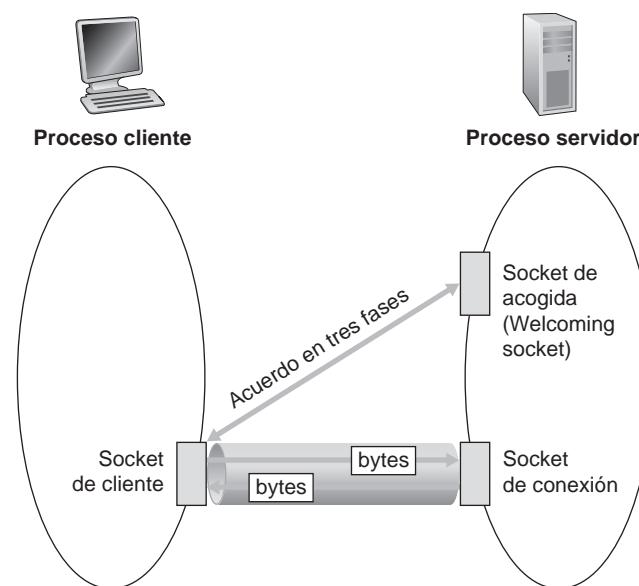


Figura 2.28 ♦ El proceso TCPServidor tiene dos sockets.

el proceso servidor recibirá (a través del socket de conexión) cada byte en el orden en que ha sido enviado. Por tanto, TCP proporciona un servicio fiable entre los procesos cliente y servidor. Además, al igual que las personas pueden entrar y salir a través de una misma puerta, el proceso cliente no sólo envía bytes a través de su socket, sino que también puede recibirlos; de forma similar, el proceso servidor no sólo puede recibir bytes, sino también enviar bytes a través de su socket de conexión.

Vamos a utilizar la misma aplicación cliente-servidor simple para mostrar la programación de sockets con TCP: el cliente envía una línea de datos al servidor, el servidor pone en mayúsculas esa línea y se la devuelve al cliente. En la Figura 2.29 se ha resaltado la actividad principal relativa al socket del cliente y el servidor que se comunican a través del servicio de transporte de TCP.

TCPCliente.py

He aquí el código para el lado del cliente de la aplicación:

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
```

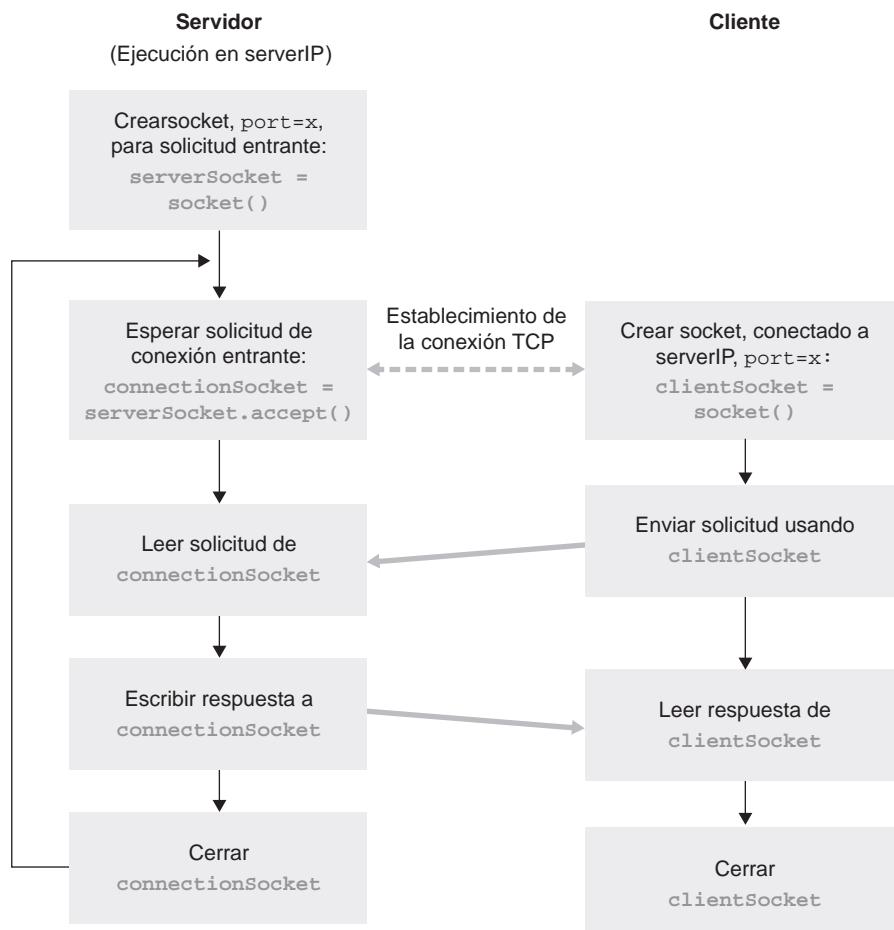


Figura 2.29 ♦ La aplicación cliente-servidor utilizando TCP.

```

sentence = raw_input('Escriba una frase en minúsculas:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print('From Server: ', modifiedSentence.decode())
clientSocket.close()

```

Fijémonos ahora en las líneas del código que difieren significativamente de la implementación para UDP. La primera de estas líneas es la de creación del socket de cliente.

```
clientSocket = socket(AF_INET, SOCK_STREAM)
```

Esta línea crea el socket de cliente, denominado `clientSocket`. De nuevo, el primer parámetro indica que la red subyacente está utilizando IPv4. El segundo parámetro indica que el socket es de tipo `SOCK_STREAM`, lo que significa que se trata de un socket TCP (en lugar de un socket UDP). Observe que de nuevo no especificamos el número de puerto del socket de cliente al crearlo; en lugar de ello, dejamos que sea el sistema operativo el que lo haga por nosotros. La siguiente línea de código es muy diferente de la que hemos visto en `UDPClient`:

```
clientSocket.connect((serverName, serverPort))
```

Recuerde que antes de que el cliente pueda enviar datos al servidor (o viceversa) empleando un socket TCP, debe establecerse primero una conexión TCP entre el cliente y el servidor. La línea anterior inicia la conexión TCP entre el cliente y el servidor. El parámetro del método `connect()` es la dirección del lado de servidor de la conexión. Después de ejecutarse esta línea, se lleva a cabo el proceso de acuerdo en tres fases y se establece una conexión TCP entre el cliente y el servidor.

```
sentence = raw_input('Escriba una frase en minúsculas:')
```

Como con `UDPClient`, la línea anterior obtiene una frase del usuario. La cadena `sentence` recopila los caracteres hasta que el usuario termina la línea con un retorno de carro. La siguiente línea de código también es muy diferente a la utilizada en `UDPClient`:

```
clientSocket.send(sentence.encode())
```

La línea anterior envía la cadena `sentence` a través del socket de cliente y la conexión TCP. Observe que el programa *no* crea explícitamente un paquete y asocia la dirección de destino al paquete, como sucedía en el caso de los sockets UDP. En su lugar, el programa cliente simplemente coloca los bytes de la cadena `sentence` en la conexión TCP. El cliente espera entonces a recibir los bytes procedentes del servidor.

```
modifiedSentence = clientSocket.recv(2048)
```

Cuando llegan los caracteres de servidor, estos se colocan en la cadena `modifiedSentence`. Los caracteres continúan acumulándose en `modifiedSentence` hasta que la línea termina con un carácter de retorno de carro. Después de mostrar la frase en mayúsculas, se cierra el socket de cliente:

```
clientSocket.close()
```

Esta última línea cierra el socket y, por tanto, la conexión TCP entre el cliente y el servidor. Esto hace que TCP en el cliente envíe un mensaje TCP al proceso TCP del servidor (véase la Sección 3.5).

TCPServidor.py

Veamos ahora el programa del servidor.

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('El servidor está listo para recibir')
while True:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.encode())
    connectionSocket.close()
```

Estudiemos ahora las líneas que difieren significativamente en UDPServidor y TCPCliente. Como en el caso de TCPCliente, el servidor crea un socket TCP con:

```
serverSocket=socket(AF_INET,SOCK_STREAM)
```

De forma similar a UDPServidor, asociamos el número de puerto de servidor, `serverPort`, con este socket:

```
serverSocket.bind(('',serverPort))
```

Pero con TCP, `serverSocket` será nuestro socket de acogida. Después de establecer esta puerta de entrada, esperaremos hasta escuchar que algún cliente llama a la puerta:

```
serverSocket.listen(1)
```

Esta línea hace que el servidor esté a la escucha de solicitudes de conexión TCP del cliente. El parámetro especifica el número máximo de conexiones en cola (al menos, 1).

```
connectionSocket, addr = serverSocket.accept()
```

Cuando un cliente llama a esta puerta, el programa invoca el método `accept()` para el `serverSocket`, el cual crea un nuevo socket en el servidor, denominado `connectionSocket`, dedicado a este cliente concreto. El cliente y el servidor completan entonces el acuerdo en tres fases, creando una conexión TCP entre el socket `clientSocket` del cliente y el socket `connectionSocket` del servidor. Con la conexión TCP establecida, el cliente y el servidor ahora pueden enviarse bytes entre sí a través de la misma. Con TCP, no solo está garantizado que todos los bytes enviados desde un lado llegan al otro lado, sino que también queda garantizado que llegarán en orden.

```
connectionSocket.close()
```

En este programa, después de enviar la frase modificada al cliente, se cierra el socket de conexión. Pero puesto que `serverSocket` permanece abierto, otro cliente puede llamar a la puerta y enviar una frase al servidor para su modificación.

Esto completa nuestra exposición acerca de la programación de sockets en TCP. Le animamos a que ejecute los dos programas en dos hosts distintos y a que los modifique para obtener objetivos ligeramente diferentes. Debería comparar la pareja de programas para UDP con los programas para TCP y ver en qué se diferencian. También le aconsejamos que realice los ejercicios sobre programación de sockets descritos al final de los Capítulos 2, 4 y 9. Por último, esperamos que algún día, después de dominar estos y otros programas de sockets más complejos, escriba sus propia aplicación de red popular, se haga muy rico y famoso, y recuerde a los autores de este libro de texto.

2.8 Resumen

En este capítulo hemos estudiado los aspectos conceptuales y de implementación de las aplicaciones de red. Hemos podido comprobar la omnipresencia de la arquitectura cliente-servidor adoptada por muchas aplicaciones de Internet y ver su uso en los protocolos HTTP, SMTP, POP3 y DNS. Hemos estudiado estos importantes protocolos del nivel de aplicación y sus correspondientes aplicaciones asociadas (la Web, la transferencia de archivos, el correo electrónico y DNS) con cierto detalle. También hemos aprendido acerca de la arquitectura P2P y cómo se utiliza en muchas aplicaciones. También abordaremos los flujos de vídeo y cómo los modernos sistemas de distribución aprovechan las redes CDN. Hemos examinado cómo puede utilizarse la API de sockets para crear aplicaciones de red. Asimismo, hemos estudiado el uso de los sockets para los servicios de transporte terminal a terminal orientados a la conexión (TCP) y sin conexión (UDP). ¡Hemos completado la primera etapa de nuestro viaje por la arquitectura de red en capas!

Al principio del libro, en la Sección 1.1, hemos proporcionado una definición algo vaga de protocolo: “el formato y el orden de los mensajes intercambiados entre dos o más entidades que se comunican, así como las acciones realizadas en la transmisión y/o recepción de un mensaje u otro evento.” El material facilitado en este capítulo, y en concreto el estudio detallado de los protocolos HTTP, SMTP, POP3 y DNS, aporta a esta definición una gran profundidad. Los protocolos son un concepto clave en las redes y nuestro estudio de los protocolos de aplicación nos ha proporcionado la oportunidad de desarrollar una idea más intuitiva sobre qué son los protocolos.

En la Sección 2.1 hemos descrito los modelos de servicio que ofrecen TCP y UDP a las aplicaciones que los invocan. En las Secciones 2.7 y 2.8 hemos visto en detalle estos modelos de servicio para el desarrollo de aplicaciones sencillas que se ejecutan sobre TCP y UDP. Sin embargo, hemos hablado poco acerca de cómo TCP y UDP proporcionan estos modelos de servicio. Por ejemplo, sabemos que TCP proporciona un servicio de datos fiable, pero todavía no sabemos cómo lo hace. En el siguiente capítulo veremos detenidamente no solo qué servicios proporcionan, sino también el cómo y el por qué del funcionamiento de los protocolos de transporte.

Ahora que ya tenemos algunos conocimientos acerca de la estructura de las aplicaciones de Internet y de los protocolos de la capa de aplicación, estamos preparados para seguir descendiendo por la pila de protocolos y examinar la capa de transporte en el Capítulo 3.

Problemas y cuestiones de repaso

Capítulo 2 Cuestiones de repaso

SECCIÓN 2.1

- R1. Enumere cinco aplicaciones de Internet no propietarias y los protocolos de la capa de aplicación que utilizan.
- R2. ¿Cuál es la diferencia entre la arquitectura de red y la arquitectura de aplicación?
- R3. En una sesión de comunicación entre dos procesos, ¿qué proceso es el cliente y qué proceso es el servidor?
- R4. En una aplicación de compartición de archivos P2P, ¿está de acuerdo con la siguiente afirmación: “No existen los lados de cliente y de servidor en una sesión de comunicación”? ¿Por qué?
- R5. ¿Qué información utiliza un proceso que se ejecuta en un host para identificar a un proceso que se ejecuta en otro host?
- R6. Suponga que desea realizar una transición desde un cliente remoto a un servidor lo más rápidamente posible. ¿Qué utilizaría, UDP o TCP? ¿Por qué?
- R7. Utilizando la Figura 2.4, podemos ver que ninguna de las aplicaciones indicadas en dicha figura presenta a la vez requisitos de temporización y de ausencia de pérdida de datos. ¿Puede

concebir una aplicación que requiera que no haya pérdida de datos y que también sea extremadamente sensible al tiempo?

- R8. Enumere las cuatro clases principales de servicios que puede proporcionar un protocolo de transporte. Para cada una de las clases de servicios, indique si UDP o TCP (o ambos) proporcionan un servicio así.
- R9. Recuerde que TCP puede mejorarse con SSL para proporcionar servicios de seguridad proceso a proceso, incluyendo mecanismos de cifrado. ¿En qué capa opera SSL, en la capa de transporte o en la capa de aplicación? Si el desarrollador de la aplicación desea mejorar TCP con SSL, ¿qué tendrá que hacer?

SECCIÓN 2.2-2.5

- R10. ¿Qué quiere decir el término protocolo de acuerdo?
- R11. ¿Por qué HTTP, SMTP y POP3 se ejecutan sobre TCP en lugar de sobre UDP?
- R12. Un sitio de comercio electrónico desea mantener un registro de compras para cada uno de sus clientes. Describa cómo se puede hacer esto utilizando cookies.
- R13. Describa cómo el almacenamiento en caché web puede reducir el retardo de recepción de un objeto solicitado. ¿Reducirá este tipo de almacenamiento el retardo de todos los objetos solicitados por el usuario o sólo el de algunos objetos? ¿Por qué?
- R14. Establezca una sesión Telnet en un servidor web y envíe un mensaje de solicitud de varias líneas. Incluya en dicho mensaje la línea de cabecera `If-modified-since:` para forzar un mensaje de respuesta con el código de estado `304 Not Modified`.
- R15. Enumere algunas aplicaciones de mensajería populares. ¿Utilizan el mismo protocolo que SMS?
- R16. Suponga que Alicia, que dispone de una cuenta de correo electrónico web (como por ejemplo Hotmail o gmail), envía un mensaje a Benito, que accede a su correo almacenado en su servidor de correo utilizando POP3. Explique cómo se transmite el mensaje desde el host de Alicia hasta el de Benito. Asegúrese de citar la serie de protocolos de la capa de aplicación que se utilizan para llevar el mensaje de un host al otro.
- R17. Imprima la cabecera de un mensaje de correo electrónico que haya recibido recientemente. ¿Cuántas líneas de cabecera `Received:` contiene? Analice cada una de las líneas de cabecera del mensaje.
- R18. Desde la perspectiva de un usuario, ¿cuál es la diferencia entre el modo “descargar y borrar” y el modo “descargar y mantener” en POP3?
- R19. ¿Pueden el servidor web y el servidor de correo electrónico de una organización tener exactamente el mismo alias para un nombre de host (por ejemplo, `foo.com`)? ¿Cuál sería el tipo especificado en el registro de recurso (RR) que contiene el nombre de host del servidor de correo?
- R20. Estudie sus mensajes de correo electrónico recibidos y examine la cabecera de un mensaje enviado desde un usuario con una dirección de correo electrónico `.edu`. ¿Es posible determinar a partir de la cabecera la dirección IP del host desde el que se envió el mensaje? Repita el proceso para un mensaje enviado desde una cuenta de Gmail.

SECCIÓN 2.5

- R21. En BitTorrent, suponga que Alicia proporciona fragmentos a Benito a intervalos de 30 segundos. ¿Devolverá necesariamente Benito el favor y proporcionará fragmentos a Alicia en el mismo intervalo de tiempo? ¿Por qué?
- R22. Suponga que un nuevo par Alicia se une a BitTorrent sin tener en su posesión ningún fragmento. Dado que no posee fragmentos, no puede convertirse en uno de los cuatro principales suministros.

tradores de ninguno de los otros pares, ya que no tiene nada que suministrar. ¿Cómo obtendrá entonces Alicia su primer fragmento?

R23. ¿Qué es una red solapada? ¿Contiene routers? ¿Cuáles son las fronteras en una red solapada?

SECCIÓN 2.6

R24. Normalmente, las redes CDN adoptan una de dos filosofías diferentes de ubicación de los servidores. Nómbrelas y descríbalas brevemente.

R25. Además de las consideraciones relativas a las redes como son los retardos, las pérdidas de paquetes y el ancho de banda, existen otros factores importantes que se deben tener en cuenta a la hora de diseñar una estrategia de selección del servidor CDN. ¿Cuáles son esos factores?

SECCIÓN 2.7

R26. El servidor UDP descrito en la Sección 2.7 solo necesitaba un socket, mientras que el servidor TCP necesitaba dos. ¿Por qué? Si el servidor TCP tuviera que soportar n conexiones simultáneas, cada una procedente de un host cliente distinto, ¿cuántos sockets necesitaría el servidor TCP?

R27. En la aplicación cliente-servidor sobre TCP descrita en la Sección 2.7, ¿por qué tiene que ser ejecutado el programa servidor antes que el programa cliente? En la aplicación cliente-servidor sobre UDP, ¿por qué el programa cliente puede ejecutarse antes que el programa servidor?

Problemas

P1. ¿Verdadero o falso?

- Un usuario solicita una página web que consta de texto y tres imágenes. Para obtener esa página, el cliente envía un mensaje de solicitud y recibe cuatro mensajes de respuesta.
- Dos páginas web diferentes (por ejemplo, www.mit.edu/research.html y www.mit.edu/students.html) se pueden enviar a través de la misma conexión persistente.
- Con las conexiones no persistentes entre un navegador y un servidor de origen, un único segmento TCP puede transportar dos mensajes de solicitud HTTP distintos.
- La línea de cabecera `Date:` del mensaje de respuesta HTTP indica cuándo el objeto fue modificado por última vez.
- Los mensajes de respuesta HTTP nunca incluyen un cuerpo de mensaje vacío.

P2. SMS, iMessage y WhatsApp son todos ellos sistemas de mensajería en tiempo real para smartphone. Después de llevar a cabo una pequeña investigación en Internet, escriba un párrafo indicando los protocolos que cada uno de estos sistemas emplea. A continuación, escriba un párrafo explicando en qué se diferencian.

P3. Un cliente HTTP desea recuperar un documento web que se encuentra en un URL dado. Inicialmente, la dirección IP del servidor HTTP es desconocida. ¿Qué protocolos de la capa de aplicación y de la capa de transporte además de HTTP son necesarios en este escenario?

P4. La siguiente cadena de caracteres ASCII ha sido capturada por Wireshark cuando el navegador enviaba un mensaje GET HTTP (es decir, este es el contenido real de un mensaje GET HTTP). Los caracteres `<cr><lf>` representan el retorno de carro y el salto de línea (es decir, la cadena de caracteres en cursiva `<cr>` del texto que sigue a este párrafo representa el carácter de retorno de carro contenido en dicho punto de la cabecera HTTP). Responda a las siguientes cuestiones, indicando en qué parte del siguiente mensaje GET HTTP se encuentra la respuesta.

```
GET /cs453/index.html HTTP/1.1<cr><lf>Host: gai
```

```
a.cs.umass.edu<cr><lf>User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.2) Gecko/20040804 Netscape/7.2 (ax) <cr><lf>Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9, text/plain;q=0.8, image/png,*/*;q=0.5<cr><lf>Accept-Language: en-us,en;q=0.5<cr><lf>Accept-Encoding: zip,deflate<cr><lf>Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7<cr><lf>Keep-Alive: 300<cr><lf>Connection:keep-alive<cr><lf><cr><lf>
```

- a. ¿Cuál es el URL del documento solicitado por el navegador?
 - b. ¿Qué versión de HTTP se está ejecutando en el navegador?
 - c. ¿Qué tipo de conexión solicita el navegador, persistente o no persistente?
 - d. ¿Cuál es la dirección IP del host en el que se está ejecutando el navegador?
 - e. ¿Qué tipo de navegador inicia este mensaje? ¿Por qué es necesario indicar el tipo de navegador en un mensaje de solicitud HTTP?
- P5. El siguiente texto muestra la respuesta devuelta por el servidor al mensaje de solicitud GET HTTP del problema anterior. Responda a las siguientes cuestiones, indicando en qué parte del siguiente mensaje se encuentran las respuestas.

```
HTTP/1.1 200 OK<cr><lf>Date: Tue, 07 Mar 2008  
12:39:45GMT<cr><lf>Server: Apache/2.0.52 (Fedora)  
<cr><lf>Last-Modified: Sat, 10 Dec 2005 18:27:46 GMT<cr><lf>ETag:  
"526c3-f22-a88a4c80"<cr><lf>Accept-  
Ranges: bytes<cr><lf>Content-Length: 3874<cr><lf>  
Keep-Alive: timeout=max=100<cr><lf>Connection:  
Keep-Alive<cr><lf>Content-Type: text/html; charset=  
ISO-8859-1<cr><lf><cr><lf><!doctype html public "-//  
//w3c//dtd html 4.0transitional//en"><lf><html><lf> <head><lf>  
<meta http-equiv="Content-Type"  
content="text/html; charset=iso-8859-1"><lf> <meta  
name="GENERATOR" content="Mozilla/4.79 [en] (Windows NT  
5.0; U) Netscape!><lf> <title>CMPSCI 453 / 591 /  
NTU-ST550ASpring 2005 homepage</title><lf></head><lf>  
<aquí continúa el texto del documento (no mostrado)>
```

- a. ¿Ha podido el servidor encontrar el documento? ¿En qué momento se suministró la respuesta con el documento?
 - b. ¿Cuándo fue modificado por última vez el documento?
 - c. ¿Cuántos bytes contiene el documento devuelto?
 - d. ¿Cuáles son los primeros cinco bytes del documento que se está devolviendo? ¿Ha acordado el servidor emplear una conexión persistente?
- P6. Utilice la especificación HTTP/1.1 (RFC 2616) para responder a las siguientes cuestiones:
- a. Explique el mecanismo de señalización entre el cliente y el servidor para indicar que se está cerrando una conexión persistente. ¿Quién puede señalizar el cierre de la conexión, el cliente, el servidor o ambos?
 - b. ¿Qué servicios de cifrado proporciona HTTP?
 - c. ¿Puede un cliente abrir tres o más conexiones simultáneas con un determinado servidor?
 - d. Un servidor o un cliente pueden cerrar una conexión de transporte entre ellos si uno detecta que la conexión ha estado inactiva durante un cierto tiempo. ¿Es posible que un lado inicie

el cierre de una conexión mientras que el otro lado está transmitiendo datos a través de dicha conexión? Explique su respuesta.

- P7. Suponga que en su navegador hace clic en un vínculo a una página web. La dirección IP correspondiente al URL asociado no está almacenado en la caché de su host local, por lo que es necesario realizar una búsqueda DNS para obtener la dirección IP. Suponga que antes de que su host reciba la dirección IP de DNS se han visitado n servidores DNS y que los tiempos de ida y vuelta (RTT) de las sucesivas visitas son RTT_1, \dots, RTT_n . Suponga además que la página web asociada con el vínculo contiene exactamente un objeto, que consta de un pequeño fragmento de texto HTML. Sea RTT_0 el tiempo RTT entre el host local y el servidor que contiene el objeto. Suponiendo un tiempo de transmisión de cero para el objeto, ¿cuánto tiempo transcurre desde que el cliente hace clic en el vínculo hasta que recibe el objeto?
- P8. Continuando con el Problema P7, suponga que el archivo HTML hace referencia a ocho objetos muy pequeños que se encuentran en el mismo servidor. Despreciando los tiempos de transmisión, ¿cuánto tiempo transcurre si se utiliza
- HTTP no persistente sin conexiones TCP en paralelo?
 - HTTP no persistente con el navegador configurado para 5 conexiones paralelo?
 - HTTP persistente?
- P9. En la red institucional conectada a Internet de la Figura 2.12, suponga que el tamaño medio de objeto es de 850.000 bits y que la tasa media de solicitudes de los navegadores de la institución a los servidores de origen es de 16 solicitudes por segundo. Suponga también que el tiempo que se tarda desde que el router en el lado de Internet del enlace de acceso reenvía una solicitud HTTP hasta que recibe la respuesta es, como media, de tres segundos (véase la Sección 2.2.5). Modele el tiempo medio de respuesta total como la suma del retardo medio de acceso (es decir, el retardo desde el router de Internet al router de la institución) y el retardo medio de Internet. Para el retardo medio de acceso, utilice la expresión $\Delta/(1 - \Delta\beta)$, donde Δ es el tiempo medio requerido para enviar un objeto a través del enlace de acceso y β es la tasa de llegada de los objetos al enlace de acceso.
- Calcule el tiempo medio de respuesta total.
 - Ahora suponga que hay instalada una caché en la LAN institucional. Suponga que la tasa de fallos es de 0,4. Calcule el tiempo de respuesta total.
- P10. Dispone de un enlace corto de 10 metros a través del cual un emisor puede transmitir a una velocidad de 150 bits/segundo en ambos sentidos. Suponga que los paquetes de datos tienen una longitud de 100.000 bits y los paquetes que contienen solo comandos de control (por ejemplo, ACK o de acuerdo) tienen una longitud de 200 bits. Suponga que hay N conexiones en paralelo y que cada una utiliza $1/N$ del ancho de banda del enlace. Considere ahora el protocolo HTTP y suponga que cada objeto descargado es de 100 kbits de largo y que el objeto inicialmente descargado contiene 10 objetos referenciados procedentes del mismo emisor. ¿Tiene sentido en este caso realizar descargas en paralelo mediante instancias paralelas de HTTP no persistente? Considere ahora HTTP persistente. ¿Cabe esperar alguna ventaja significativa respecto del caso no persistente? Justifique y explique su respuesta.
- P11. Continuando con el escenario del problema anterior, suponga que Benito comparte el enlace con otros cuatro usuarios. Benito utiliza instancias paralelas de HTTP no persistente y los otros cuatro usuarios utilizan HTTP no persistente sin descargas en paralelo.
- ¿Le ayudan a Benito las conexiones en paralelo a obtener las páginas más rápidamente? ¿Por qué?
 - Si los cinco usuarios abren cinco instancias paralelas de HTTP no persistente, ¿seguirán siendo beneficiosas las conexiones en paralelo de Benito? ¿Por qué?
- P12. Escriba un programa TCP simple para un servidor que acepte líneas de entrada procedentes de un cliente y muestre dichas líneas en la salida estándar del servidor. (Puede realizar esta tarea

modificando el programa TCPServer.py visto en el capítulo.) Compile y ejecute su programa. En cualquier otra máquina que disponga de un navegador web, configure el servidor proxy en el navegador para que apunte al host que está ejecutando su programa servidor; configure también el número de puerto de la forma apropiada. Su navegador deberá ahora enviar sus mensajes de solicitud GET a su servidor y el servidor tendrá que mostrar dichos mensajes en su salida estándar. Utilice esta plataforma para determinar si su navegador genera mensajes GET condicionales para los objetos almacenados localmente en la caché.

- P13. ¿Cuál es la diferencia entre `MAIL FROM:` en SMTP y `From:` en el propio mensaje de correo?
- P14. ¿Cómo marca SMTP el final del cuerpo de un mensaje? ¿Cómo lo hace HTTP? ¿Puede HTTP utilizar el mismo método que SMTP para marcar el final del cuerpo de un mensaje? Explique su respuesta.
- P15. Lea el documento RFC 5321 dedicado a SMTP. ¿Qué quiere decir MTA? Considere el siguiente mensaje de correo basura recibido (modificado a partir de un correo basura real). Suponiendo que únicamente el remitente de este mensaje de correo es malicioso y que los demás hosts son honestos, identifique al host malicioso que ha generado este correo basura.

```
From - Fri Nov 07 13:41:30 2008
Return-Path: <tennis5@pp33head.com>
Received: from barmail.cs.umass.edu (barmail.cs.umass.edu
[128.119.240.3]) by cs.umass.edu (8.13.1/8.12.6) for
<hg@cs.umass.edu>; Fri, 7 Nov 2008 13:27:10 -0500
Received: from asusus-4b96 (localhost [127.0.0.1]) by
barmail.cs.umass.edu (Spam Firewall) for <hg@cs.umass.edu>; Fri, 7
Nov 2008 13:27:07 -0500 (EST)
Received: from asusus-4b96 ([58.88.21.177]) by barmail.cs.umass.edu
for <hg@cs.umass.edu>; Fri, 07 Nov 2008 13:27:07 -0500 (EST)
Received: from [58.88.21.177] by inbnd55.exchangedddd.com; Sat, 8
Nov 2008 01:27:07 +0700
From: "Jonny" <tennis5@pp33head.com>
To: <hg@cs.umass.edu>

Subject: Cómo asegurar sus ahorros
```

- P16. Lea el documento RFC 1939 dedicado a POP3. ¿Cuál es el propósito del comando UIDL POP3?
- P17. Imagine que accede a su correo electrónico utilizando POP3.
 - a. Suponga que ha configurado su cliente de correo POP para operar en el modo descargar y borrar. Complete la siguiente transacción:

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: bla bla ...
S: ....bla
S: .
?
?
```

- b. Suponga que ha configurado su cliente de correo POP para operar en el modo descargar y guardar. Complete la siguiente transacción:

```

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: bla bla ...
S: .....bla
S: .
?
?
```

- c. Suponga que ha configurado su cliente de correo POP para operar en el modo descargar y guardar. Utilizando su transcripción del apartado (b), suponga que recupera los mensajes 1 y 2, sale de POP, y cinco minutos más tarde vuelve a acceder otra vez a POP para recuperar un nuevo mensaje de correo. Suponga que en ese intervalo de cinco minutos nadie le ha enviado un nuevo mensaje de correo. Proporcione una transcripción de esta segunda sesión de POP.

- P18. a. ¿Qué es una base de datos *whois*?
- Utilice varias bases de datos *whois* de Internet para obtener los nombres de dos servidores DNS. Indique qué bases de datos *whois* ha utilizado.
 - Utilice el programa *nslookup* en su host local para enviar consultas DNS a tres servidores DNS: su servidor DNS local y los dos servidores DNS que haya encontrado en el apartado (b). Intente realizar consultas para obtener registros de recursos de tipo A, NS y MX. Escriba un resumen de sus hallazgos.
 - Utilice el programa *nslookup* para localizar un servidor web que tenga varias direcciones IP. ¿Tiene el servidor web de su institución (centro de estudios o empresa) varias direcciones IP?
 - Utilice la base de datos *whois ARIN* para determinar el rango de direcciones IP utilizado en su universidad.
 - Describa cómo un atacante puede utilizar las bases de datos *whois* y la herramienta *nslookup* para realizar labores de reconocimiento en una institución antes de lanzar un ataque.
 - Explique por qué las bases de datos *whois* deben estar disponibles públicamente.
- P19. En este problema empleará la útil herramienta *dig*, disponible en los hosts Unix y Linux para explorar la jerarquía de los servidores DNS. Como se muestra en la Figura 2.19, un servidor DNS que se encuentra en un nivel superior de la jerarquía DNS delega una consulta DNS en un servidor DNS que se encuentra en un nivel más bajo de la jerarquía, devolviendo al cliente DNS el nombre de dicho servidor DNS del nivel más bajo. Lea primero la página de manual dedicada a *dig* y, a continuación, responda a las siguientes preguntas.
- Comenzando por un servidor DNS raíz (uno de los servidores raíz [a-m].root-servers.net), inicie una secuencia de consultas para obtener la dirección IP del servidor web de su departamento, utilizando la herramienta *dig*. Visualice la lista de nombres de los servidores DNS incluidos en la cadena de delegación que ha obtenido como respuesta a su consulta.
 - Repita el apartado (a) para varios sitios web populares, como por ejemplo google.com, yahoo.com o amazon.com.
- P20. Suponga que puede acceder a las cachés de los servidores DNS locales de su departamento. ¿Puede proponer una forma de determinar de manera aproximada los servidores web (situados fuera de su departamento) que son más populares entre los usuarios del departamento? Explique su respuesta.

- P21. Suponga que su departamento tiene un servidor DNS local para todas las computadoras del departamento. Usted es un usuario normal (es decir, no es un administrador de la red o del sistema). ¿Puede determinar de alguna manera si desde alguna computadora de su departamento se ha accedido hace unos pocos segundos a un determinado sitio web externo? Explique su respuesta.
- P22. Desea distribuir un archivo de $F = 15$ Gbits a N pares. El servidor tiene una velocidad de carga de $u_s = 30$ Mbps, y cada par tiene una velocidad de descarga de $d_i = 2$ Mbps y una velocidad de carga igual a u . Para $N = 10, 100$ y 1.000 , y $u = 300$ kbps, 700 kbps y 2 Mbps, prepare una gráfica que proporcione el tiempo mínimo de distribución para cada una de las combinaciones de N y u , tanto para una distribución cliente-servidor como para una distribución P2P.
- P23. Desea distribuir un archivo de F bits a N pares utilizando una arquitectura cliente-servidor. Suponga un modelo flexible, en el que el servidor puede transmitir simultáneamente a varios pares, transmitiendo a cada par a distintas velocidades, siempre y cuando la velocidad combinada no sea mayor que u_s .
- Suponga que $u_s/N \leq d_{\min}$. Especifique un esquema de distribución que tenga un tiempo de distribución de NF/u_s .
 - Suponga que $u_s/N \geq d_{\min}$. Especifique un esquema de distribución que tenga un tiempo de distribución de F/d_{\min} .
 - Demuestre que, en general, el tiempo mínimo de distribución está dado por $\max\{NF/u_s, F/d_{\min}\}$.
- P24. Desea distribuir un archivo de F bits a N pares utilizando una arquitectura P2P. Suponga un modelo flexible. Con el fin de simplificar, suponga que d_{\min} es muy grande, por lo que el ancho de banda de descarga de los pares no es nunca un cuello de botella.
- Suponga que $u_s \leq (u_s + u_1 + \dots + u_N)/N$. Especifique un esquema de distribución que tenga un tiempo de distribución de F/u_s .
 - Suponga que $u_s \geq (u_s + u_1 + \dots + u_N)/N$. Especifique un esquema de distribución que tenga un tiempo de distribución de $NF/(u_s + u_1 + \dots + u_N)$.
 - Demuestre que, en general, el tiempo mínimo de distribución está dado por $\max\{F/u_s, NF/(u_s + u_1 + \dots + u_N)\}$.
- P25. Considere una red solapada con N pares activos, disponiendo cada pareja de pares de una conexión TCP activa. Suponga también que las conexiones TCP atraviesan un total de M routers. ¿Cuántos nodos y fronteras existen en la correspondiente red solapada?
- P26. Suponga que Benito se une a un torrente BitTorrent, pero no desea suministrar datos a otros pares (lo que se denomina “ir por libre”).
- Benito afirma que puede recibir una copia completa del archivo compartido por el conjunto de usuarios. ¿Es correcto lo que dice Benito? ¿Por qué?
 - Benito añade que puede hacer más eficientes sus descargas utilizando varias computadoras (con distintas direcciones IP) del laboratorio de su departamento. ¿Cómo puede hacer esto?
- P27. Considere un sistema DASH para el que hay N versiones de un vídeo (con N diferentes velocidades y niveles de calidad) y N versiones de audio (con N diferentes velocidades y niveles de calidad). Suponga que queremos permitir que el reproductor seleccione en cualquier instante cualquiera de las N versiones de vídeo y de las N versiones de audio.
- Si creamos archivos de modo que el audio esté mezclado con el vídeo, y el servidor envíe solo un flujo multimedia en cualquier momento dado, ¿cuántos archivos necesitará almacenar el servidor (cada uno con un URL distinto)?
 - Si, por el contrario, el servidor envía los flujos de vídeo y de audio por separado y hacemos que el cliente sincronice los flujos, ¿cuántos archivos necesitará almacenar el servidor?

P28. Instale y compile los programas Python TCPCliente y UDPCliente en un host y TCPServidor y UDPServidor en otro host.

a. Suponga que ejecuta TCPCliente antes que TCPServidor. ¿Qué ocurre? ¿Por qué?

b. Suponga que ejecuta UDPCliente antes que UDPServidor. ¿Qué ocurre? ¿Por qué?

c. ¿Qué ocurre si se utilizan diferentes números de puerto para los lados cliente y servidor?

P29. Suponga que en UDPCliente.py, después de crear el socket, añadimos esta línea:

```
clientSocket.bind(('', 5432))
```

¿Será necesario modificar el programa UDPServidor.py? ¿Cuáles son los números de puerto para los sockets en UDPCliente y UDPServidor? ¿Cuáles eran antes de realizar este cambio?

P30. ¿Puede configurar su navegador para abrir múltiples conexiones simultáneas a un sitio web?

¿Cuáles son las ventajas y desventajas de disponer de un gran número de conexiones TCP simultáneas?

P31. Hemos visto que los sockets TCP de Internet tratan los datos que están siendo enviados como un flujo de bytes, pero que los sockets UDP reconocen las fronteras entre mensajes. Cite una ventaja y una desventaja de la API orientada a bytes, con respecto a la API que reconoce y preserva explícitamente las fronteras entre mensajes definidas por la aplicación.

P32. ¿Qué es el servidor web Apache? ¿Cuánto cuesta? ¿Cuál es la funcionalidad que tiene en la actualidad? Puede consultar la Wikipedia para responder a esta pregunta.

Tareas sobre programación de sockets

El sitio web de acompañamiento incluye seis tareas de programación de sockets. Las cuatro primeras se resumen a continuación. La quinta tarea utiliza el protocolo ICMP y se resume al final del Capítulo 5. La sexta tarea emplea protocolos multimedia y se resume al final del Capítulo 9. Recomendamos fuertemente a los estudiantes que completen algunas, si no todas, de estas tareas. Los estudiantes pueden encontrar la información completa acerca de estas tareas, así como importantes *snippets* del código Python, en el sitio web www.pearsonhighered.com/cs-resources.

Tarea 1: servidor web

En esta tarea, tendrá que desarrollar un servidor web simple en Python que sea capaz de procesar una única solicitud. Específicamente, el servidor web deberá (i) crear un socket de conexión al ser contactado por un cliente (navegador); (ii) recibir la solicitud HTTP a través de dicha conexión; (iii) analizar sintácticamente la solicitud para determinar qué archivo concreto se está solicitando; (iv) obtener el archivo solicitado del sistema de archivos del servidor; (v) crear un mensaje de respuesta HTTP consistente en el archivo solicitado, precedido por una serie de líneas de cabecera, y (vi) enviar la respuesta al navegador solicitante a través de la conexión TCP. Si un navegador solicita un archivo que no esté presente en su servidor, el servidor web debe devolver un mensaje de error “404 Not Found”.

En el sitio web de acompañamiento proporcionamos el esqueleto de código para el servidor web. Su tarea consiste en completar el código, ejecutar el servidor y luego probarlo, enviándole solicitudes desde navegadores que se ejecuten en diferentes hosts. Si ejecuta su servidor web en un host que ya tenga otro servidor web funcionando, deberá utilizar para su servidor web un puerto distinto del puerto 80.

Tarea 2: programa ping UDP

En esta tarea de programación, tendrá que escribir un programa cliente ping en Python. El cliente deberá enviar un simple mensaje ping a un servidor, recibir un mensaje pong correspondiente de

respuesta del servidor y determinar el retardo existente entre el instante en que el cliente envió el mensaje ping y el instante en que recibió el mensaje pong. Este retardo se denomina RTT (*Round Trip Time*, tiempo de ida y vuelta). La funcionalidad determinada por el cliente y el servidor es similar a la que ofrece el programa ping estándar disponible en los sistemas operativos modernos. Sin embargo, los programas ping estándar utilizan el protocolo ICMP (*Internet Control Message Protocol*, protocolo de mensajes de control de Internet), del cual hablaremos en el Capítulo 5. Aquí crearemos un programa ping no estándar (¡pero sencillo!) basado en UDP.

Su programa ping deberá enviar 10 mensajes ping al servidor objetivo a través de UDP. Para cada mensaje, el cliente debe determinar e imprimir el RTT cuando reciba el correspondiente mensaje pong. Como UDP es un protocolo no fiable, puede perderse algún paquete enviado por el cliente o por el servidor. Por esta razón, el cliente no puede quedarse esperando indefinidamente a recibir una respuesta a un mensaje ping. Debe hacer que el cliente espere un máximo de un segundo a recibir una respuesta del servidor; si no se recibe respuesta, el cliente debe asumir que el paquete se perdió e imprimir un mensaje informando de tal hecho.

En esta tarea, le proporcionamos el código completo del servidor (disponible en el sitio web de acompañamiento). Su trabajo consiste en escribir el código del cliente, que será muy similar al código del servidor. Le recomendamos que primero estudie con atención dicho código del servidor. Despues podrá escribir el código del cliente, cortando y pegando líneas del código del servidor según sea necesario.

Tarea 3: cliente de correo

El objetivo de esta tarea de programación es crear un cliente de correo simple que envíe correos electrónicos a cualquier destinatario. Su cliente necesitará establecer una conexión TCP con un servidor de correo (por ejemplo, el servidor de correo de Google), dialogar con el servidor de correo utilizando el protocolo SMTP, enviar un mensaje de correo electrónico a un receptor (por ejemplo, un amigo suyo) a través del servidor de correo y, finalmente, cerrar la conexión TCP con el servidor de correo.

Para esta tarea, el sitio web de acompañamiento proporciona el esqueleto de código para el cliente. Su tarea consiste en completar el código y probar su cliente enviando correos electrónicos a diferentes cuentas de usuario. También puede intentar enviar los correos a través de diferentes servidores (por ejemplo, a través de un servidor de correo de Google y del servidor de correo de su universidad).

Tarea 4: proxy web multihebra

En esta tarea, tendrá que desarrollar un proxy web. Cuando su proxy reciba de un navegador una solicitud HTTP para un cierto objeto, deberá generar una nueva solicitud HTTP para el mismo objeto y enviarla al servidor de destino. Cuando el proxy reciba desde el servidor de destino la correspondiente respuesta HTTP con el objeto, deberá crear una nueva respuesta HTTP que incluya el objeto y enviarla al cliente. Este proxy deberá ser multihebra, para poder ser capaz de gestionar múltiples solicitudes simultáneamente.

Para esta tarea, el sitio web de acompañamiento proporciona el esqueleto de código del servidor proxy. Su tarea consiste en completar el código y luego probarlo, haciendo que diferentes navegadores soliciten objetos web a través de su proxy.

Prácticas de laboratorio con Wireshark: HTTP

En la práctica de laboratorio 1 nos hemos familiarizado con el husmeador de paquetes (*sniffer*) Wireshark, así que ya estamos preparados para utilizar Wireshark e investigar el funcionamiento de los protocolos. En esta práctica de laboratorio exploraremos varios aspectos del protocolo HTTP: la

interacción básica GET/respuesta, los formatos de los mensajes HTTP, la recuperación de archivos HTML de gran tamaño, la recuperación de archivos HTML con direcciones URL incrustadas, las conexiones persistentes y no persistentes, y la autenticación y la seguridad de HTTP.

Al igual que todas las prácticas de laboratorio con Wireshark, la descripción completa de esta práctica se encuentra en el sitio web del libro, www.pearsonhighered.com/cs-resources.

Prácticas de laboratorio con Wireshark: DNS

En esta práctica de laboratorio echaremos un rápido vistazo al lado del cliente de DNS, el protocolo que traduce los nombres de host Internet en direcciones IP. Como hemos visto en la Sección 2.5, el papel del cliente en el protocolo DNS es relativamente simple: un cliente envía una consulta a su servidor DNS local y recibe una respuesta. Sin embargo, son muchas las cosas que suceden por debajo, invisibles para los clientes DNS, ya que los servidores DNS jerárquicos se comunican entre sí de forma recursiva o iterativa para resolver la consulta DNS del cliente. No obstante, desde el punto de vista del cliente DNS, el protocolo es muy simple: se plantea una consulta al servidor DNS local y dicho servidor devuelve una respuesta. En esta práctica de laboratorio vamos a observar al protocolo DNS en acción.

Al igual que todas las prácticas de laboratorio con Wireshark, la descripción completa de esta práctica de laboratorio está disponible en el sitio web del libro, www.pearsonhighered.com/cs-resources.

UNA ENTREVISTA CON...

Marc Andreessen

Marc Andreessen es el co-creador de Mosaic, el navegador que popularizó la World Wide Web en 1993. Mosaic tenía una interfaz limpia, fácilmente comprensible, y fue el primer navegador en mostrar imágenes intercaladas en el texto. En 1994, Marc Andreessen y Jim Clark fundaron Netscape, cuyo navegador fue con mucho el más popular a mediados de la década de 1990. Netscape desarrolló también el protocolo SSL (Secure Sockets Layer, capa de conectores seguros) y muchos productos de servidor para Internet, incluyendo servidores de correo y servidores web basados en SSL. Ahora es co-fundador y socio de la empresa de capital riesgo Andreessen Horowitz, supervisando la cartera de inversiones en empresas entre las que se incluyen Facebook, Foursquare, Groupon, Jawbone, Twitter y Zynga. Es miembro de numerosos consejos de administración, incluyendo los de Bump, eBay, Glam Media, Facebook y Hewlett-Packard. Es licenciado en Ciencias de la Computación por la Universidad de Illinois en Urbana-Champaign.



¿Cómo llegó a interesarse por el mundo de la computación? ¿Siempre supo que quería trabajar en tecnologías de la información?

Las revoluciones de los videojuegos y de las computadoras personales se produjeron siendo yo niño - las computadoras personales representaban la nueva frontera de la tecnología a finales de los 70 y principios de los 80. Y no eran solamente Apple y el IBM PC, sino también cientos de nuevas empresas, como Commodore y Atari. Aprendí yo solo a programar, con un libro titulado "Instant Freeze-Dried BASIC" a los 10 años y conseguí mi primera computadora (una computadora TRS-80 a color, ¡ffíjate!) a los 12.

Describa uno o dos de los proyectos más interesantes en los que haya trabajado durante su carrera. ¿Cuáles fueron los principales desafíos?

Sin ninguna duda, el proyecto más interesante fue el navegador web Mosaic original en 1992-1993; y el principal desafío fue conseguir que alguien se lo tomara en serio por aquella época. En aquel entonces, todo el mundo creía que el futuro interactivo sería proporcionado en forma de "televisión interactiva" por grandes empresas, no que iba a tomar la forma de Internet gracias a empresas de nueva creación.

¿Qué es lo que le entusiasma más acerca del futuro de las redes y de Internet? ¿Cuáles son sus principales preocupaciones?

Lo más excitante es la enorme frontera inexplorada de aplicaciones y servicios que los programadores y los emprendedores tienen en su mano explorar - Internet ha liberado la creatividad a un nivel que no creo que hayamos visto nunca antes. Mi principal preocupación es el principio de las consecuencias no deseadas: no siempre conocemos las implicaciones de lo que hacemos, como por ejemplo que Internet sea usada por los gobiernos para implementar un mayor nivel de vigilancia de los ciudadanos.

¿Hay algo en concreto de lo que los estudiantes debieran ser conscientes a medida que avanza la tecnología web?

La tasa de cambio: la lección más importante que hay que aprender es cómo aprender; cómo adaptarse de forma flexible a los cambios en tecnologías específicas y cómo mantener una mente abierta acerca de las nuevas oportunidades y posibilidades a medida que progresamos en nuestra carrera profesional.

¿Qué personas le han inspirado profesionalmente?

Vannevar Bush, Ted Nelson, Doug Engelbart, Nolan Bushnell, Bill Hewlett y Dave Packard, Ken Olsen, Steve Jobs, Steve Wozniak, Andy Grove, Grace Hopper, Hedy Lamarr, Alan Turing, Richard Stallman.

¿Qué le recomendaría a los estudiantes que quieran orientar su carrera profesional hacia la computación y la tecnología de la información?

Profundiza lo más que puedas en la comprensión de cómo se crea la tecnología, y luego complementa tus conocimientos aprendiendo cómo funciona una empresa.

¿Puede la tecnología resolver los problemas del mundo?

No, pero elevamos el nivel de vida de la gente gracias al crecimiento económico, y la mayor parte del crecimiento económico a lo largo de la Historia ha sido consecuencia de la tecnología. Así que eso es lo mejor que podemos hacer.

La capa de transporte

Entre las capas de aplicación y de red se encuentra la capa de transporte, una pieza fundamental de la arquitectura de red en capas. Desempeña el papel crítico de proporcionar directamente servicios de comunicación a los procesos de aplicación que se ejecutan en hosts diferentes. El método didáctico que vamos a aplicar a lo largo de este capítulo va a consistir en alternar las explicaciones sobre los principios de la capa de transporte con explicaciones acerca de cómo estos principios se implementan en los protocolos existentes; como siempre, haremos un especial hincapié en los protocolos de Internet, en particular en los protocolos de transporte TCP y UDP.

Comenzaremos explicando la relación existente entre las capas de transporte y de red. Para ello, examinaremos la primera función crítica de la capa de transporte: ampliar el servicio de entrega de la capa de red entre dos sistemas terminales a un servicio de entrega entre dos procesos de la capa de aplicación que se ejecutan en los sistemas terminales. Ilustraremos esta función con el protocolo de transporte sin conexión de Internet, UDP.

A continuación, volveremos a los principios y afrontaremos uno de los problemas más importantes de las redes de computadoras: cómo dos entidades pueden comunicarse de forma fiable a través de un medio que puede perder o corromper los datos. A través de una serie de escenarios cada vez más complejos (¡y realistas!), construiremos un conjunto de técnicas que estos protocolos de transporte emplean para resolver este problema. Después, mostraremos cómo esos principios están integrados en TCP, el protocolo de transporte orientado a la conexión de Internet.

Luego pasaremos al segundo problema más importante de las redes: controlar la velocidad de transmisión de las entidades de la capa de transporte con el fin de evitar, o recuperarse de, las congestiones que tienen lugar dentro de la red. Consideraremos las causas y las consecuencias de la congestión, así como las técnicas de control de congestión más comúnmente utilizadas. Una vez que haya entendido los problemas que hay detrás de los mecanismos de control de congestión, estudiaremos el método que aplica TCP para controlar la congestión.

3.1 La capa de transporte y sus servicios

En los dos capítulos anteriores hemos visto por encima cuál es la función de la capa de transporte, así como los servicios que proporciona. Repasemos rápidamente lo que ya sabemos acerca de la capa de transporte.

Un protocolo de la capa de transporte proporciona una **comunicación lógica** entre procesos de aplicación que se ejecutan en hosts diferentes. Por *comunicación lógica* queremos decir que, desde la perspectiva de la aplicación, es como si los hosts que ejecutan los procesos estuvieran conectados directamente; en realidad, los hosts pueden encontrarse en puntos opuestos del planeta, conectados mediante numerosos routers y a través de un amplio rango de tipos de enlace. Los procesos de aplicación utilizan la comunicación lógica proporcionada por la capa de transporte para enviarse mensajes entre sí, sin preocuparse por los detalles de la infraestructura física utilizada para transportar estos mensajes. La Figura 3.1 ilustra el concepto de comunicación lógica.

Como se muestra en la Figura 3.1, los protocolos de la capa de transporte están implementados en los sistemas terminales, pero no en los routers de la red. En el lado emisor, la capa de transporte convierte los mensajes que recibe procedentes de un proceso de aplicación emisor en paquetes de la capa de transporte, conocidos como **segmentos** de la capa de transporte en la terminología de Internet. Muy posiblemente, esto se hace dividiendo los mensajes de la aplicación en fragmentos más pequeños y añadiendo una cabecera de la capa de transporte a cada fragmento, con el fin de crear el segmento de la capa de transporte. A continuación, la capa de transporte pasa el segmento a la capa de red del sistema terminal emisor, donde el segmento se encapsula dentro de un paquete de la capa de red (un datagrama) y se envía al destino. Es importante destacar que los routers de la red solo actúan sobre los campos correspondientes a la capa de red del datagrama; es decir, no examinan los campos del segmento de la capa de transporte encapsulado en el datagrama. En el lado receptor, la capa de red extrae el segmento de la capa de transporte del datagrama y lo sube a la capa de transporte. A continuación, esta capa procesa el segmento recibido, poniendo los datos del segmento a disposición de la aplicación de recepción.

Para las aplicaciones de red puede haber más de un protocolo de la capa de transporte disponible. Por ejemplo, Internet tiene dos protocolos: TCP y UDP. Cada uno de estos protocolos proporciona un conjunto diferente de servicios para la capa de transporte a la aplicación que lo haya invocado.

3.1.1 Relaciones entre las capas de transporte y de red

Recuerde que la capa de transporte se encuentra justo encima de la capa de red dentro de la pila de protocolos. Mientras que un protocolo de la capa de transporte proporciona una comunicación lógica entre *procesos* que se ejecutan en hosts diferentes, un protocolo de la capa de red proporciona una comunicación lógica entre *hosts*. Esta distinción es sutil, pero importante. Examinemos esta distinción con la ayuda de una analogía.

Considere dos viviendas, una situada en la costa este de Estados Unidos y otra en la costa oeste, y que en cada hogar viven una docena de niños. Los niños de la costa este son primos de los niños de la costa oeste. A todos los niños les gusta escribirse, y cada niño escribe a todos sus primos todas las semanas, enviando una carta a través del servicio postal ordinario para cada uno de ellos y empleando un sobre para cada uno. Así, cada casa envía 144 cartas a la otra casa cada semana (estos niños podrían ahorrar mucho dinero si utilizaran el correo electrónico). En cada uno de los hogares, hay un niño (Ann en la costa oeste y Bill en la costa este) responsable de recoger y distribuir el correo. Todas las semanas, Ann visita a sus hermanos y hermanas, les recoge el correo y lo entrega a la persona del servicio postal que pasa a diario por su casa. Cuando las cartas llegan al hogar de la costa oeste, Ann también se ocupa de distribuir el correo al resto de sus hermanos y hermanas. Bill hace el mismo trabajo que Ann en la costa este.

En este ejemplo, el servicio postal proporciona una comunicación lógica entre las dos casas (el servicio postal lleva el correo de una casa a la otra, no de una persona a otra). Por otro lado, Ann y Bill proporcionan una comunicación lógica entre los primos (recogen el correo y se lo entregan a sus

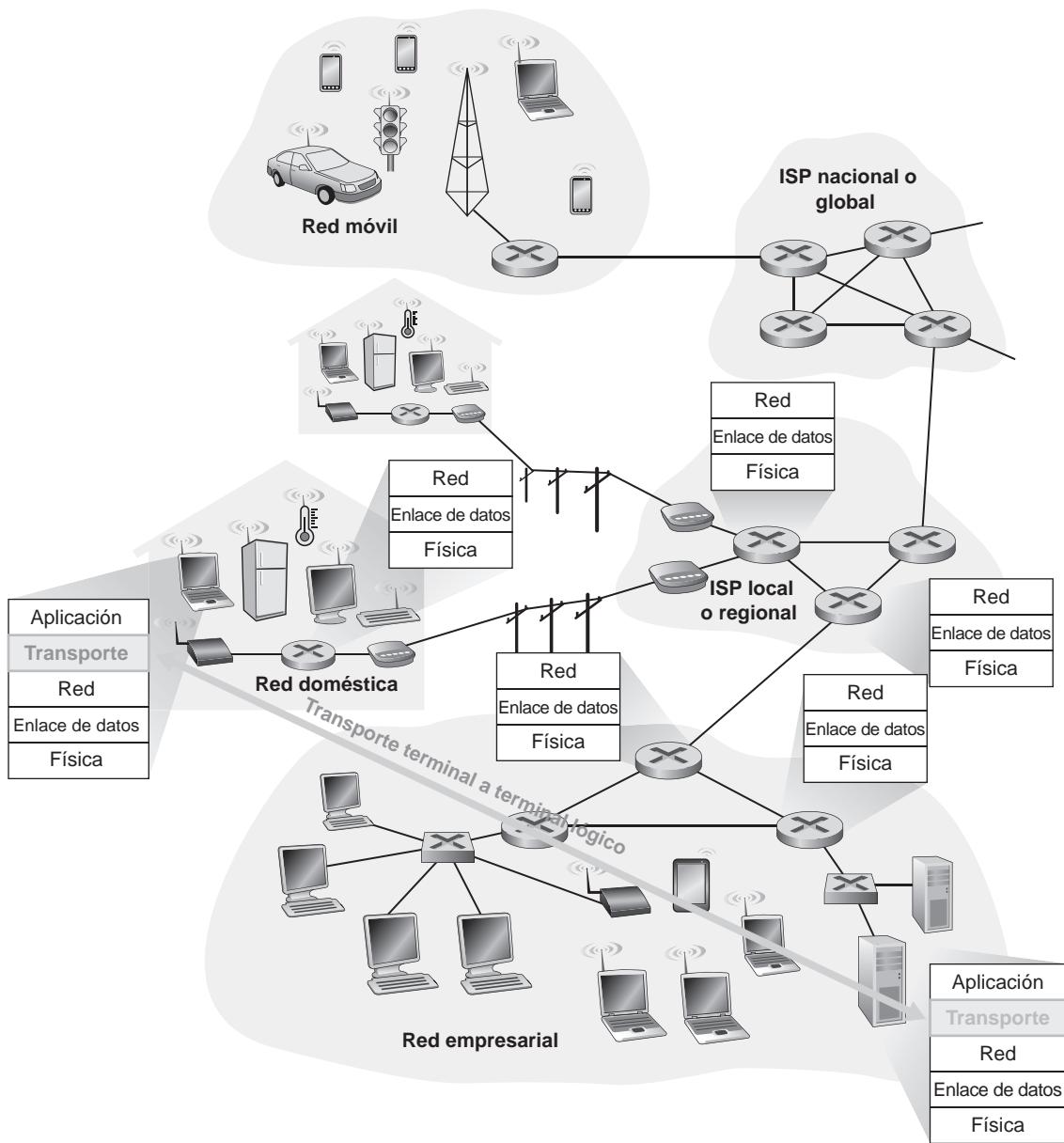


Figura 3.1 ♦ La capa de transporte proporciona una comunicación lógica en lugar de física entre los procesos de aplicación.

hermanos). Observe que desde la perspectiva de los primos, Ann y Bill *son* el servicio de correo, aun siendo ambos solamente una parte (el sistema terminal) del proceso de entrega terminal a terminal. Este ejemplo doméstico es una sencilla analogía que nos permite explicar cómo se relaciona la capa de transporte con la capa de red:

- mensajes de la aplicación = las cartas introducidas en los sobres
- procesos = los primos
- hosts (también denominados sistemas terminales) = las casas
- protocolo de la capa de transporte = Ann y Bill
- protocolo de la capa de red = el servicio postal (incluyendo a los carteros)

Continuando con esta analogía, fíjese en que Ann y Bill hacen su trabajo dentro de sus respectivas casas; es decir, no están implicados, por ejemplo, en el proceso de ordenación del correo en una oficina de correos intermedia, ni tampoco trasladan el correo de una oficina a otra. De manera similar, los protocolos de la capa de transporte residen en los sistemas terminales. Dentro de un sistema terminal, el protocolo de transporte lleva los mensajes desde los procesos de la aplicación a la frontera de la red (es decir, a la capa de red) y viceversa, pero no tiene nada que ver con cómo se transmiten los mensajes dentro del núcleo de la red. De hecho, como se ilustra en la Figura 3.1, los routers intermedios ni actúan sobre la información que la capa de transporte pueda añadir a los mensajes de la aplicación ni tampoco la reconocen.

Suponga ahora que Ann y Bill se van de vacaciones y que otra pareja de primos, por ejemplo, Susan y Harvey, les sustituyen y son los que recogen y reparten el correo en sus respectivas casas. Lamentablemente para las dos familias, Susan y Harvey no recogen y reparten el correo de la misma forma que lo hacían Ann y Bill. Al ser más pequeños, Susan y Harvey recogen y entregan el correo menos frecuentemente y, ocasionalmente, pierden algunas cartas (que a veces se come el perro). Por tanto, la pareja de primos Susan y Harvey no proporcionan el mismo conjunto de servicios (es decir, el mismo modelo de servicio) que Ann y Bill. De forma análoga, una red de computadoras puede emplear distintos protocolos de transporte, ofreciendo cada uno de ellos un modelo de servicio distinto a las aplicaciones.

Los posibles servicios que Ann y Bill pueden proporcionar están evidentemente restringidos por los posibles servicios que el servicio postal suministra. Por ejemplo, si el servicio postal no especifica el tiempo máximo que se puede tardar en entregar el correo entre ambos hogares (por ejemplo, tres días), entonces Ann y Bill no tienen ninguna forma de garantizar un retardo máximo en la entrega del correo entre cualquier pareja de primos. Del mismo modo, los servicios que un protocolo de transporte puede proporcionar a menudo están restringidos por el modelo de servicio del protocolo de la capa de red subyacente. Si el protocolo de la capa de red no proporciona garantías acerca del retardo ni del ancho de banda para los segmentos de la capa de transporte enviados entre hosts, entonces el protocolo de la capa de transporte no puede proporcionar ninguna garantía acerca del retardo o del ancho de banda a los mensajes de aplicación enviados entre procesos.

No obstante, el protocolo de transporte *puede* ofrecer ciertos servicios incluso cuando el protocolo de red subyacente no ofrezca el servicio correspondiente en la capa de red. Por ejemplo, como veremos más adelante en el capítulo, un protocolo de transporte puede ofrecer un servicio de transferencia de datos fiable a una aplicación, incluso si el protocolo de red subyacente no es fiable; es decir, incluso si el protocolo de red pierde, altera o duplica paquetes. Otro ejemplo, que examinaremos en el Capítulo 8 cuando hablaremos de la seguridad de la red, es que un protocolo de transporte puede emplear mecanismos de cifrado para garantizar que los mensajes de una aplicación no sean leídos por intrusos, incluso aunque la capa de red no pueda garantizar la privacidad de los segmentos de la capa de transporte.

3.1.2 La capa de transporte en Internet

Recuerde que Internet pone a disposición de la capa de aplicación dos protocolos de la capa de transporte diferentes. Uno de estos protocolos es el Protocolo de datagramas de usuario (**UDP**, *User Datagram Protocol*), que proporciona un servicio sin conexión no fiable a la aplicación que le invoca. El segundo de estos protocolos es el Protocolo de control de transmisión (**TCP**, *Transmission Control Protocol*), que proporciona a la aplicación que le invoca un servicio orientado a la conexión fiable. Al diseñar una aplicación de red, el desarrollador tiene que especificar el uso de uno de estos dos protocolos de transporte. Como hemos visto en las Sección 2.7, el desarrollador de la aplicación elige entre UDP y TCP cuando crea los sockets.

Para simplificar la terminología, nos referiremos a los paquetes de la capa de transporte como *segmentos*. No obstante, tenemos que decir que, en textos dedicados a Internet, como por ejemplo los RFC, también se emplea el término segmento para hacer referencia a los paquetes de la capa de transporte en el caso de TCP, pero a menudo a los paquetes de UDP se les denomina datagrama.

Pero resulta que estos mismos textos dedicados a Internet también utilizan el término *datagrama* para referirse a los paquetes de la capa de red. Por tanto, pensamos que en un libro de introducción a las redes de computadoras como este, resultará menos confuso hablar de segmentos tanto para referirse a los paquetes TCP como UDP, y reservar el término *datagrama* para los paquetes de la capa de red.

Antes de continuar con esta breve introducción a los protocolos UDP y TCP, nos será útil comentar algunas cosas acerca de la capa de red de Internet (en los Capítulos 4 y 5 examinaremos en detalle la capa de red). El protocolo de la capa de red de Internet es el protocolo IP (*Internet Protocol*). IP proporciona una comunicación lógica entre hosts. El modelo de servicio de IP es un **servicio de entrega de mejor esfuerzo (best effort)**. Esto quiere decir que IP hace todo lo que puede por entregar los segmentos entre los hosts que se están comunicando, *pero no garantiza la entrega*. En particular, no garantiza la entrega de los segmentos, no garantiza que los segmentos se entreguen en orden y no garantiza la integridad de los datos contenidos en los segmentos. Por estas razones, se dice que IP es un **servicio no fiable**. Además, sabemos que todos los hosts tienen al menos una dirección de capa de red, que se conoce como dirección IP. En el Capítulo 4 se estudia en detalle el direccionamiento IP, pero por el momento lo único que necesitamos saber es que *todo host tiene una dirección IP asociada*.

Después de haber echado una ojeada al modelo de servicio de IP, haremos un breve resumen de los modelos de servicio proporcionados por UDP y TCP. La responsabilidad principal de UDP y TCP es ampliar el servicio de entrega de IP entre dos sistemas terminales a un servicio de entrega entre dos procesos que estén ejecutándose en los sistemas terminales. Extender la entrega host a host a una entrega proceso a proceso es lo que se denomina **multiplexación y demultiplexación de la capa de transporte**, tema que desarrollaremos en la siguiente sección. UDP y TCP también proporcionan servicios de comprobación de la integridad de los datos al incluir campos de detección de errores en las cabeceras de sus segmentos. Estos dos servicios de la capa de transporte (entrega de datos proceso a proceso y comprobación de errores) son los dos únicos servicios que ofrece UDP. En particular, al igual que IP, UDP es un servicio no fiable, que no garantiza que los datos enviados por un proceso lleguen intactos (¡o que ni siquiera lleguen!) al proceso de destino. En la Sección 3.3 se aborda en detalle el protocolo UDP.

TCP, por el contrario, ofrece a las aplicaciones varios servicios adicionales. El primero y más importante es que proporciona una **transferencia de datos fiable**. Utilizando técnicas de control de flujo, números de secuencia, mensajes de reconocimiento y temporizadores (técnicas que estudiaremos en detalle en este capítulo), TCP garantiza que los datos transmitidos por el proceso emisor sean entregados al proceso receptor, correctamente y en orden. De este modo, TCP convierte el servicio no fiable de IP entre sistemas terminales en un servicio de transporte de datos fiable entre procesos. TCP también proporciona mecanismos de **control de congestión**. El control de congestión no es tanto un servicio proporcionado a la aplicación invocante, cuanto un servicio que se presta a Internet como un todo, un servicio para el bien común. En otras palabras, los mecanismos de control de congestión de TCP evitan que cualquier conexión TCP inunde con una cantidad de tráfico excesiva los enlaces y routers existentes entre los hosts que están comunicándose. TCP se esfuerza en proporcionar a cada conexión que atraviesa un enlace congestionado la misma cuota de ancho de banda del enlace. Esto se consigue regulando la velocidad a la que los lados emisores de las conexiones TCP pueden enviar tráfico a la red. El tráfico UDP, por el contrario, no está regulado. Una aplicación que emplee el protocolo de transporte UDP puede enviar los datos a la velocidad que le parezca, durante todo el tiempo que quiera.

Un protocolo que proporciona una transferencia de datos fiable y mecanismos de control de congestión necesariamente es un protocolo complejo. Vamos a necesitar varias secciones de este capítulo para examinar los principios de la transferencia de datos fiable y el control de congestión, además de otras secciones adicionales dedicadas a cubrir el propio protocolo TCP. Estos temas se tratan en las Secciones 3.4 a 3.8. El método de trabajo que hemos adoptado en este capítulo es el de alternar entre los principios básicos y el protocolo TCP. Por ejemplo, en primer lugar veremos en qué consiste en general una transferencia de datos fiable y luego veremos específicamente cómo TCP

proporciona ese servicio de transferencia de datos fiable. De forma similar, primero definiremos de forma general en qué consiste el mecanismo de control de congestión y luego veremos cómo lo hace TCP. Pero antes de entrar en estos temas, veamos qué es la multiplexación y la demultiplexación de la capa de transporte.

3.2 Multiplexación y demultiplexación

En esta sección vamos a estudiar la multiplexación y demultiplexación de la capa de transporte; es decir, la ampliación del servicio de entrega host a host proporcionado por la capa de red a un servicio de entrega proceso a proceso para las aplicaciones que se ejecutan en los hosts. Con el fin de centrar la explicación, vamos a ver este servicio básico de la capa de transporte en el contexto de Internet. Sin embargo, hay que destacar que un servicio de multiplexación/demultiplexación es necesario en todas las redes de computadoras.

En el host de destino, la capa de transporte recibe segmentos procedentes de la capa de red que tiene justo debajo. La capa de transporte tiene la responsabilidad de entregar los datos contenidos en estos segmentos al proceso de la aplicación apropiada que está ejecutándose en el host. Veamos un ejemplo. Suponga que está sentado frente a su computadora y que está descargando páginas web a la vez que ejecuta una sesión FTP y dos sesiones Telnet. Por tanto, tiene cuatro procesos de aplicación de red en ejecución: dos procesos Telnet, un proceso FTP y un proceso HTTP. Cuando la capa de transporte de su computadora recibe datos procedentes de la capa de red, tiene que dirigir los datos recibidos a uno de estos cuatro procesos. Veamos cómo hace esto.

En primer lugar, recordemos de la Sección 2.7 que un proceso (como parte de una aplicación de red) puede tener uno o más **sockets**, puertas por las que pasan los datos de la red al proceso, y viceversa. Por tanto, como se muestra en la Figura 3.2, la capa de transporte del host receptor realmente no entrega los datos directamente a un proceso, sino a un socket intermedio. Dado que en cualquier instante puede haber más de un socket en el host receptor, cada socket tiene asociado un identificador único. El formato de este identificador depende de si se trata de un socket UDP o de un socket TCP, como vamos a ver a continuación.

Veamos ahora cómo un host receptor dirige un segmento de entrada de la capa de transporte al socket apropiado. Cada segmento de la capa de transporte contiene un conjunto de campos destinados a este propósito. En el extremo receptor, la capa de transporte examina estos campos para identificar el socket receptor y, a continuación, envía el segmento a dicho socket. Esta tarea

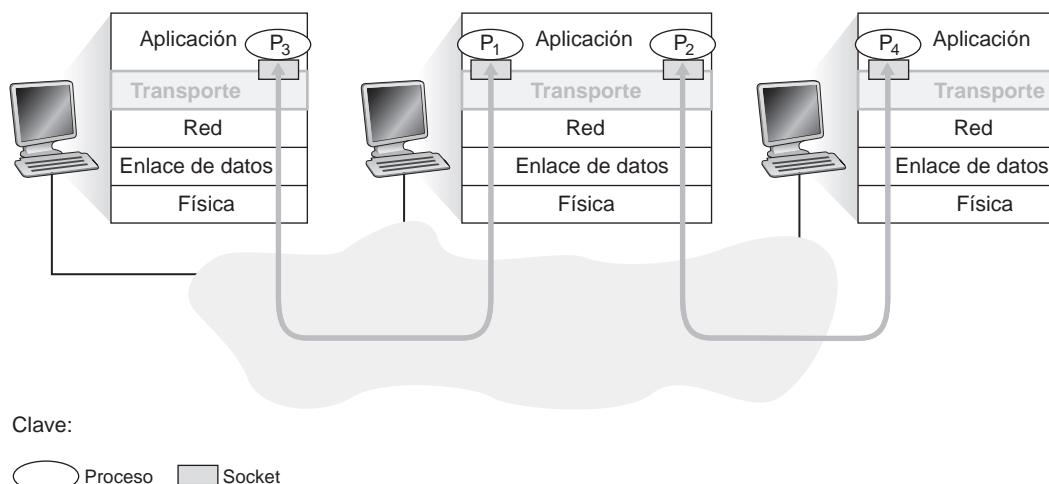


Figura 3.2 ♦ Multiplexación y demultiplexación en la capa de transporte.

de entregar los datos contenidos en un segmento de la capa de transporte al socket correcto es lo que se denomina **demultiplexación**. La tarea de reunir los fragmentos de datos en el host de origen desde los diferentes sockets, encapsulando cada fragmento de datos con la información de cabecera (la cual se utilizará después en el proceso de demultiplexación) para crear los segmentos y pasarlos a la capa de red es lo que se denomina **multiplexación**. Observe que la capa de transporte del host intermedio de la Figura 3.2 tiene que demultiplexar los segmentos que llegan de la capa de red inferior para entregárselos a los procesos P_1 o P_2 de la capa superior; esto se hace dirigiendo los datos del segmento entrante al correspondiente socket del proceso. La capa de transporte del host intermedio también tiene que recopilar los datos salientes desde estos sockets, construir los segmentos de la capa de transporte y pasar dichos segmentos a la capa de red. Aunque hemos presentado la multiplexación y la demultiplexación en el contexto de los protocolos de transporte de Internet, es importante darse cuenta de que esas técnicas son necesarias siempre que un único protocolo en una capa (en la capa de transporte o cualquier otra) sea utilizado por varios protocolos de la capa inmediatamente superior.

Para ilustrar la tarea de demultiplexación, recordemos la analogía doméstica de la sección anterior. Cada uno de los niños está identificado por su nombre. Cuando Bill recibe un lote de cartas del servicio de correos, lleva a cabo una operación de demultiplexación al determinar a quién van dirigidas las cartas y luego las entrega en mano a sus hermanos. Ann lleva a cabo una operación de multiplexación al recopilar las cartas de sus hermanos y entregar todas las cartas al cartero.

Ahora que ya entendemos las funciones de multiplexación y demultiplexación de la capa de transporte, vamos a examinar cómo se hace esto realmente en un host. Basándonos en las explicaciones anteriores, sabemos que la operación de multiplexación que se lleva a cabo en la capa de transporte requiere (1) que los sockets tengan identificadores únicos y (2) que cada segmento tenga campos especiales que indiquen el socket al que tiene que entregarse el segmento. Estos campos especiales, mostrados en la Figura 3.3, son el campo **número de puerto de origen** y el campo **número de puerto de destino**. (Los segmentos UDP y TCP contienen además otros campos, como veremos en las siguientes secciones del capítulo.) Cada número de puerto es un número de 16 bits comprendido en el rango de 0 a 65535. Los números de puerto pertenecientes al rango de 0 a 1023 se conocen como **números de puertos bien conocidos** y son restringidos, lo que significa que están reservados para ser empleados por los protocolos de aplicación bien conocidos, como por ejemplo HTTP (que utiliza el número de puerto 80) y FTP (que utiliza el número de puerto 21). Puede encontrar la lista de números de puerto bien conocidos en el documento RFC 1700 y su actualización en la dirección <http://www.iana.org> [RFC 3232]. Al desarrollar una nueva aplicación (como las aplicaciones desarrolladas en la Sección 2.7), es necesario asignar un número de puerto a la aplicación.

Ahora ya debería estar claro cómo la capa de transporte *podría* implementar el servicio de demultiplexación: cada socket del host se puede asignar a un número de puerto y, al llegar un

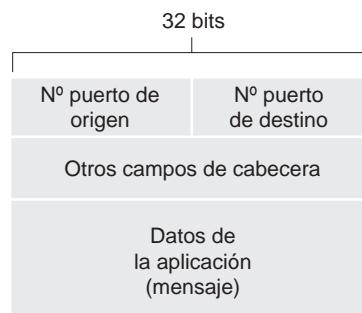


Figura 3.3 ♦ Los campos número de puerto de origen y de destino en un segmento de la capa de transporte.

segmento al host, la capa de transporte examina el número de puerto de destino contenido en el segmento y lo envía al socket correspondiente. A continuación, los datos del segmento pasan a través del socket y se entregan al proceso asociado. Como veremos, esto es básicamente lo que hace UDP. Sin embargo, también comprobaremos que la tarea de multiplexación/demultiplexación en TCP es más sutil.

Multiplexación y demultiplexación sin conexión

Recordemos de la Sección 2.7.1 que un programa Python que se ejecuta en un host puede crear un socket UDP mediante la línea de código:

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
```

Cuando se crea un socket UDP de este modo, la capa de transporte asigna automáticamente un número de puerto al socket. En particular, la capa de transporte asigna un número de puerto comprendido en el rango de 1024 a 65535 que actualmente no esté siendo utilizado en ese host por ningún otro puerto UDP. Alternativamente, podemos añadir una línea a nuestro programa Python después de crear el socket para asociar un número de puerto específico (por ejemplo, 19157) a este socket UDP mediante el método `bind()` del socket:

```
clientSocket.bind(('', 19157))
```

Si el desarrollador de la aplicación que escribe el código estuviera implementando el lado del servidor de un “protocolo bien conocido”, entonces tendría que asignar el correspondiente número de puerto bien conocido. Normalmente, el lado del cliente de la aplicación permite a la capa de transporte asignar de forma automática (y transparente) el número de puerto, mientras que el lado de servidor de la aplicación asigna un número de puerto específico.

Una vez asignados los números de puerto a los sockets UDP, podemos describir de forma precisa las tareas de multiplexación y demultiplexación en UDP. Suponga que un proceso del host A, con el puerto UDP 19157, desea enviar un fragmento de datos de una aplicación a un proceso con el puerto UDP 46428 en el host B. La capa de transporte del host A crea un segmento de la capa de transporte que incluye los datos de aplicación, el número de puerto de origen (19157), el número de puerto de destino (46428) y otros dos valores (que veremos más adelante, pero que por el momento no son importantes para el tema que nos ocupa). La capa de transporte pasa a continuación el segmento resultante a la capa de red. La capa de red encapsula el segmento en un datagrama IP y hace el máximo esfuerzo por entregar el segmento al host receptor. Si el segmento llega al host receptor B, la capa de transporte del mismo examina el número de puerto de destino especificado en el segmento (46428) y entrega el segmento a su socket identificado por el puerto 46428. Observe que el host B podría estar ejecutando varios procesos, cada uno de ellos con su propio socket UDP y número de puerto asociado. A medida que los segmentos UDP llegan de la red, el host B dirige (demultiplexa) cada segmento al socket apropiado examinando el número de puerto de destino del segmento.

Es importante observar que un socket UDP queda completamente identificado por una tupla que consta de una dirección IP de destino y un número de puerto de destino. En consecuencia, si dos segmentos UDP tienen diferentes direcciones IP y/o números de puerto de origen, pero la misma dirección IP de *destino* y el mismo número puerto de *destino*, entonces los dos segmentos se enviarán al mismo proceso de destino a través del mismo socket de destino.

Es posible que se esté preguntando en este momento cuál es el propósito del número de puerto de origen. Como se muestra en la Figura 3.4, en el segmento A a B, el número de puerto de origen forma parte de una “dirección de retorno”; es decir, si B desea devolver un segmento a A, el puerto de destino en el segmento de B a A tomará su valor del valor del puerto de origen del segmento de A a B. (La dirección de retorno completa es el número de puerto de origen y la dirección IP de A.) Por ejemplo, recuerde el programa de servidor UDP estudiado en la Sección 2.7. En `UDPServer.py`, el servidor utiliza el método `recvfrom()` para extraer el número de puerto (de origen) del lado

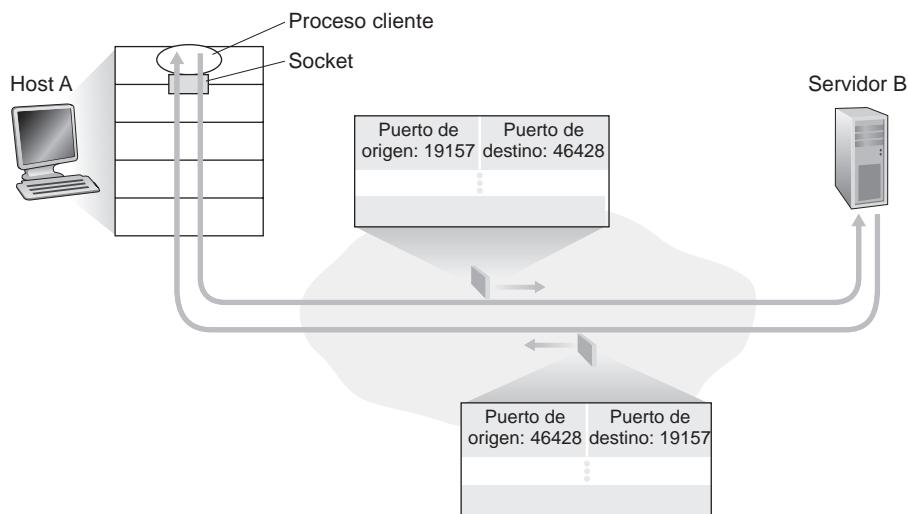


Figura 3.4 ♦ Inversión de los números de puerto de origen y de destino.

del cliente del segmento que recibe desde el cliente; a continuación, envía un segmento nuevo al cliente, utilizando como número de puerto de destino el número de puerto de origen extraído del segmento recibido.

Multiplexación y demultiplexación orientadas a la conexión

Para entender la demultiplexación TCP, tenemos que tener en cuenta los sockets TCP y el establecimiento de las conexiones TCP. Una sutil diferencia entre un socket TCP y un socket UDP es que el primero queda identificado por una tupla de cuatro elementos: dirección IP de origen, número de puerto de origen, dirección IP de destino, número de puerto de destino. Por tanto, cuando un segmento TCP llega a un host procedente de la red, el host emplea los cuatro valores para dirigir (demultiplexar) el segmento al socket apropiado. En particular, y al contrario de lo que ocurre con UDP, dos segmentos TCP entrantes con direcciones IP de origen o números de puerto de origen diferentes (con la excepción de un segmento TCP que transporte la solicitud original de establecimiento de conexión) serán dirigidos a dos sockets distintos. Con el fin de profundizar un poco, consideremos de nuevo el ejemplo de programación cliente-servidor TCP de la Sección 2.7.2:

- La aplicación de servidor TCP tiene un “socket de acogida”, que está a la espera de las solicitudes de establecimiento de conexión procedentes de los clientes TCP en el puerto 12000 (véase la Figura 2.29).
- El cliente TCP crea un socket y envía un segmento de establecimiento de conexión con la líneas de código:

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, 12000))
```

- Una solicitud de establecimiento de conexión no es nada más que un segmento TCP con el número de puerto de destino 12000 y un conjunto especial de bits de establecimiento de conexión en la cabecera TCP (que veremos en la Sección 3.5). El segmento también incluye un número de puerto de origen, que habrá sido seleccionado por el cliente.
- Cuando el sistema operativo del host que está ejecutando el proceso servidor recibe el segmento de entrada de solicitud de conexión con el puerto de destino 12000, localiza el proceso de

servidor que está esperando para aceptar una conexión en el número de puerto 12000. El proceso de servidor crea entonces un nuevo socket:

```
connectionSocket, addr = serverSocket.accept()
```

- Además, la capa de transporte en el servidor toma nota de los cuatro valores siguientes contenidos en el segmento de solicitud de conexión: (1) el número de puerto de origen en el segmento, (2) la dirección IP del host de origen, (3) el número de puerto de destino en el segmento y (4) su propia dirección IP. El socket de conexión recién creado queda identificado por estos cuatro valores; así, todos los segmentos que lleguen después y cuyo puerto de origen, dirección IP de origen, puerto de destino y dirección IP de destino se correspondan con estos cuatro valores serán enviados a este socket. Una vez establecida la conexión TCP, el cliente y el servidor podrán enviarse datos entre sí.

El host servidor puede dar soporte a muchos sockets TCP simultáneos, estando cada socket asociado a un proceso y con cada socket identificado por su tupla de cuatro elementos. Cuando un segmento TCP llega al host, los cuatro campos (dirección IP de origen, puerto de origen, dirección IP de destino y puerto de destino) se utilizan para dirigir (demultiplexar) el segmento al socket apropiado.

Esta situación se ilustra en la Figura 3.5, en la que el host C inicia dos sesiones HTTP con el servidor B y el host A inicia una sesión HTTP también con B. Los hosts A y C y el servidor B tienen sus propias direcciones IP únicas (A, C y B, respectivamente). El host C asigna dos números de puerto de origen diferentes (26145 y 7532) a sus dos conexiones HTTP. Dado que el host A está seleccionando los números de puerto de origen independientemente de C, también puede asignar el número de puerto de origen 26145 a su conexión HTTP. Pero esto no es un problema: el servidor B todavía podrá demultiplexar correctamente las dos conexiones con el mismo número de puerto de origen, ya que tienen direcciones IP de origen diferentes.

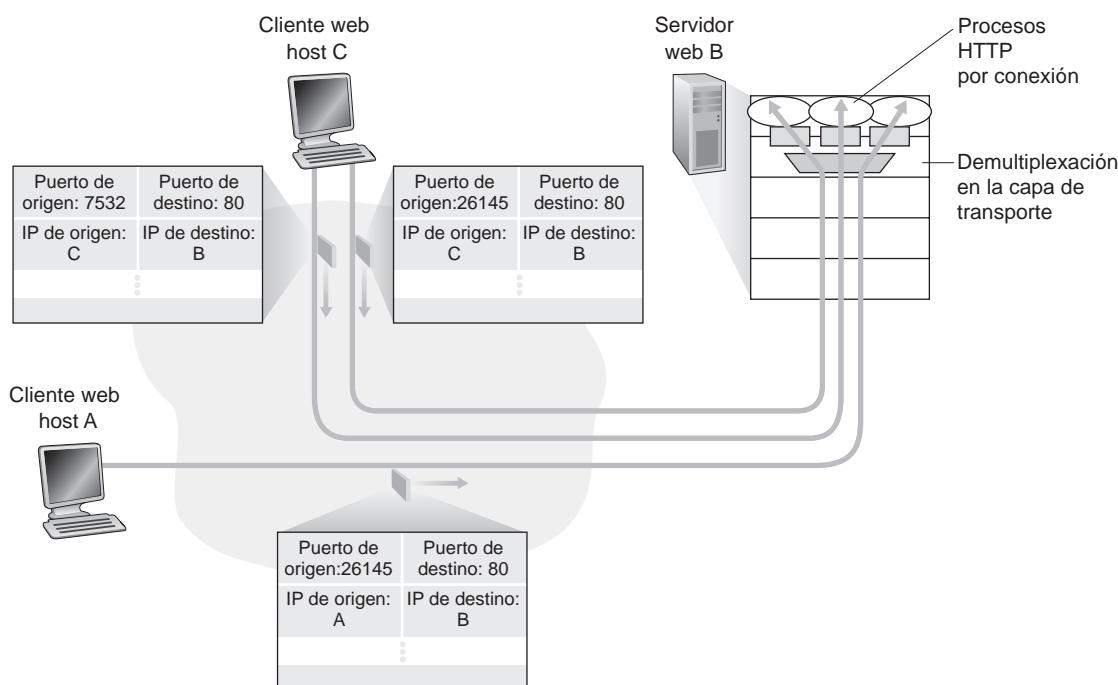


Figura 3.5 ♦ Dos clientes utilizando el mismo número de puerto de destino (80) para comunicarse con la misma aplicación de servidor web.

SEGURIDAD

EXPLORACIÓN DE PUERTOS

Hemos visto que un proceso servidor espera pacientemente en un puerto abierto a ser contactado por un cliente remoto. Algunos puertos están reservados para aplicaciones bien conocidas (como por ejemplo servidores web, FTP, DNS y SMTP); otros puertos, por convenio, son utilizados por aplicaciones populares (como por ejemplo Microsoft 2000 SQL Server que está a la escucha en el puerto UDP 1434). Por tanto, si determinamos que un puerto está abierto en un host, podemos ser capaces de hacer corresponder dicho puerto a una aplicación específica que se ejecute en el host. Esto es muy útil para los administradores de sistemas, ya que suelen estar interesados en saber qué aplicaciones de red están ejecutándose en los hosts de sus redes. Pero los atacantes, con el fin de detectar "las brechas en el muro", también desean saber qué puertos están abiertos en los hosts objetivo. Si se localiza un host que está ejecutando una aplicación con un defecto de seguridad conocido (por ejemplo, si un servidor SQL que está a la escucha en el puerto 1434 fuera objeto de un desbordamiento de buffer, permitiendo a un usuario remoto ejecutar código arbitrario en el host vulnerable, un defecto explotado por el gusano Slammer [CERT 2003-04])), entonces dicho host estaría en condiciones para recibir un ataque.

Determinar qué aplicaciones están a la escucha en qué puertos es una tarea relativamente fácil. Además, existen numerosos programas de dominio público, conocidos como escáneres de puertos, que realizan este trabajo. Quizá el más ampliamente utilizado de estos programas es nmap, que está disponible gratuitamente en <http://nmap.org> y se incluye en la mayor parte de las distribuciones de Linux. Para TCP, nmap explora secuencialmente los puertos buscando puertos que acepten conexiones TCP. Para UDP, nmap también explora secuencialmente los puertos buscando puertos UDP que respondan a segmentos UDP transmitidos. En ambos casos, nmap devuelve una lista de puertos abiertos, cerrados o inalcanzables. Un host que ejecute nmap puede tratar de explorar cualquier host objetivo situado en *cualquier lugar* de Internet. En la Sección 3.5.6 volveremos a hablar de nmap, al tratar la gestión de las conexiones TCP.

Servidores web y TCP

Antes de dar por terminada esta sección, es interesante comentar algunas cosas acerca de los servidores web y de cómo utilizan los números de puerto. Considere un host que está ejecutando un servidor web, como por ejemplo un servidor web Apache en el puerto 80. Cuando los clientes (por ejemplo, los navegadores) envían segmentos al servidor, *todos* los segmentos tendrán el puerto de destino 80. En particular, tanto los segmentos para el establecimiento de la conexión inicial como los segmentos que transportan los mensajes de solicitud HTTP utilizarán como puerto de destino el puerto 80. Como acabamos de describir, el servidor diferencia los segmentos procedentes de los distintos clientes mediante las direcciones IP de origen y los números de puerto de origen.

La Figura 3.5 nos muestra un servidor web que genera un nuevo proceso para cada conexión. Como se muestra en esta figura, cada uno de estos procesos tiene su propio socket de conexión a través del cual llegan las solicitudes HTTP y se envían las respuestas HTTP. Sin embargo, también hemos mencionado que no existe siempre una correspondencia uno a uno entre los sockets de conexión y los procesos. De hecho, los servidores web actuales de altas prestaciones a menudo solo utilizan un proceso y crean una nueva hebra con un nuevo socket de conexión para cada nueva conexión de un cliente (una hebra es una especie de subproceso de baja complejidad). Si realizó la primera tarea de programación del Capítulo 2, habrá creado un servidor web que hace exactamente esto. En un servidor así, en cualquier instante puede haber muchos sockets de conexión (con distintos identificadores) asociados al mismo proceso.

Si el cliente y el servidor están utilizando HTTP persistente, entonces mientras dure la conexión persistente el cliente y el servidor intercambiarán mensajes HTTP a través del mismo socket de

servidor. Sin embargo, si el cliente y el servidor emplean HTTP no persistente, entonces se creará y cerrará una nueva conexión TCP para cada pareja solicitud/respuesta y, por tanto, se creará (y luego se cerrará) un nuevo socket para cada solicitud/respuesta. Esta creación y cierre de sockets tan frecuente puede afectar seriamente al rendimiento de un servidor web ocupado (aunque se pueden utilizar una serie de trucos del sistema operativo para mitigar el problema). Los lectores interesados en los problemas que el uso de HTTP persistente y no persistente plantea a los sistemas operativos pueden consultar [Nielsen 1997; Nahum 2002].

Ahora que ya hemos visto en qué consiste la multiplexación y la demultiplexación en la capa de transporte, podemos pasar a tratar uno de los protocolos de transporte de Internet: UDP. En la siguiente sección veremos que UDP añade algo más al protocolo de la capa de red que un servicio de multiplexación/demultiplexación.

3.3 Transporte sin conexión: UDP

En esta sección vamos a ocuparnos de UDP, vamos a ver cómo funciona y qué hace. Le animamos a releer la Sección 2.1, en la que se incluye una introducción al modelo de servicio de UDP y la Sección 2.7.1, en la que se explica la programación de sockets con UDP.

Para iniciar nuestro análisis de UDP, suponga que queremos diseñar un protocolo de transporte simple y poco sofisticado. ¿Cómo haríamos esto? En primer lugar, podemos considerar la utilización de un protocolo de transporte vacío. Es decir, en el lado emisor, tomará los mensajes del proceso de la aplicación y los pasará directamente a la capa de red; en el lado de recepción, tomará los mensajes procedentes de la capa de red y los pasará directamente al proceso de la aplicación. Pero, como hemos aprendido en la sección anterior, hay algunas cosas mínimas que tenemos que hacer. Como mínimo, la capa de transporte tiene que proporcionar un servicio de multiplexación/demultiplexación que permita transferir los datos entre la capa de red y el proceso de la capa de aplicación correcto.

UDP, definido en el documento [RFC 768], hace casi lo mínimo que un protocolo de transporte debe hacer. Además de la función de multiplexación/demultiplexación y de algún mecanismo de comprobación de errores, no añade nada a IP. De hecho, si el desarrollador de la aplicación elige UDP en lugar de TCP, entonces prácticamente es la aplicación la que se comunica directamente con IP. UDP toma los mensajes procedentes del proceso de la aplicación, asocia los campos correspondientes a los números de puerto de origen y de destino para proporcionar el servicio de multiplexación/demultiplexación, añade dos campos pequeños más y pasa el segmento resultante a la capa de red. La capa de red encapsula el segmento de la capa de transporte en un datagrama IP y luego hace el mejor esfuerzo por entregar el segmento al host receptor. Si el segmento llega al host receptor, UDP utiliza el número de puerto de destino para entregar los datos del segmento al proceso apropiado de la capa de aplicación. Observe que con UDP no tiene lugar una fase de establecimiento de la conexión entre las entidades de la capa de transporte emisora y receptora previa al envío del segmento. Por esto, se dice que UDP es un protocolo *sin conexión*.

DNS es un ejemplo de un protocolo de la capa de aplicación que habitualmente utiliza UDP. Cuando la aplicación DNS de un host desea realizar una consulta, construye un mensaje de consulta DNS y lo pasa a UDP. Sin llevar a cabo ningún proceso para establecer una conexión con la entidad UDP que se ejecuta en el sistema terminal de destino, el protocolo UDP del lado del host añade campos de cabecera al mensaje y pasa el segmento resultante a la capa de red, la cual encapsula el segmento UDP en un datagrama y lo envía a un servidor de nombres. La aplicación DNS que se ejecuta en el host que ha hecho la consulta espera entonces hasta recibir una respuesta a su consulta. Si no la recibe (posiblemente porque la red subyacente ha perdido la consulta o la respuesta), bien intenta enviar la consulta a otro servidor de nombres o bien informa a la aplicación invocante de que no puede obtener una respuesta.

Es posible que ahora se esté preguntando por qué un desarrollador de aplicaciones podría decidir crear una aplicación sobre UDP en lugar de sobre TCP. ¿No sería preferible emplear siempre TCP,

puesto que proporciona un servicio de transferencia de datos fiable y UDP no? La respuesta es no, ya que muchas aplicaciones están mejor adaptadas a UDP por las siguientes razones:

- *Mejor control en el nivel de aplicación sobre qué datos se envían y cuándo.* Con UDP, tan pronto como un proceso de la capa de aplicación pasa datos a UDP, UDP los empaqueta en un segmento UDP e inmediatamente entrega el segmento a la capa de red. Por el contrario, TCP dispone de un mecanismo de control de congestión que regula el flujo del emisor TCP de la capa de transporte cuando uno o más de los enlaces existentes entre los hosts de origen y de destino están excesivamente congestionados. TCP también continuará reenviando un segmento hasta que la recepción del mismo haya sido confirmada por el destino, independientemente de cuánto se tarde en llevar a cabo esta entrega fiable. Puesto que las aplicaciones en tiempo real suelen requerir una velocidad mínima de transmisión, no permiten un retardo excesivo en la transmisión de los segmentos y pueden tolerar algunas pérdidas de datos, el modelo de servicio de TCP no se adapta demasiado bien a las necesidades de este tipo de aplicaciones. Como veremos más adelante, estas aplicaciones pueden utilizar UDP e implementar, como parte de la aplicación, cualquier funcionalidad adicional que sea necesaria más allá del servicio básico de entrega de segmentos de UDP.
- *Sin establecimiento de la conexión.* Como se explicará más adelante, TCP lleva a cabo un proceso de establecimiento de la conexión en tres fases antes de iniciar la transferencia de datos. UDP inicia la transmisión sin formalidades preliminares. Por tanto, UDP no añade ningún retardo a causa del establecimiento de una conexión. Probablemente, esta es la razón principal por la que DNS opera sobre UDP y no sobre TCP (DNS sería mucho más lento si se ejecutara sobre TCP). HTTP utiliza TCP en lugar de UDP, ya que la fiabilidad es crítica para las páginas web con texto. Pero, como hemos comentado brevemente en la Sección 2.2, el retardo debido al establecimiento de la conexión TCP en HTTP contribuye de forma notable a los retardos asociados con la descarga de documentos web. De hecho, el protocolo QUIC (*Quick UDP Internet Connection*, [Iyengar 2015]), utilizado en el navegador Chrome de Google, utiliza UDP como su protocolo de transporte subyacente e implementa la fiabilidad en un protocolo de la capa de aplicación por encima de UDP.
- *Sin información del estado de la conexión.* TCP mantiene información acerca del estado de la conexión en los sistemas terminales. En el estado de la conexión se incluye información acerca de los buffers de recepción y envío, de los parámetros de control de congestión y de los parámetros relativos al número de secuencia y de reconocimiento. En la Sección 3.5 veremos que esta información de estado es necesaria para implementar el servicio fiable de transferencia de datos y proporcionar los mecanismos de control de congestión de TCP. Por el contrario, UDP no mantiene información del estado de la conexión y no controla ninguno de estos parámetros. Por esta razón, un servidor dedicado a una aplicación concreta suele poder soportar más clientes activos cuando la aplicación se ejecuta sobre UDP que cuando lo hace sobre TCP.
- *Poca sobrecarga debida a la cabecera de los paquetes.* Los segmentos TCP contienen 20 bytes en la cabecera de cada segmento, mientras que UDP solo requiere 8 bytes.

La tabla de la Figura 3.6 enumera aplicaciones de Internet muy populares junto con los protocolos de transporte que utilizan. Como era de esperar, el correo electrónico, el acceso remoto a terminales, la Web y la transferencia de archivos se ejecutan sobre TCP, ya que todas estas aplicaciones necesitan el servicio fiable de transferencia de datos de TCP. No obstante, muchas aplicaciones importantes se ejecutan sobre UDP en lugar de sobre TCP. Por ejemplo, UDP se utiliza para transmitir los datos de la administración de red (SNMP, véase la Sección 5.7). En este caso, UDP es preferible a TCP, ya que las aplicaciones de administración de la red a menudo se tienen que ejecutar cuando la red se encuentra en un estado de sobrecarga (precisamente en las situaciones en las que es difícil realizar transferencias de datos fiables y con control de la congestión). Además, como ya hemos mencionado anteriormente, DNS se ejecuta sobre UDP, evitando de este modo los retardos de establecimiento de la conexión TCP.

Aplicación	Protocolo de la capa de aplicación	Protocolo de transporte subyacente
Correo electrónico	SMTP	TCP
Acceso a terminales remotos	Telnet	TCP
Web	HTTP	TCP
Transferencia de archivos	FTP	TCP
Servidor de archivos remoto	NFS	Normalmente UDP
Flujos multimedia	Normalmente proprietario	UDP o TCP
Telefonía por Internet	Normalmente proprietario	UDP o TCP
Administración de red	SNMP	Normalmente UDP
Traducción de nombres	DNS	Normalmente UDP

Figura 3.6 ♦ Aplicaciones de Internet populares y sus protocolos de transporte subyacentes.

Como se indica en la Figura 3.6, actualmente tanto UDP como TCP se utilizan con aplicaciones multimedia, como la telefonía por Internet, las videoconferencias en tiempo real y la reproducción de flujos de vídeos y audios almacenados. En el Capítulo 9 nos ocuparemos de estas aplicaciones. Por el momento, basta con mencionar que todas estas aplicaciones toleran la pérdida de una pequeña cantidad de paquetes, por lo que una transferencia de datos fiable no es absolutamente crítica para que la aplicación funcione correctamente. Además, las aplicaciones en tiempo real, como la telefonía por Internet y la videoconferencia, responden muy mal a los mecanismos de control de congestión de TCP. Por estas razones, los desarrolladores de aplicaciones multimedia pueden elegir ejecutar sus aplicaciones sobre UDP y no sobre TCP. Cuando la tasa de pérdidas de paquetes es baja y teniendo en cuenta que algunas organizaciones bloquean el tráfico UDP por razones de seguridad (véase el Capítulo 8), TCP se está convirtiendo en un protocolo cada vez más atractivo para el transporte de flujos multimedia.

Aunque hoy en día es común emplear UDP para ejecutar las aplicaciones multimedia, continúa siendo un tema controvertido. Como ya hemos dicho, UDP no proporciona mecanismos de control de congestión, y estos mecanismos son necesarios para impedir que la red entre en un estado de congestión en el que se realice muy poco trabajo útil. Si todo el mundo deseara reproducir flujos de vídeo a alta velocidad sin utilizar ningún mecanismo de control de congestión, se produciría tal desbordamiento de paquetes en los routers que muy pocos paquetes UDP lograrían recorrer con éxito la ruta entre el origen y el destino. Además, las altas tasas de pérdidas inducidas por los emisores UDP no controlados harían que los emisores TCP (que, como veremos, reducen sus velocidades de transmisión para hacer frente a la congestión) disminuyeran dramáticamente sus velocidades. Por tanto, la ausencia de un mecanismo de control de congestión en UDP puede dar lugar a altas tasas de pérdidas entre un emisor y un receptor UDP y al estrangulamiento de las sesiones TCP, lo cual es un problema potencialmente serio [Floyd 1999]. Muchos investigadores han propuesto nuevos mecanismos para forzar a todas las fuentes de datos, incluyendo las de tipo UDP, a llevar a cabo un control adaptativo de la congestión [Mahdavi 1997; Floyd 2000; Kohler 2006; RFC 4340].

Antes de pasar a examinar la estructura de los segmentos UDP, tenemos que comentar que es posible que una aplicación disponga de un servicio fiable de transferencia de datos utilizando UDP. Esto puede conseguirse si las características de fiabilidad se incorporan a la propia aplicación (por ejemplo, añadiendo mecanismos de reconocimiento y retransmisión, tales como los que vamos

a ver en la siguiente sección). Como ya hemos mencionado antes, el protocolo QUIC [Iyengar 2015] utilizado en el navegador Chrome de Google implementa la fiabilidad en un protocolo de la capa de aplicación por encima de UDP. Pero se trata de una tarea nada sencilla que requiere una intensa tarea de depuración de las aplicaciones. No obstante, incorporar los mecanismos de fiabilidad directamente en la aplicación permite que ella misma “se lo guise y se lo coma”. Es decir, los procesos de aplicación pueden comunicarse de forma fiable sin estar sujetos a las restricciones de la velocidad de transmisión impuestas por el mecanismo de control de congestión de TCP.

3.3.1 Estructura de los segmentos UDP

La estructura de los segmentos UDP, mostrada en la Figura 3.7, está definida en el documento RFC 768. Los datos de la aplicación ocupan el campo de datos del segmento UDP. Por ejemplo, para DNS, el campo de datos contiene un mensaje de consulta o un mensaje de respuesta. En el caso de una aplicación de flujo de audio, las muestras del audio llenarán el campo de datos. La cabecera UDP solo tiene cuatro campos, y cada uno de ellos tiene una longitud de dos bytes. Como se ha explicado en la sección anterior, los números de puerto permiten al host de destino pasar los datos de la aplicación al proceso apropiado que está ejecutándose en el sistema terminal de destino (es decir, realizar la función de demultiplexación). El campo de longitud especifica el número de bytes del segmento UDP (la cabecera más los datos). Es necesario un valor de longitud explícito ya que el tamaño del campo de datos puede variar de un segmento UDP al siguiente. El host receptor utiliza la suma de comprobación para detectar si se han introducido errores en el segmento. En realidad, la suma de comprobación también se calcula para unos pocos campos de la cabecera IP, además de para el segmento UDP. Pero por el momento no vamos a tener en cuenta este detalle para ver el bosque a través de los árboles. Veamos ahora cómo se calcula la suma de comprobación. En la Sección 6.2 se describen los principios básicos para la detección de errores. El campo longitud especifica la longitud del segmento UDP en bytes, incluyendo la cabecera.

3.3.2 Suma de comprobación de UDP

La suma de comprobación de UDP proporciona un mecanismo de detección de errores. Es decir, se utiliza para determinar si los bits contenidos en el segmento UDP han sido alterados según se desplazaban desde el origen hasta el destino (por ejemplo, a causa de la existencia de ruido en los enlaces o mientras estaban almacenados en un router). UDP en el lado del emisor calcula el complemento a 1 de la suma de todas las palabras de 16 bits del segmento, acarreando cualquier desbordamiento obtenido durante la operación de suma sobre el bit de menor peso. Este resultado se almacena en el campo suma de comprobación del segmento UDP. He aquí un ejemplo sencillo de cálculo de una suma de comprobación. Puede obtener información detallada acerca de una implementación eficiente del cálculo en el RFC 1071, así como de su rendimiento con datos reales en [Stone 1998; Stone 2000]. Por ejemplo, suponga que tenemos las siguientes tres palabras de 16 bits:

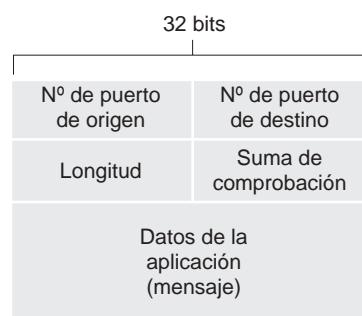


Figura 3.7 ♦ Estructura de un segmento UDP.

0110011001100000
0101010101010101
1000111100001100

La suma de las dos primeras palabras de 16 bits es:

0110011001100000
<u>0101010101010101</u>
1011101110110101

Sumando la tercera palabra a la suma anterior, obtenemos,

1011101110110101
<u>1000111100001100</u>
0100101011000010

Observe que en esta última suma se produce un desbordamiento, el cual se acarrea sobre el bit de menor peso. El complemento a 1 se obtiene convirtiendo todos los 0 en 1 y todos los 1 en 0. Por tanto, el complemento a 1 de la suma 0100101011000010 es 1011010100111101, que es la suma de comprobación. En el receptor, las cuatro palabras de 16 bits se suman, incluyendo la suma de comprobación. Si no se han introducido errores en el paquete, entonces la suma en el receptor tiene que ser 1111111111111111. Si uno de los bits es un 0, entonces sabemos que el paquete contiene errores.

Es posible que se esté preguntando por qué UDP proporciona una suma de comprobación, cuando muchos protocolos de la capa de enlace (incluyendo el popular protocolo Ethernet) también proporcionan mecanismos de comprobación de errores. La razón de ello es que no existe ninguna garantía de que todos los enlaces existentes entre el origen y el destino proporcionen un mecanismo de comprobación de errores; es decir, uno de los enlaces puede utilizar un protocolo de la capa de enlace que no proporcione comprobación de errores. Además, incluso si los segmentos se transfieren correctamente a través del enlace, es posible que se introduzcan errores de bit cuando un segmento se almacena en la memoria de un router. Dado que no están garantizadas ni la fiabilidad enlace a enlace, ni la detección de errores durante el almacenamiento en memoria, UDP tiene que proporcionar un mecanismo de detección de errores en la capa de transporte, *terminal a terminal*, si el servicio de transferencia de datos terminal a terminal ha de proporcionar la de detección de errores. Este es un ejemplo del famoso **principio terminal a terminal** del diseño de sistemas [Saltzer 1984], que establece que como cierta funcionalidad (en este caso, la detección de errores) debe implementarse terminal a terminal: “las funciones incluidas en los niveles inferiores pueden ser redundantes o escasamente útiles si se comparan con el coste de proporcionarlas en el nivel superior”.

Dado que IP está pensado para ejecutarse sobre prácticamente cualquier protocolo de la capa 2, resulta útil para la capa de transporte proporcionar un mecanismo de comprobación de errores como medida de seguridad. Aunque UDP proporciona un mecanismo de comprobación de errores, no hace nada para recuperarse del error. Algunas implementaciones de UDP simplemente descartan el segmento dañado y otras lo pasan a la aplicación junto con una advertencia.

Hasta aquí llega nuestra exposición sobre UDP. Pronto veremos que TCP ofrece a las aplicaciones un servicio de transferencia de datos fiable, así como otros servicios que UDP no proporciona. Naturalmente, TCP también es más complejo que UDP. Sin embargo, antes de abordar TCP, nos resultará útil volver unos pasos atrás y ocuparnos primero de los principios que subyacen a una transferencia de datos fiable.

3.4 Principios de un servicio de transferencia de datos fiable

En esta sección vamos a considerar el problema de la transferencia de datos fiable en un contexto general. Este enfoque es conveniente porque el problema de implementar servicios de transferencia de datos fiables no solo aparece en la capa de transporte, sino también en la capa de enlace y en la capa de aplicación. El problema general tiene por tanto una gran relevancia en las redes de computadoras. En efecto, si tuviéramos que identificar la lista de los diez problemas más importantes que afectan a las redes, este sería un candidato a encabezar dicha lista. En la siguiente sección examinaremos TCP y, en concreto, mostraremos que TCP aplica muchos de los principios que vamos a describir.

La Figura 3.8 ilustra el marco de trabajo que vamos a emplear en nuestro estudio sobre la transferencia de datos fiable. La abstracción del servicio proporcionada a las entidades de la capa superior es la de un canal fiable a través del cual se pueden transferir datos. Disponiendo de un canal fiable, ninguno de los bits de datos transferidos está corrompido (cambia de 0 a 1, o viceversa) ni se pierde, y todos se entregan en el orden en que fueron enviados. Este es precisamente el modelo de servicio ofrecido por TCP a las aplicaciones de Internet que lo invocan.

Es la responsabilidad de un **protocolo de transferencia de datos fiable** implementar esta abstracción del servicio. Esta tarea es complicada por el hecho de que la capa que hay *por debajo* del protocolo de transferencia de datos puede ser no fiable. Por ejemplo, TCP es un protocolo de transferencia de datos fiable que se implementa encima de una capa de red terminal a terminal no fiable (IP). De forma más general, la capa que hay debajo de los dos puntos terminales que se

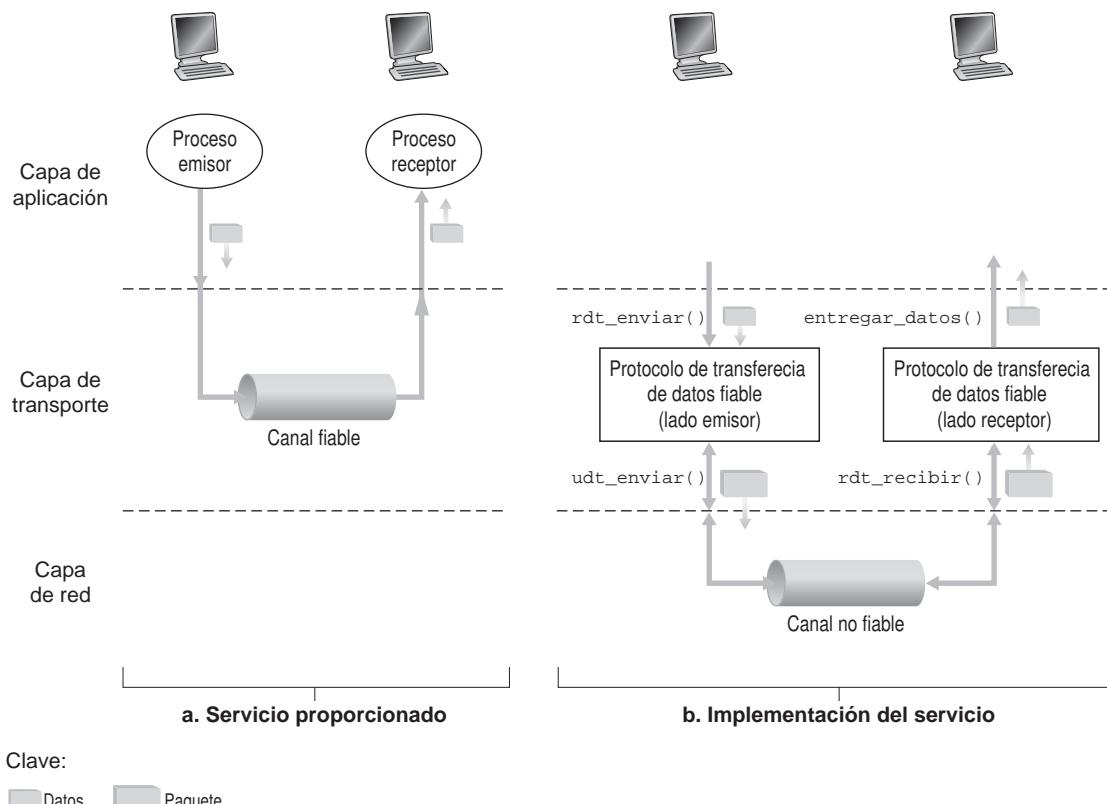


Figura 3.8 ♦ Transferencia de datos fiable: modelo del servicio e implementación del servicio.

comunican de forma fiable puede ser un único enlace físico (como en el caso de un protocolo de transferencia de datos a nivel de enlace) o una interred global (como en el caso de un protocolo del nivel de transporte). Para nuestros propósitos, sin embargo, podemos considerar esta capa inferior simplemente como un canal punto a punto no fiable.

En esta sección vamos a desarrollar de forma incremental los lados del emisor y del receptor de un protocolo de transferencia de datos fiable, considerando modelos cada vez más complejos del canal subyacente. Consideremos por ejemplo qué mecanismos del protocolo son necesarios cuando el canal subyacente puede corromper los bits o perder paquetes completos. Una suposición que vamos a adoptar a lo largo de esta exposición es que los paquetes se suministraran en el orden en que fueron enviados, siendo posible que algunos paquetes se pierdan; es decir, es decir, el canal subyacente no reordena los paquetes. La Figura 3.8(b) ilustra las interfaces de nuestro protocolo de transferencia de datos. El lado emisor del protocolo de transferencia de datos será invocado desde la capa superior mediante una llamada a `rdt_enviar()`, que pasará los datos que haya que entregar a la capa superior en el lado receptor. Aquí `rdt` hace referencia al protocolo de transferencia de datos fiable (*reliable data transfer*) y `_enviar` indica que el lado emisor de `rdt` está siendo llamado. (¡El primer paso para desarrollar un buen protocolo es elegir un buen nombre!) En el lado receptor, `rdt_recibir()` será llamado cuando llegue un paquete desde el lado receptor del canal. Cuando el protocolo `rdt` desea suministrar datos a la capa superior, lo hará llamando a `entregar_datos()`. De aquí en adelante utilizaremos el término “paquete” en lugar de “segmento” de la capa de transporte. Puesto que la teoría desarrollada en esta sección se aplica a las redes de computadoras en general y no solo a la capa de transporte de Internet, quizás resulte más apropiado el término genérico “paquete”.

En esta sección únicamente consideraremos el caso de la **transferencia de datos unidireccional**, es decir, los datos se transfieren desde el lado emisor al receptor. El caso de la **transferencia de datos bidireccional** (es decir, *full-duplex*) conceptualmente no es más difícil, pero es considerablemente más tediosa de explicar. Aunque solo abordemos la transferencia de datos unidireccional, tendremos en cuenta que los lados emisor y receptor de nuestro protocolo necesitan transmitir paquetes en *ambas* direcciones, como se indica en la Figura 3.8. Veremos brevemente que, además de intercambiar paquetes que contengan los datos que van a transferirse, los lados emisor y receptor de `rdt` también intercambian paquetes de control de una parte a otra. Ambos lados, emisor y receptor, de `rdt` envían paquetes al otro lado haciendo una llamada a `udt_enviar()` (donde `udt` hace referencia a una *transferencia de datos no fiable [unreliable data transfer]*).

3.4.1 Construcción de un protocolo de transferencia de datos fiable

Ahora vamos a ver una serie de protocolos de complejidad creciente, hasta llegar a un protocolo de transferencia de datos fiable sin defectos.

Transferencia de datos fiable sobre un canal totalmente fiable: `rdt1.0`

En primer lugar consideraremos el caso más simple, en el que el canal subyacente es completamente fiable. El protocolo en sí, que denominaremos `rdt1.0`, es trivial. En la Figura 3.9 se muestran las definiciones de las **máquinas de estados finitos (FSM, Finite-State Machine)** para el emisor y el receptor de `rdt1.0`. La máquina de estados finitos de la Figura 3.9(a) define el funcionamiento del emisor, mientras que la FSM de la Figura 3.9(b) define el funcionamiento del receptor. Es importante observar que existen máquinas de estados finitos *separadas* para el emisor y el receptor. Cada una de las máquinas de esta figura tiene solo un estado. Las flechas en la descripción de las FSM indican la transición del protocolo de un estado a otro. Puesto que en este caso cada una de las máquinas de la Figura 3.9 solo tiene un estado, necesariamente una transición es del estado a sí mismo (veremos diagramas más complicados enseguida). El suceso que provoca la transición se muestra encima de la línea horizontal que etiqueta la transición y las acciones que se toman cuando tiene lugar el suceso se indican debajo de la línea horizontal. Cuando no se lleve a cabo

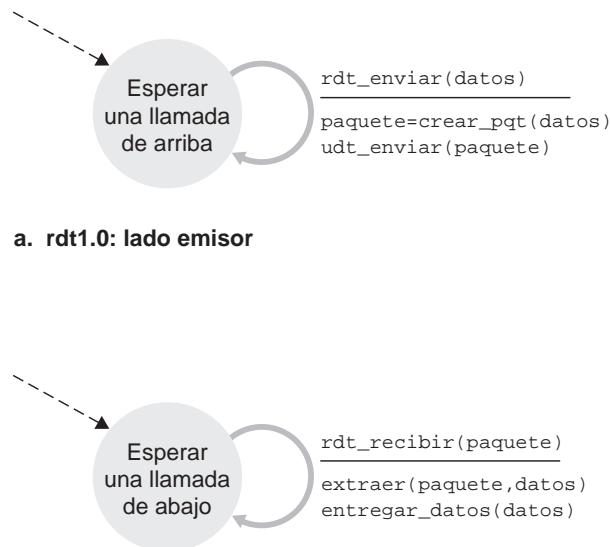


Figura 3.9 ♦ rdt1.0 – Protocolo para un canal totalmente fiable.

ninguna acción al ocurrir un suceso, o cuando no se produzca un suceso y se realice una acción, utilizaremos el símbolo Λ por debajo o por encima de la horizontal, respectivamente, para indicar de manera explícita la ausencia de una acción o de un suceso. El estado inicial de la máquina FSM está indicado mediante la línea de puntos. Aunque las máquinas de estados finitos de la Figura 3.9 tienen un único estado, las que veremos a continuación tendrán múltiples, por lo que es importante identificar el estado inicial de cada máquina FSM.

El lado emisor de rdt simplemente acepta datos de la capa superior a través del suceso `rdt_enviar(datos)`, crea un paquete que contiene los datos (mediante la acción `crear_pqt(datos)`) y envía el paquete al canal. En la práctica, el suceso `rdt_enviar(datos)` resultaría de una llamada a procedimiento (por ejemplo, a `rdt_enviar()`) realizada por la aplicación de la capa superior.

En el lado receptor, rdt recibe un paquete del canal subyacente a través del suceso `rdt_recibir(paqute)`, extrae los datos del paquete (mediante la acción `extraer(paqute, datos)`) y pasa los datos a la capa superior (mediante la acción `entregar_datos(datos)`). En la práctica, el suceso `rdt_recibir(paqute)` resultaría de una llamada a procedimiento (por ejemplo, a `rdt_recibir()`) desde el protocolo de la capa inferior.

En este protocolo tan simple no existe ninguna diferencia entre una unidad de datos y un paquete. Además, todo el flujo de paquetes va del emisor al receptor, ya que disponiendo de un canal totalmente fiable no existe la necesidad en el lado receptor de proporcionar ninguna realimentación al emisor, puesto que no hay nada que pueda ser incorrecto. Observe que también hemos supuesto que el receptor podía recibir los datos tan rápido como el emisor los enviara. Luego tampoco existe la necesidad de que el receptor le pida al emisor que vaya más despacio.

Transferencia de datos fiable sobre un canal con errores de bit: rdt2.0

Un modelo más realista del canal subyacente sería uno en el que los bits de un paquete pudieran corromperse. Normalmente, tales errores de bit se producen en los componentes físicos de una red cuando un paquete se transmite, se propaga o accede a un buffer. Por el momento vamos a continuar suponiendo que todos los paquetes transmitidos son recibidos (aunque sus bits pueden estar corrompidos) en el orden en que se enviaron.

Antes de desarrollar un protocolo que permita una comunicación fiable a través de un canal así, vamos a ver cómo las personas se enfrentan a esta situación. Imagine cómo dictaría un mensaje

largo a través del teléfono. En un escenario típico, la persona que escucha el mensaje podría decir “De acuerdo” después de cada frase que escuche, comprenda y apunte. Si la persona que escucha el mensaje no oye una frase, le pedirá que la repita. Este protocolo de dictado de mensajes utiliza tanto **reconocimientos positivos** (“De acuerdo”) como **reconocimientos negativos** (“Por favor, repita”). Estos mensajes de control permiten al receptor hacer saber al emisor qué es lo que ha recibido correctamente y qué ha recibido con errores y por tanto debe repetir. En una red de computadoras, los protocolos de transferencia de datos fiables basados en tales retransmisiones se conocen como **protocolos ARQ (Automatic Repeat reQuest, Solicitud automática de repetición)**.

En los protocolos ARQ se requieren, fundamentalmente, tres capacidades de protocolo adicionales para gestionar la presencia de errores de bit:

- *Detección de errores.* En primer lugar, se necesita un mecanismo que permita al receptor detectar que se han producido errores de bit. Recuerde de la sección anterior que UDP utiliza el campo de suma de comprobación de Internet precisamente para este propósito. En el Capítulo 6, examinaremos técnicas de detección y corrección de errores con más detalle; estas técnicas permiten al receptor detectar y, posiblemente, corregir los errores de bit en los paquetes. Por el momento, solo necesitamos saber que estas técnicas requieren que el emisor envíe al receptor bits adicionales (junto con los bits de los datos originales que se desean transferir) y dichos bits también se tendrán en cuenta para el cálculo de la suma de comprobación del paquete de datos `rdt2.0`.
- *Realimentación del receptor.* Dado que el emisor y el receptor normalmente se ejecutan en sistemas terminales diferentes, posiblemente separados por miles de kilómetros, la única forma de que el emisor sepa lo que ocurre en el receptor (en este caso, si un paquete ha sido recibido correctamente o no) es que el receptor envíe explícitamente información de realimentación al emisor. Las respuestas de acuse de recibo o reconocimiento positivo (ACK) y reconocimiento negativo (NAK) en el escenario del dictado de mensajes son ejemplos de esa realimentación. De forma similar, nuestro protocolo `rdt2.0` enviará paquetes ACK y NAK de vuelta desde el receptor al emisor. En principio, estos paquetes solo necesitan tener una longitud de un bit; por ejemplo, un valor 0 indicaría un reconocimiento negativo (NAK) y un valor 1 indicaría un reconocimiento positivo (ACK).
- *Retransmisión.* Un paquete que se recibe con errores en el receptor será retransmitido por el emisor.

La Figura 3.10 muestra la representación de la máquina de estados finitos para `rdt2.0`, un protocolo de transferencia de datos que dispone de mecanismos de detección de errores y paquetes de reconocimiento positivo y negativo.

El lado emisor de `rdt2.0` tiene dos estados. En el estado más a la izquierda, el protocolo del lado emisor está a la espera de datos procedentes de la capa superior. Cuando se produce el suceso `rdt_enviar(datos)`, el emisor creará un paquete (`pqtenv`) conteniendo los datos que van a ser transmitidos, junto con una suma de comprobación del paquete (como se ha visto en la Sección 3.3.2, por ejemplo, para el caso de un segmento UDP), y luego el paquete se envía mediante la operación `udt_enviar(pqtenv)`. En el estado más a la derecha, el protocolo del emisor está a la espera de un paquete de reconocimiento positivo ACK o negativo NAK procedente del receptor. Si se recibe un paquete ACK (la notación `rdt_recibir(pqtrcb) && esACK(pqtrcb)` mostrada en la Figura 3.10 corresponde a este suceso), el emisor sabe que el paquete más recientemente transmitido ha sido recibido correctamente y, por tanto, el protocolo vuelve al estado de espera de datos procedentes de la capa superior. Si se recibe un reconocimiento negativo NAK, el protocolo retransmite el último paquete y espera a recibir un mensaje ACK o NAK del receptor en respuesta al paquete de datos retransmitido. Es importante observar que cuando el emisor está en el estado “Esperar ACK o NAK”, *no puede* obtener más datos de la capa superior; es decir, el suceso `rdt_enviar()` no puede ocurrir; esto solo ocurrirá después de que el emisor reciba un ACK y salga de ese estado. Por tanto, el emisor no enviará ningún nuevo fragmento de datos hasta estar seguro

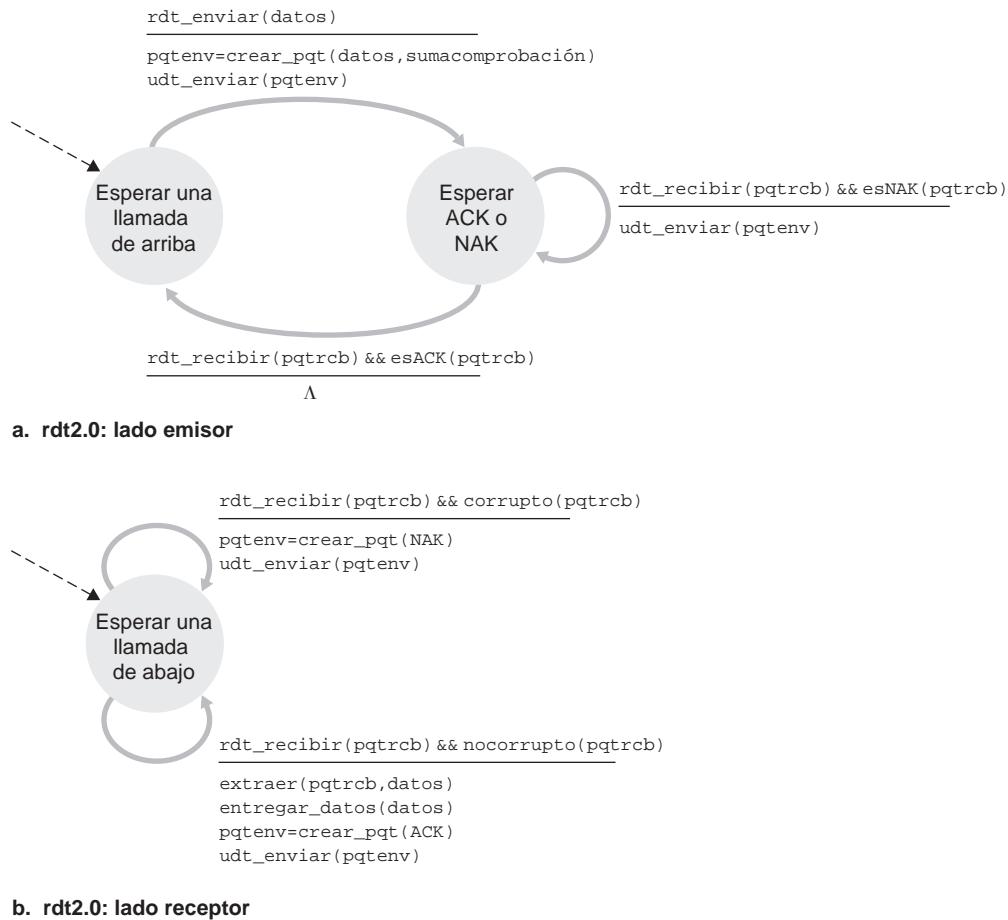


Figura 3.10 ♦ rdt2.0 – Protocolo para un canal con errores de bit.

de que el receptor ha recibido correctamente el paquete actual. Debido a este comportamiento, los protocolos como rdt2.0 se conocen como **protocolos de parada y espera (stop-and-wait protocol)**.

El lado receptor de la máquina de estados finitos para rdt2.0 tiene un único estado. Cuando llega un paquete, el receptor responde con un reconocimiento positivo ACK o negativo NAK, dependiendo de si el paquete recibido es correcto o está corrompido. En la Figura 3.10, la notación `rdt_recibir(pqtrcb) && corrupto(pqtrcb)` corresponde al suceso en el que se ha recibido un paquete y resulta ser erróneo.

Puede parecer que el protocolo rdt2.0 funciona pero, lamentablemente, tiene un defecto fatal. ¡No hemos tenido en cuenta la posibilidad de que el paquete ACK o NAK pueda estar corrompido! (Antes de continuar, debería pararse a pensar en cómo se podría resolver este problema.) Lamentablemente, este descuido no es tan inocuo como puede parecer. Como mínimo, tendremos que añadir bits de suma de comprobación a los paquetes ACK/NAK para detectar tales errores. La cuestión más complicada es cómo puede recuperarse el protocolo de los errores en los paquetes ACK o NAK. La dificultad está en que si un paquete ACK o NAK está corrompido, el emisor no tiene forma de saber si el receptor ha recibido o no correctamente el último fragmento de datos transmitido.

Consideraremos ahora tres posibilidades para gestionar los paquetes ACK o NAK corruptos:

- Para abordar la primera posibilidad, veamos lo que podría hacer una persona en el escenario del dictado de mensajes. Si la persona que está dictando el mensaje no entiende la respuesta “De acuerdo” o “Por favor, repita” del receptor, probablemente diría “¿Cómo dice?” (lo que introduce un nuevo tipo de paquete del emisor al receptor en nuestro protocolo). A continuación, el receptor repetiría la respuesta. ¿Pero qué ocurriría si el “Cómo dice” está corrompido? El receptor no tendría ni idea de si esa frase formaba parte del dictado o era una solicitud de que repitiera la última respuesta, por lo que probablemente respondería con un “¿Cómo dice usted?”. Y, a su vez, por supuesto, dicha respuesta también podría verse alterada. Evidentemente, el problema se complica.
- Una segunda alternativa consistiría en añadir los suficientes bits de suma de comprobación como para permitir al emisor no solo detectar, sino también recuperarse de los errores de bit. De este modo se resuelve el problema inmediato de un canal que puede corromper los paquetes de datos, pero no perderlos.
- Un tercer método consistiría simplemente en que el emisor reenviara el paquete de datos actual al recibir un paquete ACK o NAK alterado. Sin embargo, este método introduce **paquetes duplicados** en el canal emisor-receptor. La principal dificultad con los paquetes duplicados es que el receptor no sabe si el último paquete ACK o NAK enviado fue recibido correctamente en el emisor. Por tanto, *a priori*, no puede saber si un paquete entrante contiene datos nuevos o se trata de una retransmisión.

Una solución sencilla a este nuevo problema (y que ha sido adoptada en prácticamente todos los protocolos de transferencia de datos existentes, incluido TCP) consiste en añadir un nuevo campo al paquete de datos, y hacer que el emisor numere sus paquetes de datos colocando un **número de secuencia** en este campo. Entonces bastará con que el receptor compruebe ese número de secuencia para determinar si el paquete recibido es o no una retransmisión. Para el caso de este protocolo de parada y espera simple, un número de secuencia de 1 bit será suficiente, ya que le permitirá al receptor saber si el emisor está retransmitiendo el paquete previamente transmitido (el número de secuencia del paquete recibido tiene el mismo número de secuencia que el paquete recibido más recientemente) o si se trata de un paquete nuevo (el número de secuencia es distinto, está desplazado “hacia adelante” en aritmética de módulo 2). Puesto que estamos suponiendo que disponemos de un canal que no pierde datos, los paquetes ACK y NAK no tienen que indicar el número de secuencia del paquete que están confirmando. El emisor sabe que un paquete ACK o NAK recibido (esté alterado o no) fue generado en respuesta a su paquete de datos transmitido más recientemente.

Las Figuras 3.11 y 3.12 muestran la descripción de la máquina de estados finitos para `rdt2.1`, nuestra versión revisada de `rdt2.0`. Ahora las máquinas de estados finitos de los lados emisor y receptor de `rdt2.1` tienen el doble de estados que antes. Esto se debe a que ahora el estado del protocolo tiene que reflejar si el paquete que está enviando actualmente el emisor o el que está esperando el receptor tiene que incluir un número de secuencia igual a 0 o a 1. Observe que las acciones en aquellos casos en los que un paquete con un número de secuencia de 0 está siendo enviado o es esperado son imágenes especulares de aquellos casos en los que el número de secuencia del paquete es 1; las únicas diferencias se encuentran en la gestión del número de secuencia.

En el protocolo `rdt2.1`, el receptor envía tanto respuestas de reconocimiento positivo como negativo al emisor. Cuando recibe un paquete fuera de secuencia, el receptor envía un paquete ACK para el paquete que ha recibido. Cuando recibe un paquete corrompido, el receptor envía una respuesta de reconocimiento negativo. Podemos conseguir el mismo efecto que con una respuesta NAK si, en lugar de enviar una NAK, enviamos una respuesta de reconocimiento positivo (ACK) para el último paquete recibido correctamente. Un emisor que recibe dos respuestas ACK para el mismo paquete (es decir, recibe **respuestas ACK duplicadas**) sabe que el receptor no ha recibido correctamente el paquete que sigue al que está siendo reconocido (respuesta ACK) dos veces. Nuestro protocolo de transferencia de datos fiable sin respuestas de tipo NAK para un canal con errores de bit es `rdt2.2`, el cual se ilustra en las Figuras 3.13 y 3.14. Una util variación entre los protocolos `rdt2.1` y `rdt2.2` es que ahora el receptor tiene que incluir el número de secuencia

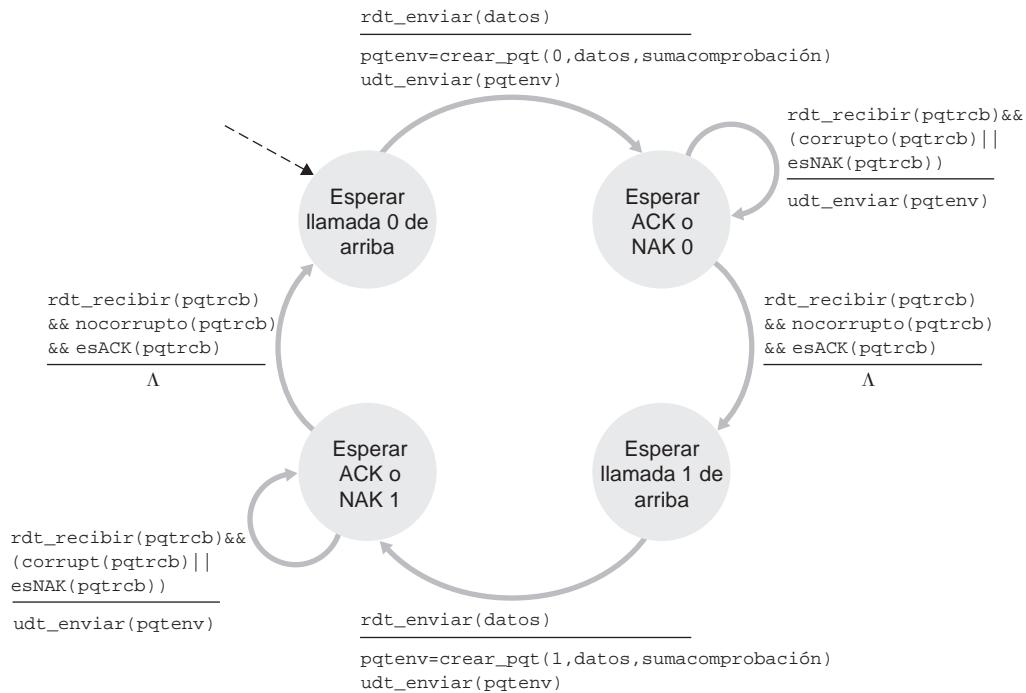


Figura 3.11 ♦ Lado emisor de rdt2.1

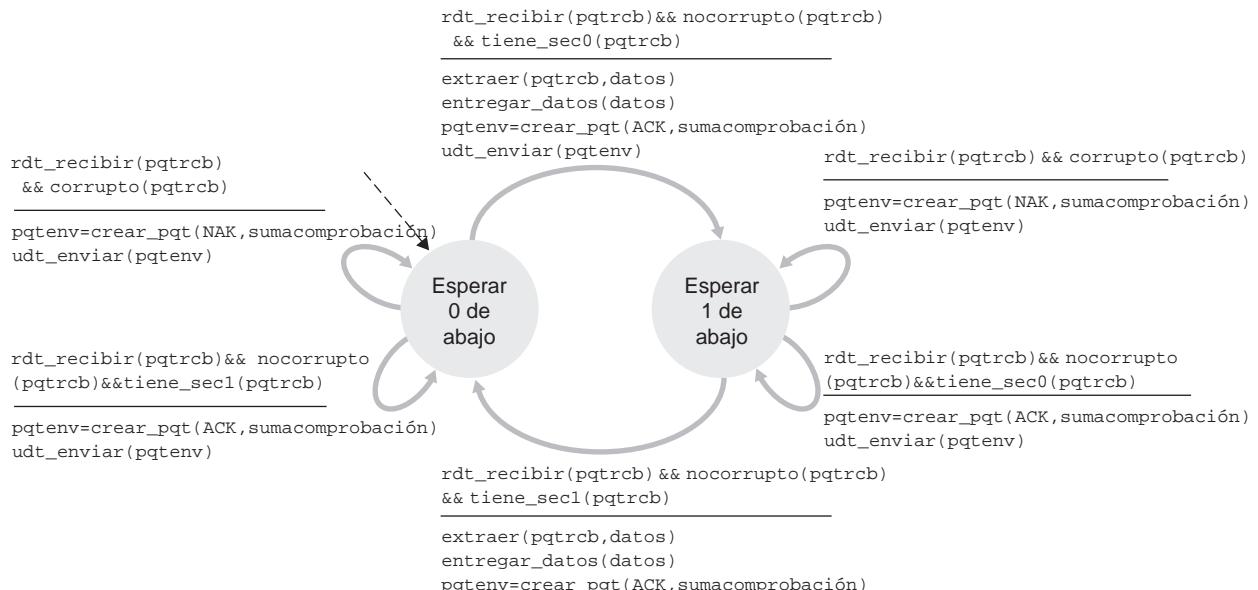
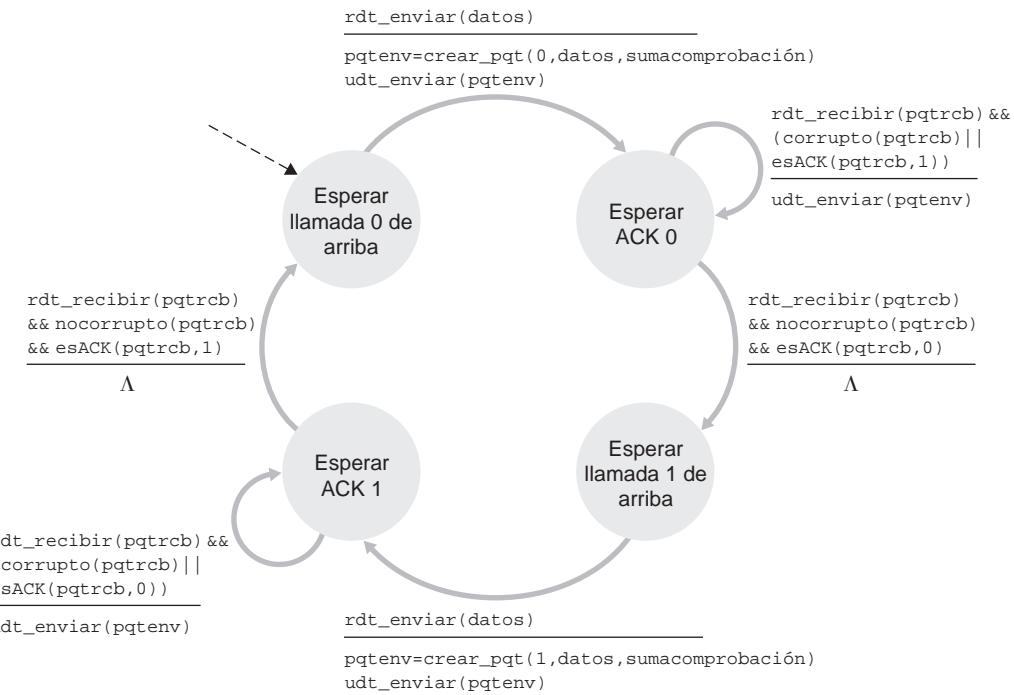
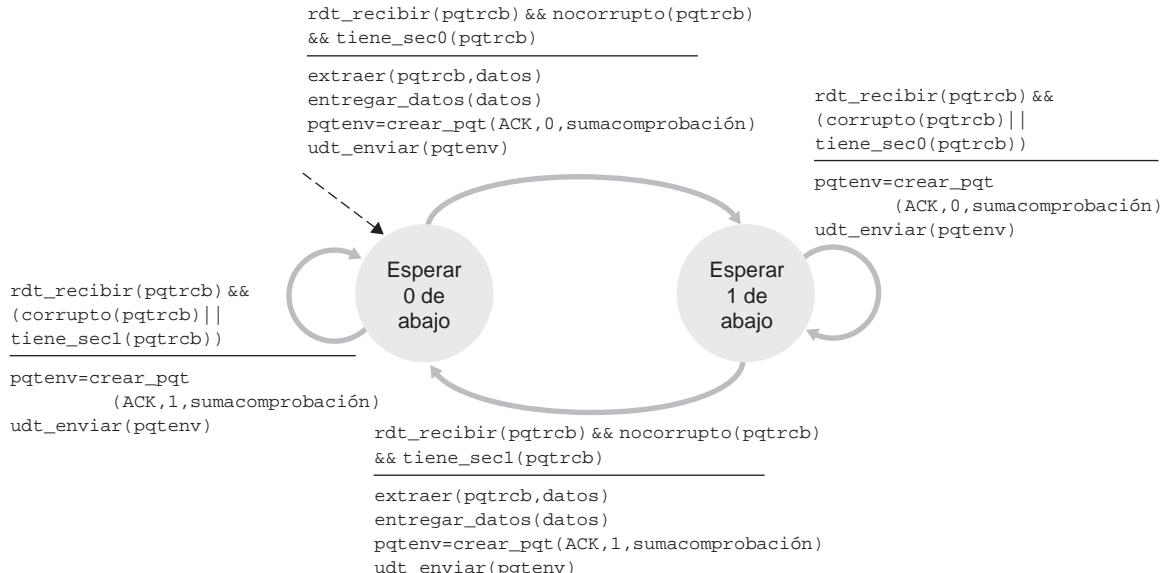


Figura 3.12 ♦ Lado receptor de rdt2.1

del paquete que está siendo confirmado mediante un mensaje ACK (lo que hace incluyendo el argumento ACK , 0 o ACK , 1 en `crear_pqt()` en la máquina de estados finitos de recepción), y el emisor tiene que comprobar el número de secuencia del paquete que está siendo confirmado por el mensaje ACK recibido (lo que se hace incluyendo el argumento 0 o 1 en `esACK()` en la máquina de estados finitos del emisor).

**Figura 3.13** ♦ Lado emisor de rdt2.2.**Figura 3.14** ♦ Lado receptor de rdt2.2**Transferencia de datos fiable sobre un canal con pérdidas y errores de bit: rdt3.0**

Suponga ahora que además de bits corrompidos, el canal subyacente también puede *perder* paquetes, un suceso no desconocido en las redes de computadoras de hoy en día (incluyendo Internet). Por tanto, ahora el protocolo tiene que preocuparse por dos problemas más: cómo detectar la pérdida de paquetes y qué hacer cuando se pierde un paquete. El uso de la suma

de comprobación (*checksum*), los números de secuencia, los paquetes ACK y la retransmisión de paquetes, técnicas que ya hemos desarrollado en el protocolo rdt2.2, nos van a permitir abordar este último problema. Para tratar el primero será necesario añadir un nuevo mecanismo de protocolo.

Hay disponibles muchas formas de abordar la pérdida de paquetes (varias de ellas se exploran en los ejercicios del final del capítulo). Veamos cómo el emisor puede detectar la pérdida de paquetes y cómo puede recuperarse de la misma. Suponga que el emisor transmite un paquete de datos y bien el propio paquete o el mensaje ACK del receptor para ese paquete se pierde. En cualquiera de estos dos casos, al emisor no le llega ninguna respuesta procedente del receptor. Si el emisor está dispuesto a esperar el tiempo suficiente como para *estar seguro* de que se ha perdido un paquete, simplemente puede retransmitirlo. Se puede comprobar de un modo sencillo que efectivamente este protocolo funciona.

Pero, ¿cuánto tiempo tiene que esperar el emisor para estar seguro de que se ha perdido un paquete? Es evidente que el emisor tiene que esperar al menos un tiempo igual al retardo de ida y vuelta entre el emisor y el receptor (lo que puede incluir el almacenamiento temporal en los buffers de los routers intermedios) más una cierta cantidad de tiempo que será necesaria para procesar un paquete en el receptor. En muchas redes, este retardo máximo del caso peor es muy difícil incluso de estimar y aún más de conocer con precisión. Además, idealmente, el protocolo debería recuperarse de la pérdida de paquetes tan pronto como fuera posible; pero si espera un tiempo igual al retardo en el caso peor, eso significa una larga espera hasta iniciar el mecanismo de recuperación de errores. Por tanto, el método que se adopta en la práctica es que el emisor seleccione juiciosamente un intervalo de tiempo tal que sea probable que un paquete se haya perdido, aunque no sea seguro que tal pérdida se haya producido. Si dentro de ese intervalo de tiempo no se ha recibido un ACK, el paquete se retransmite. Observe que si un paquete experimenta un retardo particularmente grande, el emisor puede retransmitirlo incluso aunque ni el paquete de datos ni su correspondiente ACK se hayan perdido. Esto introduce la posibilidad de que existan **paquetes de datos duplicados** en el canal emisor-receptor. Afortunadamente, el protocolo rdt2.2 ya dispone de la funcionalidad (los números de secuencia) para afrontar la existencia de paquetes duplicados.

Desde el punto de vista del emisor, la retransmisión es la solución para todo. El emisor no sabe si se ha perdido un paquete de datos, se ha perdido un mensaje ACK, o simplemente el paquete o el ACK están retardados. En todos los casos, la acción es la misma: retransmitir. La implementación de un mecanismo de retransmisión basado en el tiempo requiere un **temporizador de cuenta atrás** que pueda interrumpir al emisor después de que haya transcurrido un determinado periodo de tiempo. Por tanto, el emisor necesitará poder (1) iniciar el temporizador cada vez que envíe un paquete (bien por primera vez o en una retransmisión), (2) responder a una interrupción del temporizador (ejecutando las acciones apropiadas) y (3) detener el temporizador.

La Figura 3.15 muestra la máquina de estados finitos del emisor para rdt3.0, un protocolo que transfiere datos de forma fiable a través de un canal que puede corromper o perder paquetes; en los problemas de repaso, se le pedirá que defina la máquina de estados finitos del receptor para rdt3.0. La Figura 3.16 muestra cómo opera el protocolo cuando no se pierden ni se retardan los paquetes y cómo gestiona la pérdida de paquetes de datos. En la Figura 3.16, el tiempo va avanzando desde la parte superior del diagrama hacia la parte inferior del mismo; observe que el instante de recepción de un paquete es necesariamente posterior al instante de envío de un paquete como consecuencia de los retardos de transmisión y de propagación. En las Figuras 3.16(b)–(d), los corchetes en el lado del emisor indican los instantes de inicio y fin del temporizador. Algunos de los aspectos más sutiles de este protocolo se verán en los ejercicios incluidos al final del capítulo. Dado que los números de secuencia de los paquetes alternan entre 0 y 1, el protocolo rdt3.0 se denomina en ocasiones **protocolo de bit alternante**.

Hasta aquí hemos ensamblado los elementos clave de un protocolo de transferencia de datos. Las sumas de comprobación, los números de secuencia, los temporizadores y los paquetes de reconocimiento positivo y negativo desempeñan un papel fundamental y necesario en el funcionamiento del protocolo. A continuación vamos a trabajar con un protocolo de transferencia de datos fiable.



Nota de video

Desarrollo de un protocolo y representación de una FSM para un protocolo de la capa de aplicación simple.

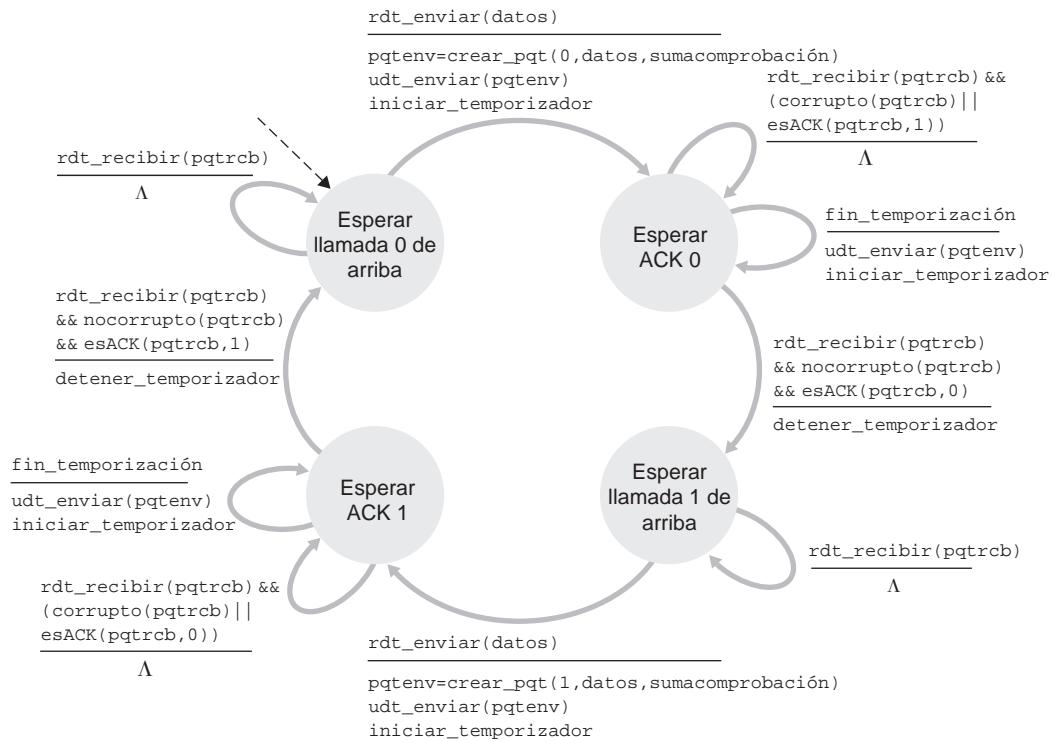


Figura 3.15 ♦ Lado emisor de rdt3.0

3.4.2 Protocolo de transferencia de datos fiable con procesamiento en cadena

El protocolo rdt3.0 es un protocolo funcionalmente correcto, pero es muy improbable que haya alguien a quien satisfaga su rendimiento, especialmente en las redes de alta velocidad actuales. La base del problema del rendimiento de rdt3.0 se encuentra en el hecho de que es un protocolo de parada y espera.

Para entender el impacto sobre el rendimiento de este comportamiento de parada y espera, vamos a considerar un caso ideal de dos hosts, uno localizado en la costa oeste de Estados Unidos y el otro en la costa este, como se muestra en la Figura 3.17. El retardo de propagación de ida y vuelta, RTT, a la velocidad de la luz entre estos dos sistemas terminales es aproximadamente igual a 30 milisegundos. Suponga que están conectados mediante un canal cuya velocidad de transmisión, R , es de 1 Gbps (10^9 bits por segundo). Con un tamaño de paquete, L , de 1.000 bytes (8.000 bits) por paquete, incluyendo los campos de cabecera y los datos, el tiempo necesario para transmitir el paquete por un enlace de 1 Gbps es:

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bits/paquete}}{10^9 \text{ bits/segundo}} = 8 \text{ microsegundos}$$

La Figura 3.18(a) muestra que, con nuestro protocolo de parada y espera, si el emisor comienza a transmitir el paquete en el instante $t = 0$, entonces en el instante $t = L/R = 8$ microsegundos el último bit entra en el canal en el lado del emisor. El paquete entonces tarda 15 milisegundos en atravesar el país, emergiendo el último bit del paquete en el lado del receptor en el instante $t = RTT/2 + L/R = 15,008$ milisegundos. Con el fin de simplificar, vamos a suponer que los

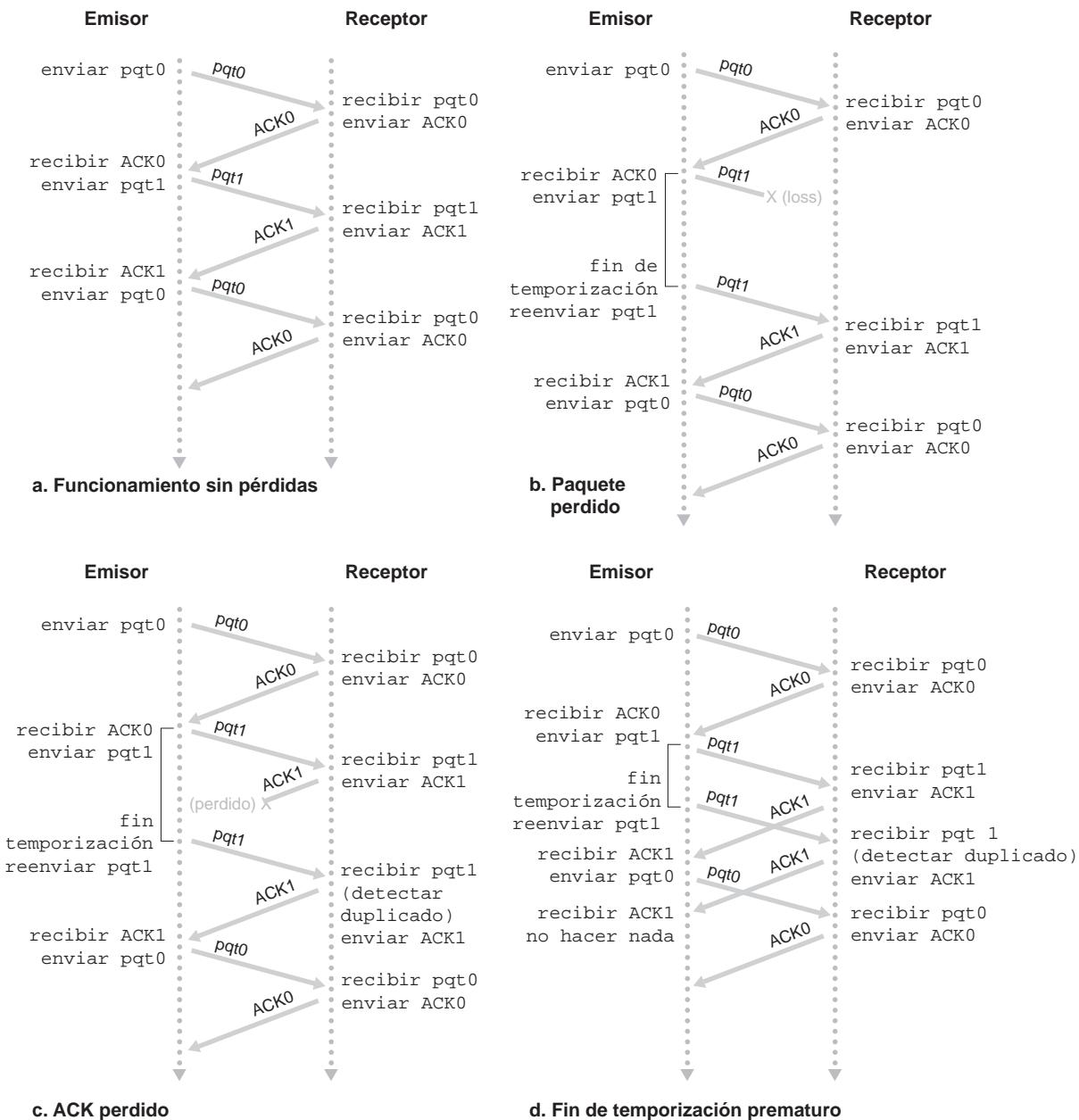


Figura 3.16 ♦ Funcionamiento de rdt3.0, el protocolo de bit alternante.

paquetes de reconocimiento ACK son extremadamente pequeños (por lo que podemos ignorar su tiempo de transmisión) y que el receptor puede enviar un ACK tan pronto como ha recibido el último bit de un paquete de datos, llegando dicho ACK al emisor en $t = RTT + L/R = 30,008$ ms. En esta situación, ahora el emisor puede transmitir el siguiente mensaje. Por tanto, en 30,008 milisegundos, el emisor ha estado transmitiendo durante solo 0,008 milisegundos. Si definimos la tasa de **utilización** del emisor (o del canal) como la fracción de tiempo que el emisor está realmente ocupado enviando bits al canal, el análisis de la Figura 3.18(a) muestra que el protocolo de parada y espera tiene una tasa de utilización del emisor, U_{emisor} , bastante mala de

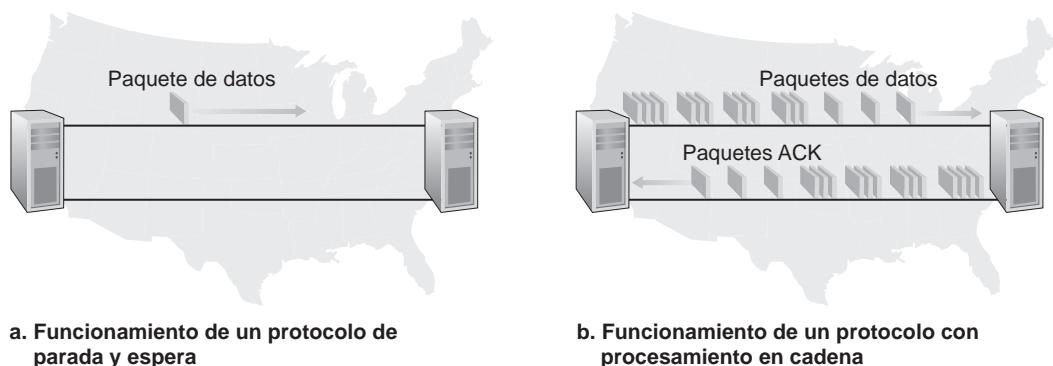


Figura 3.17 ♦ Protocolo de parada y espera y protocolo con procesamiento en cadena.

$$U_{emisor} = \frac{L / R}{RTT + L / R} = \frac{0,008}{30,008} = 0,00027$$

Es decir, ¡el emisor solo está ocupado 2,7 diezmilésimas del tiempo! En otras palabras, el emisor solo ha podido enviar 1.000 bytes en 30,008 milisegundos, una tasa de transferencia efectiva de solo 267 kbps, incluso disponiendo de un enlace a 1 Gbps. Imagine al infeliz administrador de la red que ha pagado una fortuna por un enlace con una capacidad de un gigabit para obtener una tasa de transferencia de únicamente 267 kilobits por segundo. Este es un ejemplo gráfico de cómo los protocolos de red pueden limitar las capacidades proporcionadas por el hardware de red subyacente. Además, hemos despreciado los tiempos de procesamiento del protocolo de la capa inferior tanto en el emisor como en el receptor, así como los retardos de procesamiento y de cola que pueden tener lugar en cualquiera de los routers intermedios existentes entre el emisor y el receptor. La inclusión de estos efectos solo serviría para incrementar el retardo y acentuar más su pésimo rendimiento.

La solución a este problema de rendimiento concreto es simple: en lugar de operar en el modo parada y espera, el emisor podría enviar varios paquetes sin esperar a los mensajes de reconocimiento, como se ilustra en la Figura 3.17(b). La Figura 3.18(b) muestra que si el emisor transmite tres paquetes antes de tener que esperar a los paquetes de reconocimiento, la utilización del emisor prácticamente se triplica. Dado que los muchos paquetes que están en tránsito entre el emisor y el receptor pueden visualizarse como el relleno de un conducto (*pipeline*), esta técnica se conoce como **pipelining** o **procesamiento en cadena**. El procesamiento en cadena tiene las siguientes consecuencias en los protocolos de transferencia de datos fiables:

- El rango de los números de secuencia tiene que incrementarse, dado que cada paquete en tránsito (sin contar las retransmisiones) tiene que tener un número de secuencia único y pueden coexistir múltiples paquetes en tránsito que no hayan sido confirmados mediante un reconocimiento.
- Los lados emisor y receptor de los protocolos pueden tener que almacenar en buffer más de un paquete. Como mínimo, el emisor tendrá en el buffer los paquetes que han sido transmitidos pero que todavía no han sido reconocidos. Como veremos enseguida, también puede ser necesario almacenar en el buffer del receptor los paquetes recibidos correctamente.
- El rango necesario de los números de secuencia y los requisitos de buffer dependerán de la forma en que un protocolo de transferencia de datos responda a la pérdida de paquetes y a los paquetes corrompidos o excesivamente retardados. Hay disponibles dos métodos básicos que permiten la recuperación de errores mediante procesamiento en cadena: **Retroceder N** y la **repetición selectiva**.

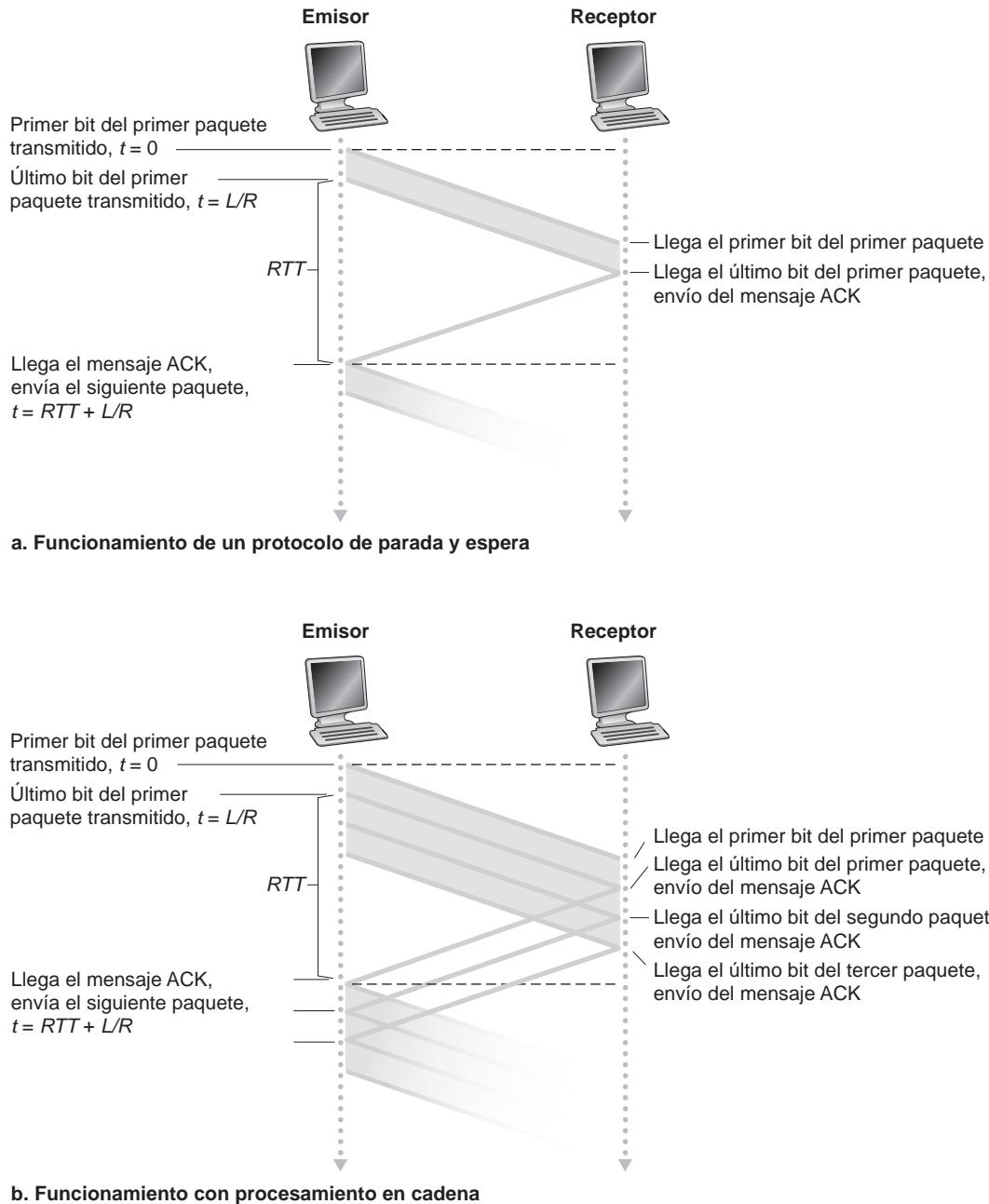


Figura 3.18 ♦ Proceso de transmisión con un protocolo de parada y espera y un protocolo con procesamiento en cadena.

3.4.3 Retroceder N (GBN)

En un **protocolo GBN (Go-Back-N, Retroceder N)**, el emisor puede transmitir varios paquetes (si están disponibles) sin tener que esperar a que sean reconocidos, pero está restringido a no tener más de un número máximo permitido, N , de paquetes no reconocidos en el canal. En esta sección vamos a describir el protocolo GBN con cierto detalle. Pero antes de seguir leyendo, le animamos a practicar con el applet GBN (¡un applet impresionante!) disponible en el sitio web del libro.

La Figura 3.19 muestra la parte correspondiente al emisor del rango de los números de secuencia en un protocolo GBN. Si definimos la *base* como el número de secuencia del paquete no reconocido más antiguo y *signumsec* como el número de secuencia más pequeño no utilizado (es decir, el número de secuencia del siguiente paquete que se va a enviar), entonces se pueden identificar los cuatro intervalos en rango de los números de secuencia. Los números de secuencia pertenecientes al intervalo $[0, \text{base}-1]$ corresponden a paquetes que ya han sido transmitidos y reconocidos. El intervalo $[\text{base}, \text{signumsec}-1]$ corresponde a paquetes que ya han sido enviados pero todavía no se han reconocido. Los números de secuencia del intervalo $[\text{signumsec}, \text{base}+N-1]$ se pueden emplear para los paquetes que pueden ser enviados de forma inmediata, en caso de que lleguen datos procedentes de la capa superior. Y, por último, los números de secuencia mayores o iguales que $\text{base}+N$ no pueden ser utilizados hasta que un paquete no reconocido que se encuentre actualmente en el canal sea reconocido (específicamente, el paquete cuyo número de secuencia sea igual a *base*).

Como sugiere la Figura 3.19, el rango de los números de secuencia permitidos para los paquetes transmitidos pero todavía no reconocidos puede visualizarse como una ventana de tamaño N sobre el rango de los números de secuencia. Cuando el protocolo opera, esta ventana se desplaza hacia adelante sobre el espacio de los números de secuencia. Por esta razón, N suele denominarse **tamaño de ventana** y el propio protocolo GBN se dice que es un **protocolo de ventana deslizante**. Es posible que se esté preguntando, para empezar, por qué debemos limitar a N el número de paquetes no reconocidos en circulación. ¿Por qué no permitir un número ilimitado de tales paquetes? En la Sección 3.5 veremos que el control de flujo es una de las razones para imponer un límite en el emisor. En la Sección 3.7 examinaremos otra razón al estudiar el mecanismo de control de congestión de TCP.

En la práctica, el número de secuencia de un paquete se incluye en un campo de longitud fija de la cabecera del paquete. Si k es el número de bits contenido en el campo que especifica el número de secuencia del paquete, el rango de los números de secuencia será $[0, 2^k - 1]$. Con un rango finito de números de secuencia, todas las operaciones aritméticas que impliquen a los números de secuencia tendrán que efectuarse utilizando aritmética en módulo 2^k . (Es decir, el espacio de números de secuencia puede interpretarse como un anillo de tamaño 2^k , donde el número de secuencia $2^k - 1$ va seguido por el número de secuencia 0.) Recuerde que el protocolo rdt3.0 dispone de un número de secuencia de 1 bit y de un rango de números de secuencia de $[0, 1]$. Algunos de los problemas incluidos al final del capítulo exploran las consecuencias de disponer de un rango finito de números de secuencia. En la Sección 3.5 veremos que TCP utiliza un campo para el número de secuencia de 32 bits, donde los números de secuencia de TCP cuentan los bytes del flujo de datos, en lugar de los paquetes.

Las Figuras 3.20 y 3.21 proporcionan una descripción ampliada de la máquina de estados finitos de los lados emisor y receptor de un protocolo GBN basado en paquetes de reconocimiento ACK y que no emplea paquetes de reconocimiento NAK. Nos referiremos a esta descripción de una FSM como *FSM ampliada*, ya que hemos añadido variables (similares a las variables de un lenguaje de programación) para *base* y *signumsec*, y operaciones sobre dichas variables y acciones condicionales que las implican. Observe que la especificación de una FSM ampliada empieza a asemejarse a una especificación de un lenguaje de programación. [Bochman 1984]

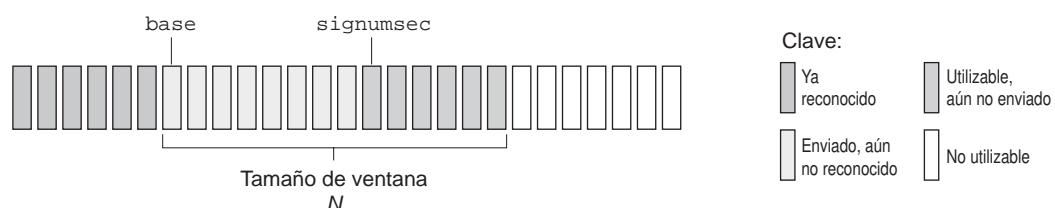


Figura 3.19 ♦ Números de secuencia en el emisor en el protocolo Retroceder N.

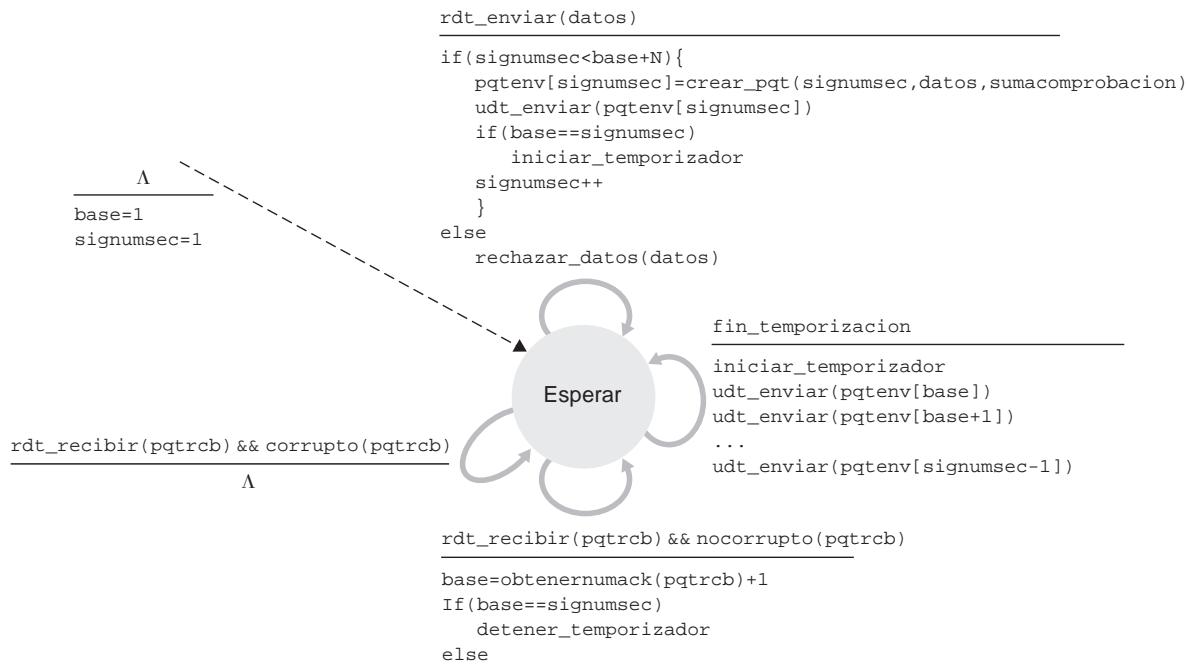


Figura 3.20 ♦ Descripción de la FSM ampliada del lado emisor de GBN.

proporciona un excelente repaso de otras ampliaciones de las técnicas de máquinas de estados finitos, así como de técnicas basadas en lenguajes de programación para la especificación de protocolos.

El emisor del protocolo GBN tiene que responder a tres tipos de sucesos:

- *Invocación desde la capa superior.* Cuando se llama a `rdt_enviar()` desde la capa superior, lo primero que hace el emisor es ver si la ventana está llena; es decir, si hay N paquetes no reconocidos en circulación. Si la ventana no está llena, se crea y se envía un paquete y se actualizan las variables de la forma apropiada. Si la ventana está llena, el emisor simplemente devuelve los datos a la capa superior, indicando de forma implícita que la ventana está llena. Probablemente entonces la capa superior volverá a intentarlo más tarde. En una implementación real, muy posiblemente el emisor almacenaría en el buffer estos datos (pero no los enviaría de forma inmediata) o dispondría de un mecanismo de sincronización (por ejemplo, un semáforo o un indicador) que permitiría a la capa superior llamar a `rdt_enviar()` solo cuando la ventana no estuviera llena.
- *Recepción de un mensaje de reconocimiento ACK.* En nuestro protocolo GBN, un reconocimiento de un paquete con un número de secuencia n implica un **reconocimiento acumulativo**, lo que indica que todos los paquetes con un número de secuencia menor o igual que n han sido correctamente recibidos por el receptor. Volveremos sobre este tema enseguida al examinar el lado receptor de GBN.
- *Un suceso de fin de temporización.* El nombre de este protocolo, “Retroceder N”, se deriva del comportamiento del emisor en presencia de paquetes perdidos o muy retardados. Como en los protocolos de parada y espera, se empleará un temporizador para recuperarse de la pérdida de paquetes de datos o de reconocimiento de paquetes. Si se produce un fin de temporización, el emisor reenvía *todos* los paquetes que haya transmitido anteriormente y que todavía no hayan sido reconocidos. El emisor de la Figura 3.20 utiliza un único temporizador, que puede interpretarse como un temporizador para los paquetes transmitidos pero todavía no reconocidos. Si

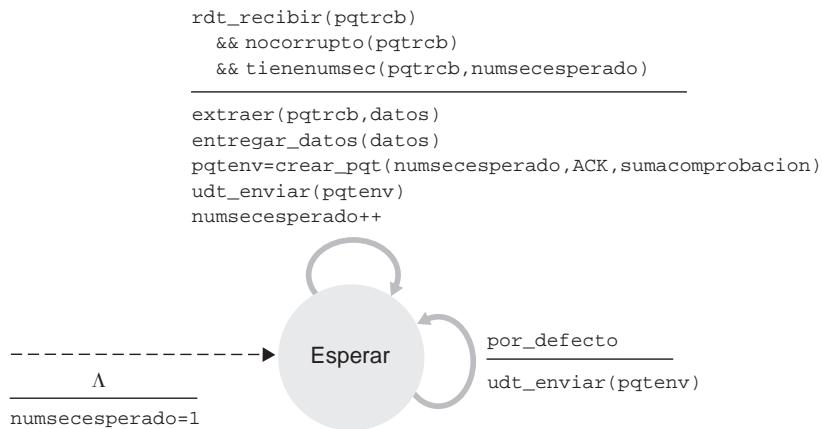


Figura 3.21 ♦ Descripción de la FSM ampliada del lado receptor de GBN.

se recibe un paquete ACK pero existen más paquetes transmitidos adicionales no reconocidos, entonces se reinicia el temporizador. Si no hay paquetes no reconocidos en circulación, el temporizador se detiene.

Las acciones del receptor en el protocolo GBN también son simples. Si un paquete con un número de secuencia n se recibe correctamente y en orden (es decir, los últimos datos entregados a la capa superior proceden de un paquete con el número de secuencia $n - 1$), el receptor envía un paquete ACK para el paquete n y entrega la parte de los datos del paquete a la capa superior. En todos los restantes casos, el receptor descarta el paquete y reenvía un mensaje ACK para el paquete recibido en orden más recientemente. Observe que dado que los paquetes se entregan a la capa superior de uno en uno, si el paquete k ha sido recibido y entregado, entonces todos los paquetes con un número de secuencia menor que k también han sido entregados. Por tanto, el uso de confirmaciones acumulativas es una opción natural del protocolo GBN.

En nuestro protocolo GBN, el receptor descarta los paquetes que no están en orden. Aunque puede parecer algo tonto y una pérdida de tiempo descartar un paquete recibido correctamente (pero desordenado), existe una justificación para hacerlo. Recuerde que el receptor debe entregar los datos en orden a la capa superior. Suponga ahora que se espera el paquete n , pero llega el paquete $n + 1$. Puesto que los datos tienen que ser entregados en orden, el receptor *podría* guardar en el buffer el paquete $n + 1$ y luego entregar ese paquete a la capa superior después de haber recibido y entregado el paquete n . Sin embargo, si se pierde el paquete n , tanto él como el paquete $n + 1$ serán retransmitidos como resultado de la regla de retransmisión del protocolo GBN en el lado de emisión. Por tanto, el receptor puede simplemente descartar el paquete $n + 1$. La ventaja de este método es la simplicidad del almacenamiento en el buffer del receptor (el receptor no necesita almacenar en el buffer *ninguno* de los paquetes entregados desordenados). Por tanto, mientras el emisor tiene que mantener los límites inferior y superior de su ventana y la posición de `signumsec` dentro de esa ventana, el único fragmento de información que el receptor debe mantener es el número de secuencia del siguiente paquete en orden. Este valor se almacena en la variable `numsecesperado`, como se muestra en la máquina de estados finitos del receptor de la Figura 3.21. Por supuesto, la desventaja de descartar un paquete correctamente recibido es que la subsiguiente retransmisión de dicho paquete puede perderse o alterarse y, por tanto, ser necesarias aún más retransmisiones.

La Figura 3.22 muestra el funcionamiento del protocolo GBN para el caso de un tamaño de ventana de cuatro paquetes. A causa de esta limitación del tamaño de ventana, el emisor transmite los paquetes 0 a 3, pero tiene que esperar a que uno o más de estos paquetes sean reconocidos antes de continuar. A medida que se reciben los sucesivos paquetes ACK (por ejemplo, ACK0 y ACK1), la

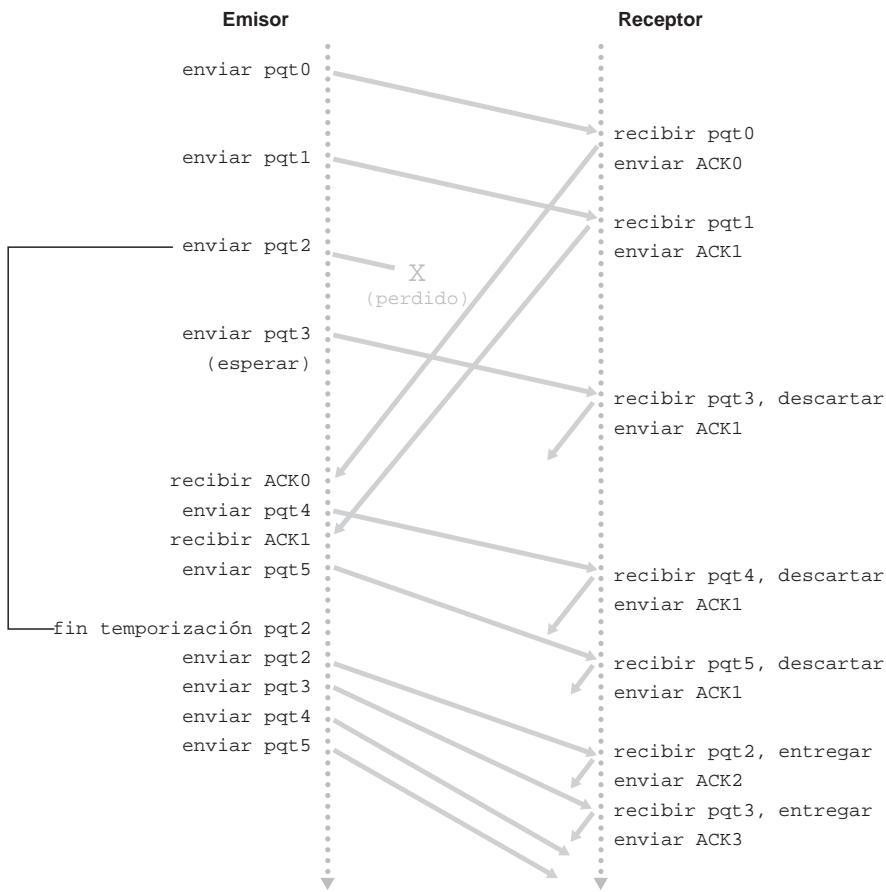


Figura 3.22 ♦ Funcionamiento del protocolo GBN (retroceder N).

ventana se desplaza hacia adelante y el emisor puede transmitir un nuevo paquete (pqt4 y pqt5, respectivamente). En el lado receptor se pierde el paquete 2 y, por tanto, los paquetes 3, 4 y 5 no se reciben en el orden correcto y, lógicamente, se descartan.

Antes de terminar esta exposición acerca de GBN, merece la pena destacar que una implementación de este protocolo en una pila de protocolos tendría probablemente una estructura similar a la de la máquina de estados finitos ampliada de la Figura 3.20. Posiblemente, la implementación se realizaría mediante varios procedimientos que implementasen las acciones que habría que realizar en respuesta a los distintos sucesos que pueden producirse. En una **programación basada en sucesos**, los diversos procedimientos son llamados (invocados) por otros procedimientos de la pila de protocolos, o como resultado de una interrupción. En el emisor, estos sucesos podrían ser (1) una llamada de una entidad de la capa superior para invocar `rdt_enviar()`, (2) una interrupción del temporizador y (3) una llamada de la capa inferior para invocar `rdt_recibir()` cuando llega un paquete. Los ejercicios sobre programación incluidos al final del capítulo le darán la oportunidad de implementar en la práctica estas rutinas en una red simulada, pero realista.

Observe que el protocolo GBN incorpora casi todas las técnicas que veremos en la Sección 3.5 al estudiar los componentes del servicio de transferencia de datos fiable de TCP. Estas técnicas incluyen el uso de números de secuencia, reconocimientos acumulativos, sumas de comprobación y una operación de fin de temporización/retransmisión.

3.4.4 Repetición selectiva (SR)

El protocolo GBN permite al emisor, en teoría, “llenar el conducto” mostrado en la Figura 3.17 con paquetes, evitando así los problemas de utilización del canal que hemos visto con los protocolos de parada y espera. Sin embargo, hay algunos escenarios en los que el propio GBN presenta problemas de rendimiento. En particular, cuando el tamaño de la ventana y el producto ancho de banda-retardo son grandes puede haber muchos paquetes en el canal. En este caso, un único paquete erróneo podría hacer que el protocolo GBN retransmitiera una gran cantidad de paquetes, muchos de ellos de forma innecesaria. A medida que la probabilidad de errores en el canal aumenta, este puede comenzar a llenarse con estas retransmisiones innecesarias. En nuestro escenario del dictado de mensajes, imagine que si cada vez que se altera una palabra, las 1.000 palabras que la rodean (por ejemplo, con un tamaño de ventana de 1.000 palabras) tuvieran que ser repetidas, el dictado se ralentizaría a causa de la repetición de palabras.

Como su nombre sugiere, los protocolos de repetición selectiva evitan las retransmisiones innecesarias haciendo que el emisor únicamente retransmita aquellos paquetes que se sospeche que llegaron al receptor con error (es decir, que se perdieron o estaban corrompidos). Esta retransmisión individualizada y necesaria requerirá que el receptor confirme *individualmente* qué paquetes ha recibido correctamente. De nuevo, utilizaremos una ventana de tamaño N para limitar el número de paquetes no reconocidos y en circulación en el canal. Sin embargo, a diferencia de GBN, el emisor ya habrá recibido mensajes ACK para algunos de los paquetes de la ventana. En la Figura 3.23 se muestra el espacio de números de secuencia del lado de emisión del protocolo SR. La Figura 3.24 detalla las distintas acciones que lleva a cabo el emisor del protocolo SR.

El receptor de SR confirmará que un paquete se ha recibido correctamente tanto si se ha recibido en el orden correcto como si no. Los paquetes no recibidos en orden se almacenarán en el buffer hasta que se reciban los paquetes que faltan (es decir, los paquetes con números de secuencia menores), momento en el que un lote de paquetes puede entregarse en orden a la capa superior. En la Figura 3.25 se enumeran las acciones tomadas por el receptor de SR. La Figura 3.26 muestra un

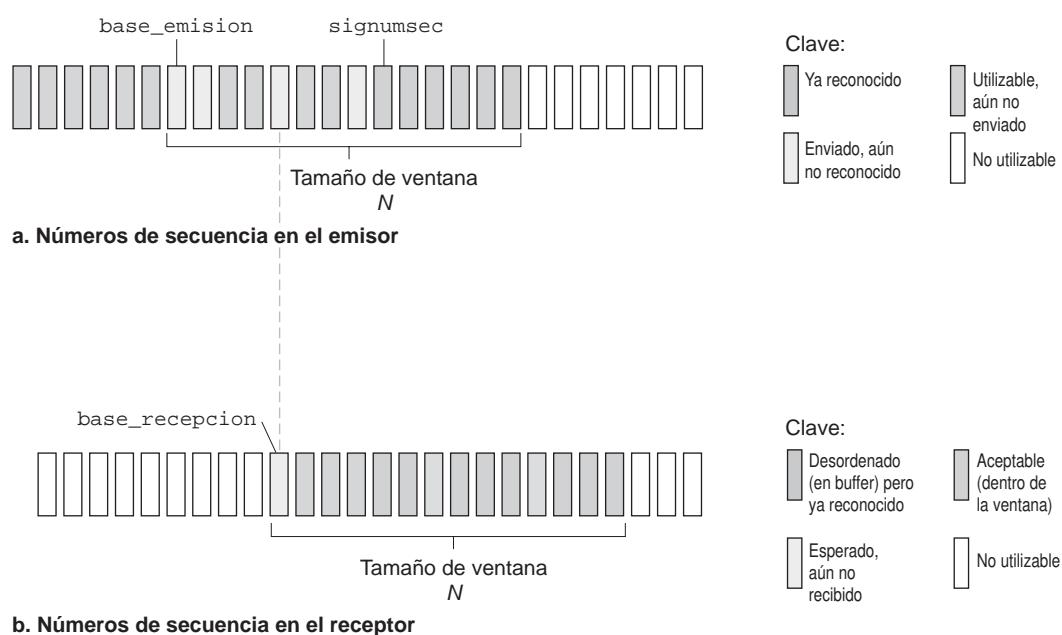


Figura 3.23 ♦ Espacios de números de secuencia del lado emisor y del lado receptor en el protocolo de repetición selectiva (SR).

1. *Datos recibidos de la capa superior.* Cuando se reciben datos de la capa superior, el emisor de SR comprueba el siguiente número de secuencia disponible para el paquete. Si el número de secuencia se encuentra dentro de la ventana del emisor, los datos se empaquetan y se envían; en caso contrario, bien se almacenan en el buffer o bien se devuelven a la capa superior para ser transmitidos más tarde, como en el caso del protocolo GBN.
2. *Fin de temporización.* De nuevo, se emplean temporizadores contra la pérdida de paquetes. Sin embargo, ahora, cada paquete debe tener su propio temporizador lógico, ya que solo se transmitirá un paquete al producirse el fin de la temporización. Se puede utilizar un mismo temporizador hardware para imitar el funcionamiento de varios temporizadores lógicos [Varghese 1997].
3. *ACK recibido.* Si se ha recibido un mensaje ACK, el emisor de SR marca dicho paquete como que ha sido recibido, siempre que esté dentro de la ventana. Si el número de secuencia del paquete es igual a `base_emision`, se hace avanzar la base de la ventana, situándola en el paquete no reconocido que tenga el número de secuencia más bajo. Si la ventana se desplaza y hay paquetes que no han sido transmitidos con números de secuencia que ahora caen dentro de la ventana, entonces esos paquetes se transmiten.

Figura 3.24 ♦ Sucesos y acciones en el lado emisor del protocolo SR.

1. *Se ha recibido correctamente un paquete cuyo número de secuencia pertenece al intervalo `[base_recepcion, base_recepcion+N-1]`.* En este caso, el paquete recibido cae dentro de la ventana del receptor y se devuelve al emisor un paquete ACK selectivo. Si el paquete no ha sido recibido con anterioridad, se almacena en el buffer. Si este paquete tiene un número de secuencia igual a la base de la ventana de recepción (`base_recepcion` en la Figura 3.22), entonces este paquete y cualquier paquete anteriormente almacenado en el buffer y numerado consecutivamente (comenzando por `base_recepcion`) se entregan a la capa superior. La ventana de recepción avanza entonces el número de paquetes suministrados entregados a la capa superior. Por ejemplo, en la Figura 3.26, cuando se recibe un paquete con el número de secuencia `base_recepcion = 2`, este y los paquetes 3, 4 y 5 pueden entregarse a la capa superior.
2. *Se ha recibido correctamente un paquete cuyo número de secuencia pertenece al intervalo `[base_recepcion-N, base_recepcion-1]`.* En este caso, se tiene que generar un mensaje ACK, incluso aunque ese paquete haya sido reconocido anteriormente por el receptor.
3. *En cualquier otro caso.* Ignorar el paquete.

Figura 3.25 ♦ Sucesos y acciones en el lado receptor del protocolo SR.

ejemplo del funcionamiento de SR en presencia de paquetes perdidos. Observe que, en esta figura, inicialmente el receptor almacena en el buffer los paquetes 3, 4 y 5, y luego los entrega, junto con el paquete 2, a la capa superior una vez que se ha recibido dicho paquete 2.

Es importante observar en el Paso 2 de la Figura 3.25 que el receptor vuelve a reconocer (en lugar de ignorar) los paquetes ya recibidos con determinados números de secuencia *inferiores* al número base actual de la ventana. Puede comprobar fácilmente que este doble reconocimiento es necesario. Por ejemplo, dados los espacios de números de secuencia del emisor y del receptor

de la Figura 3.23, si no hay ningún paquete ACK para el paquete `base_emision` propagándose desde el receptor al emisor, finalmente el emisor retransmitirá el paquete `base_emision`, incluso aunque esté claro (¡para nosotros, pero no para el emisor!) que el receptor ya ha recibido dicho paquete. Si el receptor no hubiera confirmado este paquete, la ventana del emisor nunca avanzaría. Este ejemplo ilustra un aspecto importante de los protocolos SR (y de otros muchos protocolos). El emisor y el receptor no siempre tienen una visión idéntica de lo que se ha recibido correctamente y de lo que no. En los protocolos SR, esto significa que las ventanas del emisor y del receptor no siempre coinciden.

La falta de sincronización entre las ventanas del emisor y del receptor tiene consecuencias importantes cuando nos enfrentamos con la realidad de un rango finito de números de secuencia. Por ejemplo, imagine lo que ocurriría con un rango de cuatro números de secuencia de paquete, 0, 1, 2, 3, y un tamaño de ventana de tres. Suponga que los paquetes 0 a 2 se transmiten y son recibidos correctamente y reconocidos por el receptor. En esta situación, la ventana del receptor se encontraría sobre los paquetes cuarto, quinto y sexto, que tienen los números de secuencia 3, 0 y 1, respectivamente. Ahora consideremos dos escenarios. En el primero, mostrado en la Figura 3.27(a), los mensajes ACK para los tres primeros paquetes se pierden y el emisor los retransmite. A continuación, el receptor recibe un paquete con el número de secuencia 0, una copia del primer paquete enviado.

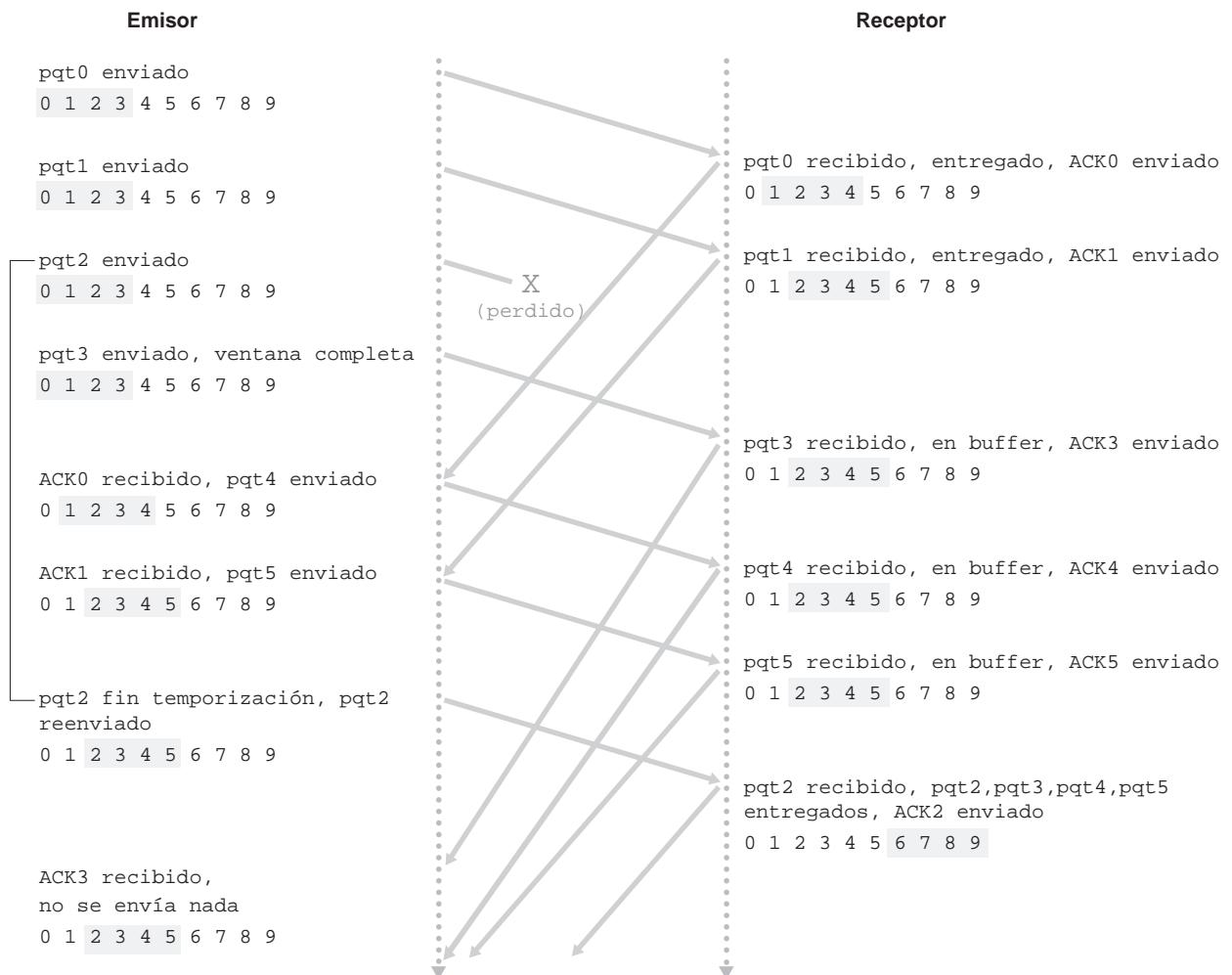


Figura 3.26 ♦ Funcionamiento del protocolo SR.

En el segundo escenario, mostrado en la Figura 3.27(b), los mensajes ACK correspondientes a los tres primeros paquetes se entregan correctamente. El emisor hace avanzar su ventana y envía los paquetes cuarto, quinto y sexto, con los número de secuencia 3, 0 y 1, respectivamente. El paquete con el número de secuencia 3 se pierde pero llega el paquete con el número de secuencia 0, un paquete que contiene datos *nuevos*.

Examinemos ahora el punto de vista del receptor en la Figura 3.27, el cual tiene delante una cortina imaginaria entre el emisor y el receptor, ya que el receptor no puede “ver” las acciones que lleva a cabo el emisor. Todo lo que ve el receptor es la secuencia de mensajes que recibe del canal y que él envía al mismo. En lo que respecta al receptor, los dos escenarios de la Figura 3.27 son *idénticos*. No hay forma de diferenciar la retransmisión del primer paquete de la transmisión inicial

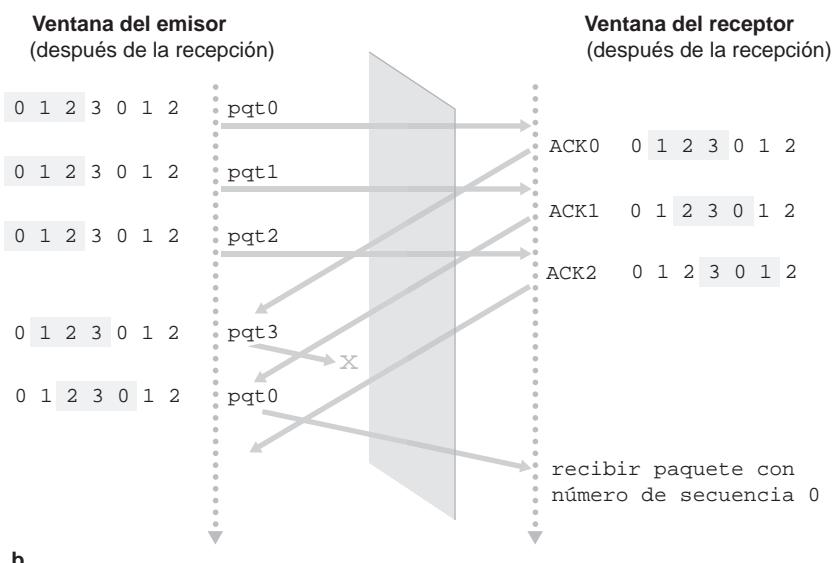
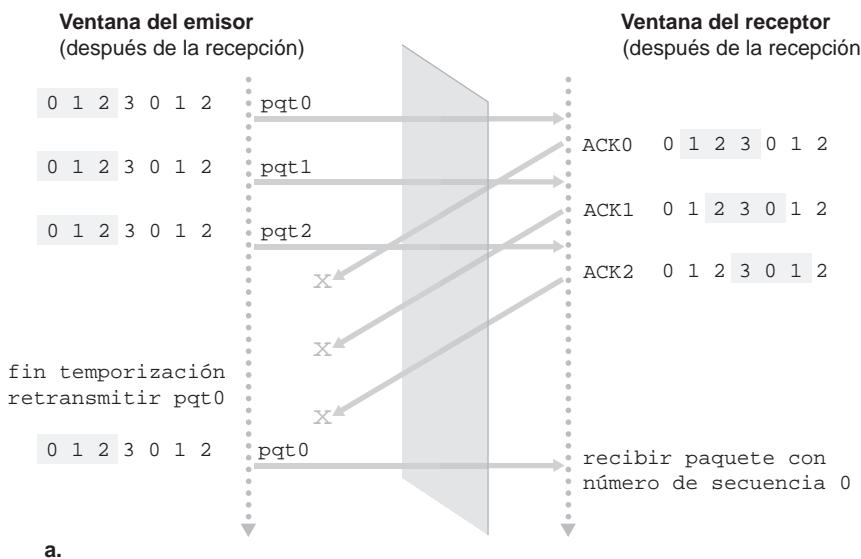


Figura 3.27 ♦ Dilema del receptor de los protocolos SR con ventanas demasiado grandes: ¿un nuevo paquete o una retransmisión?

del quinto paquete. Evidentemente, un tamaño de ventana que sea una unidad menor que el tamaño del espacio de números de secuencia no puede funcionar. Entonces, ¿cuál tiene que ser el tamaño de la ventana? Uno de los problemas incluidos al final del capítulo le pedirá que demuestre que el tamaño de la ventana tiene que ser menor o igual que la mitad del tamaño del espacio de números de secuencia en los protocolos SR.

En el sitio web de acompañamiento encontrará un applet que simula el funcionamiento del protocolo SR. Intente llevar a cabo los mismos experimentos que realizó con el applet de GBN. ¿Coincidieron los resultados con los que cabría esperar?

Con esto hemos terminado con nuestra exposición sobre los protocolos fiables de transferencia de datos. Hemos visto *muchos* de los numerosos mecanismos básicos que contribuyen a proporcionar una transferencia de datos fiable. La Tabla 3.1 resume estos mecanismos. Ahora que ya hemos visto todos estos mecanismos en funcionamiento y que hemos adquirido una “visión de conjunto”, le animamos a que repase esta sección con el fin de ver cómo estos mecanismos se fueron añadiendo para cubrir los modelos cada vez más complejos (y realistas) del canal que conecta al emisor y el receptor, o para mejorar el rendimiento de los protocolos.

Concluimos esta exposición sobre los protocolos de transferencia de datos fiables haciendo una suposición más sobre el modelo del canal subyacente. Recuerde que hemos supuesto que los paquetes no se pueden reordenar dentro del canal existente entre el emisor y el receptor. Generalmente, esta suposición es razonable cuando el emisor y el receptor están conectados simplemente mediante un cable físico. Sin embargo, cuando el “canal” que los conecta es una red puede tener lugar la reordenación de paquetes. Una manifestación de la reordenación de

Mecanismo	Uso, Comentarios
Suma de comprobación (Checksum)	Utilizada para detectar errores de bit en un paquete transmitido.
Temporizador	Se emplea para detectar el fin de temporización y retransmitir un paquete, posiblemente porque el paquete (o su mensaje ACK correspondiente) se ha perdido en el canal. Puesto que se puede producir un fin de temporización si un paquete está retardado pero no perdido (fin de temporización prematura), o si el receptor ha recibido un paquete pero se ha perdido el correspondiente ACK del receptor al emisor, puede ocurrir que el receptor reciba copias duplicadas de un paquete.
Número de secuencia	Se emplea para numerar secuencialmente los paquetes de datos que fluyen del emisor hacia el receptor. Los saltos en los números de secuencia de los paquetes recibidos permiten al receptor detectar que se ha perdido un paquete. Los paquetes con números de secuencia duplicados permiten al receptor detectar copias duplicadas de un paquete.
Reconocimiento (ACK)	El receptor utiliza estos paquetes para indicar al emisor que un paquete o un conjunto de paquetes ha sido recibido correctamente. Los mensajes de reconocimiento suelen contener el número de secuencia del paquete o los paquetes que están confirmando. Dependiendo del protocolo, los mensajes de reconocimiento pueden ser individuales o acumulativos.
Reconocimiento negativo (NAK)	El receptor utiliza estos paquetes para indicar al emisor que un paquete no ha sido recibido (NAK) correctamente. Normalmente, los mensajes de reconocimiento negativo contienen el número de secuencia de dicho paquete erróneo.
Ventana, procesamiento en cadena	El emisor puede estar restringido para enviar únicamente paquetes cuyo número de secuencia caiga dentro de un rango determinado. Permitiendo que se transmitan varios paquetes aunque no estén todavía reconocidos, se puede incrementar la tasa de utilización del emisor respecto al modo de operación de los protocolos de parada y espera. Veremos brevemente que el tamaño de la ventana se puede establecer basándose en la capacidad del receptor para recibir y almacenar en buffer los mensajes, o en el nivel de congestión de la red, o en ambos parámetros.

Tabla 3.1 ♦ Resumen de los mecanismos para la transferencia de datos fiable y su uso.

paquetes es que pueden aparecer copias antiguas de un paquete con un número de secuencia o de reconocimiento x , incluso aunque ni la ventana del emisor ni la del receptor contengan x . Con la reordenación de paquetes, puede pensarse en el canal como en un buffer que almacena paquetes y que espontáneamente puede transmitirlos en *cualquier* instante futuro. Puesto que los números de secuencia pueden reutilizarse, hay que tener cuidado para prevenir la aparición de esos paquetes duplicados. En la práctica, lo que se hace es asegurarse de que no se reutilice un número de secuencia hasta que el emisor esté “seguro” de que los paquetes enviados anteriormente con el número de secuencia x ya no se encuentran en la red. Esto se hace suponiendo que un paquete no puede “vivir” en la red durante más tiempo que un cierto periodo temporal máximo fijo. En las ampliaciones de TCP para redes de alta velocidad se supone un tiempo de vida máximo de paquete de aproximadamente tres minutos [RFC 1323]. [Sunshine 1978] describe un método para utilizar números de secuencia tales que los problemas de reordenación pueden ser eliminados por completo.

3.5 Transporte orientado a la conexión: TCP

Ahora que ya hemos visto los principios básicos de la transferencia de datos fiable, vamos a centrarnos en TCP, un protocolo de la capa de transporte de Internet, fiable y orientado a la conexión. En esta sección veremos que para proporcionar una transferencia de datos fiable, TCP confía en muchos de los principios básicos expuestos en la sección anterior, incluyendo los mecanismos de detección de errores, las retransmisiones, los reconocimientos acumulativos, los temporizadores y los campos de cabecera para los números de secuencia y de reconocimiento. El protocolo TCP está definido en los documentos RFC 793, RFC 1122, RFC 1323, RFC 2018 y RFC 2581.

3.5.1 La conexión TCP

Se dice que TCP está **orientado a la conexión** porque antes de que un proceso de la capa aplicación pueda comenzar a enviar datos a otro, los dos procesos deben primero “establecer una comunicación” entre ellos; es decir, tienen que enviarse ciertos segmentos preliminares para definir los parámetros de la transferencia de datos que van a llevar a cabo a continuación. Como parte del proceso de establecimiento de la conexión TCP, ambos lados de la misma iniciarán muchas variables de estado TCP (muchas de las cuales se verán en esta sección y en la Sección 3.7) asociadas con la conexión TCP.

La “conexión” TCP no es un circuito terminal a terminal con multiplexación TDM o FDM como lo es una red de comutación de circuitos. En su lugar, la “conexión” es una conexión lógica, con un estado común que reside solo en los niveles TCP de los dos sistemas terminales que se comunican. Dado que el protocolo TCP se ejecuta únicamente en los sistemas terminales y no en los elementos intermedios de la red (routers y switches de la capa de enlace), los elementos intermedios de la red no mantienen el estado de la conexión TCP. De hecho, los routers intermedios son completamente inconscientes de las conexiones TCP; los routers ven los datagramas, no las conexiones.

Una conexión TCP proporciona un **servicio full-duplex**: si existe una conexión TCP entre el proceso A que se ejecuta en un host y el proceso B que se ejecuta en otro host, entonces los datos de la capa de aplicación pueden fluir desde el proceso A al proceso B en el mismo instante que los datos de la capa de aplicación fluyen del proceso B al proceso A. Una conexión TCP casi siempre es una conexión **punto a punto**, es decir, entre un único emisor y un único receptor. La “multidifusión” (véase el material disponible en línea para este texto), la transferencia de datos desde un emisor a muchos receptores en una única operación de envío, no es posible con TCP. Con TCP, dos hosts son compañía y tres multitud.

Veamos ahora cómo se establece una conexión TCP. Suponga que un proceso que se está ejecutando en un host desea iniciar una conexión con otro proceso que se ejecuta en otro host.

Recuerde que el proceso que inicia la conexión es el *proceso cliente*, y el otro proceso es el *proceso servidor*. El proceso de la aplicación cliente informa en primer lugar a la capa de transporte del cliente que desea establecer una conexión con un proceso del servidor. Recuerde que en la Sección 2.7.2, hemos visto un programa cliente en Python que hacía esto ejecutando el comando:

```
clientSocket.connect((serverName, serverPort))
```

donde `serverName` es el nombre del servidor y `serverPort` identifica al proceso del servidor. El proceso TCP en el cliente procede entonces a establecer una conexión TCP con el TCP del servidor. Al finalizar esta sección veremos en detalle el procedimiento de establecimiento de la conexión. Por el momento, nos basta con saber que el cliente primero envía un segmento TCP especial; el servidor responde con un segundo segmento TCP especial y, por último, el cliente responde de nuevo con un tercer segmento especial. Los dos primeros segmentos no transportan ninguna carga útil; es decir, no transportan datos de la capa de aplicación; el tercero de estos segmentos es el que puede llevar la carga útil. Puesto que los tres segmentos son intercambiados entre dos hosts, este procedimiento de establecimiento de la conexión suele denominarse **acuerdo en tres fases**.

Una vez que se ha establecido una conexión TCP, los dos procesos de aplicación pueden enviarse datos entre sí. Consideraremos la transmisión de datos desde el proceso cliente al proceso servidor. El proceso cliente pasa un flujo de datos a través del socket (la puerta del proceso), como se ha descrito en la Sección 2.7. Una vez que los datos atraviesan la puerta, se encuentran en manos del protocolo TCP que se ejecuta en el cliente. Como se muestra en la Figura 3.28, TCP dirige estos datos al **buffer de emisión** de la conexión, que es uno de los buffers que se definen durante el proceso inicial del acuerdo en tres fases. De vez en cuando, TCP tomará fragmentos de datos del buffer de emisión y los pasa a la capa de red. La especificación de TCP [RFC 793] es bastante vaga en lo que respecta a especificar cuándo TCP debe realmente enviar los datos almacenados en el buffer, enunciando que TCP “debe transmitir esos datos en segmentos según su propia conveniencia”. La cantidad máxima de datos que pueden cogerse y colocarse en un segmento está limitada por el **tamaño máximo de segmento (MSS, Maximum Segment Size)**. Normalmente,



HISTORIA

VINTON CERF, ROBERT KAHN Y TCP/IP

A principios de la década de 1970 comenzaron a proliferar las redes de conmutación de paquetes, siendo ARPAnet (la red precursora de Internet) solo una más de muchas redes. Cada una de estas redes utilizaba su propio protocolo. Dos investigadores, Vinton Cerf y Robert Kahn, se dieron cuenta de la importancia de interconectar estas redes e inventaron un protocolo interred denominado TCP/IP (*Transmission Control Protocol/Internet Protocol*). Aunque Cerf y Kahn comenzaron viendo el protocolo como una sola entidad, más tarde lo dividieron en dos partes, TCP e IP, que operaban por separado. Cerf y Kahn publicaron un estudio sobre TCP/IP en mayo de 1974 en *IEEE Transactions on Communications Technology* [Cerf 1974].

El protocolo TCP/IP, que es la base de la Internet actual, fue diseñado antes que los PC, las estaciones de trabajo, los smartphones y las tabletas, antes de la proliferación de las tecnologías de las redes Ethernet, por cable, DSL, WiFi y otras tecnologías de red, y antes que la Web, las redes sociales y los flujos multimedia. Cerf y Kahn vieron la necesidad que existía de un protocolo de red que, por un lado, proporcionara un amplio soporte para las aplicaciones que ya estaban definidas y que, por otro lado, permitiera interoperar a los hosts y los protocolos de la capa de enlace.

En 2004, Cerf y Kahn recibieron el premio Turing Award de ACM, que está considerado como el “Premio Nobel de la Informática” por su trabajo pionero sobre los procesos de comunicación entre redes, incluyendo el diseño y la implementación de los protocolos básicos de comunicación de Internet (TCP/IP) y por su liderazgo en el mundo de las redes.

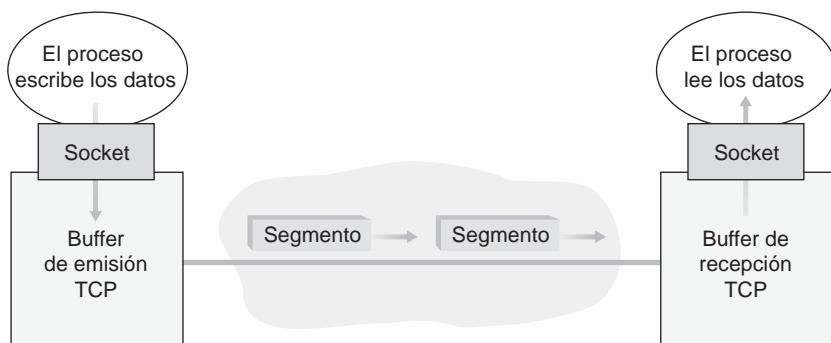


Figura 3.28 ♦ Buffers de emisión y recepción de TCP.

el MSS queda determinado en primer lugar por la longitud de la trama más larga de la capa de enlace que el host emisor local puede enviar [que es la **unidad máxima de transmisión**, (MTU, *Maximum Transmission Unit*)], y luego el MSS se establece de manera que se garantice que un segmento TCP (cuando se encapsula en un datagrama IP) más la longitud de la cabecera TCP/IP (normalmente 40 bytes) se ajuste a una única trama de la capa de enlace. Los protocolos de la capa de enlace Ethernet y PPP tienen una MTU de 1.500 bytes. Un valor común de MTU es 1.460 bytes. También se han propuesto métodos para descubrir la MTU de la ruta (la trama más larga de la capa de enlace que puede enviarse a través de todos los enlaces desde el origen hasta el destino) [RFC 1191] y establecer el MSS basándose en el valor de la MTU de la ruta. Observe que el MSS es la cantidad máxima de datos de la capa de aplicación en el segmento, no el tamaño máximo del segmento TCP incluyendo las cabeceras. Esta terminología resulta confusa, pero tenemos que convivir con ella, porque está muy extendida.

TCP empareja cada fragmento de datos del cliente con una cabecera TCP, formando **segmentos TCP**. Los segmentos se pasan a la capa de red, donde son encapsulados por separado dentro de datagramas IP de la capa de red. Los datagramas IP se envían entonces a la red. Cuando TCP recibe un segmento en el otro extremo, los datos del mismo se colocan en el buffer de recepción de la conexión TCP, como se muestra en la Figura 3.28. La aplicación lee el flujo de datos de este buffer. Cada lado de la conexión tiene su propio buffer de emisión y su propio buffer de recepción (puede ver el applet de control de flujo en línea en <http://www.awl.com/kurose-ross>, que proporciona una animación de los buffers de emisión y de recepción).

Por tanto, una conexión TCP consta de buffers, variables y un socket de conexión a un proceso en un host, y otro conjunto de buffers, variables y un socket de conexión a un proceso en otro host. Como hemos mencionado anteriormente, no se asignan ni buffers ni variables a la conexión dentro de los elementos de red (routers, switches y repetidores) existentes entre los hosts.

3.5.2 Estructura del segmento TCP

Después de haber visto de forma breve la conexión TCP, examinemos la estructura de un segmento TCP. El segmento TCP consta de campos de cabecera y un campo de datos. El campo de datos contiene un fragmento de los datos de la aplicación. Como hemos mencionado anteriormente, el MSS limita el tamaño máximo del campo de datos de un segmento. Cuando TCP envía un archivo grande, como por ejemplo una imagen como parte de una página web, normalmente divide el archivo en fragmentos de tamaño MSS (excepto el último fragmento, que normalmente será más pequeño que MSS). Sin embargo, las aplicaciones interactivas suelen transmitir fragmentos de datos que son más pequeños que el MSS; por ejemplo, en las aplicaciones de inicio de sesión remoto (*remote login*) como Telnet, el campo de datos del segmento TCP suele tener únicamente un byte. Puesto que habitualmente la cabecera de TCP tiene 20 bytes (12 bytes más que la cabecera de UDP), los segmentos enviados mediante Telnet solo pueden tener una longitud de 21 bytes.

La Figura 3.29 muestra la estructura del segmento TCP. Al igual que con UDP, la cabecera incluye los **números de puerto de origen y de destino**, que se utilizan para multiplexar y demultiplexar los datos de y para las aplicaciones de la capa superior. También, al igual que UDP, la cabecera incluye un **campo de suma de comprobación**. La cabecera de un segmento TCP también contiene los siguientes campos:

- El **campo Número de secuencia** de 32 bits y el **campo Número de reconocimiento** también de 32 bits son utilizados por el emisor y el receptor de TCP para implementar un servicio de transferencia de datos fiable, como se explica más adelante.
- El campo **Ventana de recepción** de 16 bits se utiliza para el control de flujo. Veremos en breve que se emplea para indicar el número de bytes que un receptor está dispuesto a aceptar.
- El campo **Longitud de cabecera** de 4 bits especifica la longitud de la cabecera TCP en palabras de 32 bits. La cabecera TCP puede tener una longitud variable a causa del campo opciones de TCP (normalmente, este campo está vacío, por lo que la longitud de una cabecera TCP típica es de 20 bytes).
- El campo **Opciones** es opcional y de longitud variable. Se utiliza cuando un emisor y un receptor negocian el tamaño máximo de segmento (MSS) o como un factor de escala de la ventana en las redes de alta velocidad. También se define una opción de marca temporal. Consulte los documentos RFC 854 y RFC 1323 para conocer detalles adicionales.
- El campo **Indicador** tiene 6 bits. El bit **ACK** se utiliza para indicar que el valor transportado en el campo de reconocimiento es válido; es decir, el segmento contiene un reconocimiento para un segmento que ha sido recibido correctamente. Los bits **RST**, **SYN** y **FIN** se utilizan para el establecimiento y cierre de conexiones, como veremos al final de esta sección. Los bits **CWR** y **ECE** se emplean en la notificación de congestión explícita, como veremos en la Sección 3.7.2. La activación del bit **PSH** indica que el receptor deberá pasar los datos a la capa superior de forma inmediata. Por último, el bit **URG** se utiliza para indicar que hay datos en este segmento que la entidad de la capa superior del lado emisor ha marcado como “urgentes”. La posición de este último byte de estos datos urgentes se indica mediante el **campo puntero de datos urgentes** de 16 bits. TCP tiene que informar a la entidad de la capa superior del lado receptor si existen datos

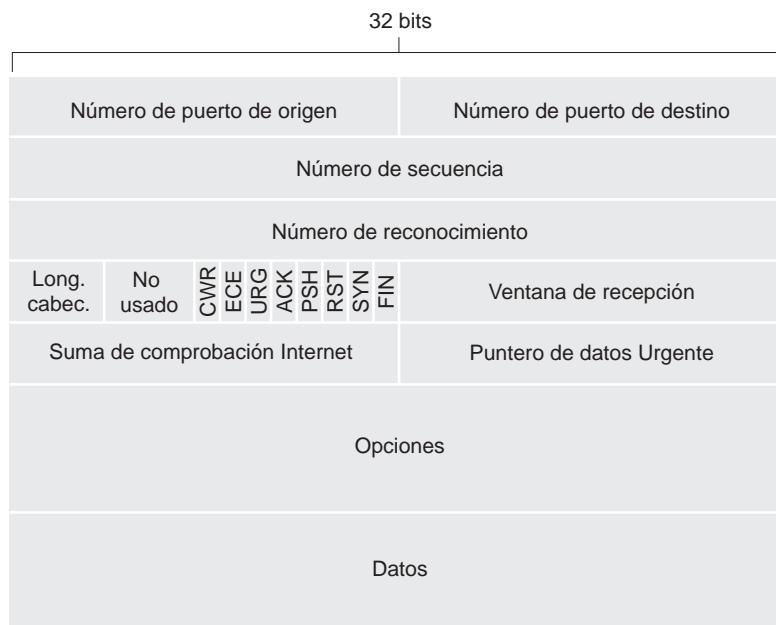


Figura 3.29 ♦ Estructura del segmento TCP.

urgentes y pasarle un puntero a la posición donde finalizan los datos urgentes. En la práctica, PSH, URG y el puntero a datos urgentes no se utilizan. Sin embargo, hemos hablado de estos campos con el fin de proporcionar al lector la información completa.

Nuestra experiencia como profesores es que nuestros estudiantes, en ocasiones, encuentran las exposiciones sobre los formatos de paquetes bastante áridas y quizás algo aburridas. Puede leer una exposición más divertida y amena sobre los campos de cabecera TCP, especialmente si le gustan los Legos™ como a nosotros, en [Pomeranz 2010].

Números de secuencia y números de reconocimiento

Dos de los campos más importantes de la cabecera de un segmento TCP son el campo número de secuencia y el campo número de reconocimiento. Estos campos son una parte crítica del servicio de transferencia de datos fiable de TCP. Pero antes de ver cómo se utilizan estos campos para proporcionar una transferencia de datos fiable, explicaremos en primer lugar lo que pone exactamente TCP en esos campos.

TCP percibe los datos como un flujo de bytes no estructurado pero ordenado. El uso que hace TCP de los números de secuencia refleja este punto de vista, en el sentido de que los números de secuencia hacen referencia al flujo de bytes transmitido y *no* a la serie de segmentos transmitidos. El **número de secuencia de un segmento** es por tanto el número del primer byte del segmento dentro del flujo de bytes. Veamos un ejemplo. Suponga que un proceso del host A desea enviar un flujo de datos a un proceso del host B a través de una conexión TCP. El protocolo TCP en el host A numerará implícitamente cada byte del flujo de datos. Suponga también que el flujo de datos consta de un archivo de 500.000 bytes, que el tamaño MSS es de 1.000 bytes y que el primer byte del flujo de datos está numerado como 0. Como se muestra en la Figura 3.30, TCP construye 500 segmentos a partir del flujo de datos. El primer segmento tiene asignado el número de secuencia 0, el segundo segmento tiene asignado el número de secuencia 1.000, el tercero tendrá asignado el número de secuencia 2.000, etc. Cada número de secuencia se inserta en el campo número de secuencia de la cabecera del segmento TCP apropiado.

Consideremos ahora los números de reconocimiento, que son algo más complicados que los números de secuencia. Recuerde que TCP es una conexión full-duplex, de modo que el host A puede estar recibiendo datos del host B mientras envía datos al host B (como parte de la misma conexión TCP). Todos los segmentos que llegan procedentes del host B tienen un número de secuencia para los datos que fluyen de B a A. *El número de reconocimiento que el host A incluye en su segmento es el número de secuencia del siguiente byte que el host A está esperando del host B.* Veamos algunos ejemplos para comprender qué es lo que ocurre aquí. Suponga que el host A ha recibido todos los bytes numerados de 0 a 535 procedentes de B y suponga también que está enviando un segmento al host B. El host A está esperando al byte 536 y todos los bytes que le siguen del flujo de datos del host B. Por tanto, el host A incluye 536 en el campo número de reconocimiento del segmento que envía a B.

Otro ejemplo: suponga que el host A ha recibido un segmento del host B que contiene los bytes de 0 a 535 y otro segmento que contiene los bytes de 900 a 1.000. Por alguna razón, el host A no ha

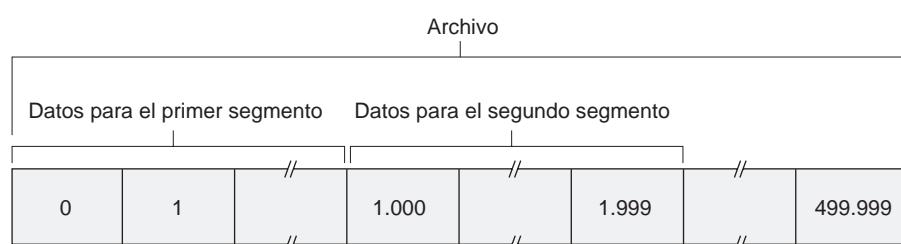


Figura 3.30 ♦ División de los datos del archivos en segmentos TCP.

recibido todavía los bytes de 536 a 899. En este ejemplo, el host A está esperando el byte 536 (y los que le siguen) para volver a crear el flujo de datos de B. Por tanto, el siguiente segmento de A a B contendrá el número 536 en el campo de número de reconocimiento. Dado que TCP solo confirma los bytes hasta el primer byte que falta en el flujo, se dice que TCP proporciona **reconocimientos acumulativos**.

Este último ejemplo plantea también un problema importante aunque sutil. El host A ha recibido el tercer segmento (bytes 900 a 1.000) antes de recibir el segundo segmento (bytes 536 a 899). Por tanto, el tercer segmento no ha llegado en orden. El sutil problema es el siguiente: ¿qué hace un host cuando no recibe los segmentos en orden a través de una conexión TCP? Curiosamente, los RFC dedicados a TCP no imponen ninguna regla y dejan la decisión a las personas que programan las implementaciones de TCP. Básicamente, tenemos dos opciones: (1) el receptor descarta de forma inmediata los segmentos que no han llegado en orden (lo que, como hemos anteriormente, puede simplificar el diseño del receptor) o (2) el receptor mantiene los bytes no ordenados y espera a que lleguen los bytes que faltan con el fin de llenar los huecos. Evidentemente, esta última opción es más eficiente en términos de ancho de banda de la red, y es el método que se utiliza en la práctica.

En la Figura 3.30 hemos supuesto que el número de secuencia inicial era cero. En la práctica, ambos lados de una conexión TCP eligen aleatoriamente un número de secuencia inicial. Esto se hace con el fin de minimizar la posibilidad de que un segmento que todavía está presente en la red a causa de una conexión anterior que ya ha terminado entre dos hosts pueda ser confundido con un segmento válido de una conexión posterior entre esos dos mismos hosts (que también estén usando los mismos números de puerto que la conexión antigua) [Sunshine 1978].

Telnet: caso de estudio de los números de secuencia y de reconocimiento

Telnet, definido en el documento RFC 854, es un popular protocolo de la capa de aplicación utilizado para los inicios de sesión remotos. Se ejecuta sobre TCP y está diseñado para trabajar entre cualquier pareja de hosts. A diferencia de las aplicaciones de transferencia masiva de datos vistas en el Capítulo 2, Telnet es una aplicación interactiva. Vamos a ver aquí un ejemplo, ya que ilustra muy bien los números de secuencia y de reconocimiento de TCP. Debemos mencionar que actualmente muchos usuarios prefieren utilizar el protocolo SSH en lugar de Telnet, porque los datos enviados a través de una conexión Telnet (¡incluidas las contraseñas!) no están cifrados, lo que hace que Telnet sea vulnerable a los ataques de personas que quieran escuchar la conexión (como veremos en la Sección 8.7).

Supongamos que el host A inicia una sesión Telnet con el host B. Puesto que el host A inicia la sesión, se etiqueta como el cliente y el host B como el servidor. Cada carácter escrito por el usuario (en el cliente) se enviará al host remoto; el host remoto devolverá una copia de cada carácter, que será mostrada en la pantalla del usuario Telnet. Este “eco” se emplea para garantizar que los caracteres vistos por el usuario Telnet ya han sido recibidos y procesados en el sitio remoto. Por tanto, cada carácter atraviesa la red dos veces entre el instante en el que usuario pulsa una tecla y el instante en el que el carácter se muestra en el monitor del usuario.

Suponga ahora que el usuario escribe una única letra, la ‘C’, y luego se va a por un café. Examinemos los segmentos TCP que están siendo enviados entre el cliente y el servidor. Como se muestra en la Figura 3.31, suponemos que los números de secuencia iniciales para el cliente y el servidor son, respectivamente, 42 y 79. Recuerde que el número de secuencia de un segmento es el número de secuencia del primer byte del campo de datos. Por tanto, el primer segmento enviado por el cliente tendrá el número de secuencia 42 y el primer segmento enviado desde el servidor tendrá el número de secuencia 79. Recuerde que el número de reconocimiento es el número de secuencia del siguiente byte de datos que el host está esperando. Cuando ya se ha establecido una conexión TCP pero todavía no se enviado ningún dato, el cliente está esperando la llegada del byte 79 y el servidor está esperando al byte 42.

Como se muestra en la Figura 3.31, se envían tres segmentos. El primer segmento se transmite desde el cliente al servidor, conteniendo la representación ASCII de 1 byte de la letra ‘C’ en su campo de datos. Este primer segmento también contiene el número 42 en su campo número de secuencia, como ya hemos descrito. Además, dado que el cliente todavía no ha recibido ningún dato procedente del servidor, este primer segmento contendrá el número 79 en su campo número de reconocimiento.

El segundo segmento se envía desde el servidor al cliente y además sirve a un doble propósito. En primer lugar, proporciona un reconocimiento de los datos que ha recibido el servidor. Al incluir el número 43 en el campo número de reconocimiento, el servidor está diciendo al cliente que ha recibido correctamente todo hasta el byte 42 y ahora está esperando los bytes 43 y posteriores. El segundo propósito de este segmento es devolver el eco de la letra ‘C’. Por tanto, el segundo segmento contiene la representación ASCII de la letra ‘C’ en su campo de datos. Este segundo segmento tiene el número de secuencia 79, el número de secuencia inicial del flujo de datos servidor-cliente de esta conexión TCP, ya que se trata del primer byte de datos que el servidor está enviando. Observe que la confirmación de los datos enviados desde el cliente al servidor se transporta en un segmento que contiene los datos enviados del servidor al cliente; se dice que este reconocimiento está **superpuesto** al segmento de datos enviado por el servidor al cliente.

El tercer segmento se envía desde el cliente al servidor. Su único propósito es confirmar los datos que ha recibido procedentes del servidor (recuerde que el segundo segmento contenía datos, la letra ‘C’, del servidor para el cliente). Este segmento tiene un campo de datos vacío (es decir, el paquete de reconocimiento no va superpuesto a los datos que van del cliente al servidor). El segmento contiene el número 80 en el campo número de reconocimiento porque el cliente ha recibido el flujo de bytes hasta el byte con el número de secuencia 79 y ahora está esperando los bytes 80 y subsiguientes. Es posible que le parezca extraño que este segmento también tenga un número de secuencia aunque no contenga datos pero, dado que los segmentos TCP disponen de un campo número de secuencia, es necesario incluir siempre en ese campo un valor.

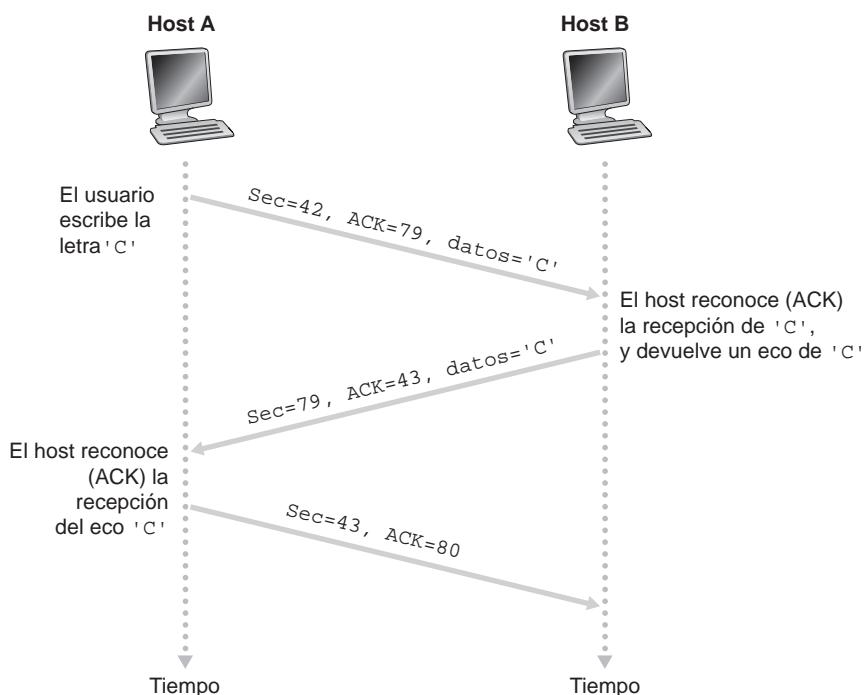


Figura 3.31 ♦ Números de secuencia y de reconocimiento en una aplicación Telnet simple sobre TCP.

3.5.3 Estimación del tiempo de ida y vuelta y fin de temporización

TCP, al igual que nuestro protocolo `rdt` de la Sección 3.4, utiliza un mecanismo de fin de temporización/retransmisión para recuperarse de la pérdida de segmentos. Aunque conceptualmente esto es muy simple, surgen muchos problemas sutiles al implementar dicho mecanismo en un protocolo real como, por ejemplo, TCP. Quizá la cuestión más obvia es la longitud de los intervalos de fin de temporización. Evidentemente, el intervalo de fin de temporización debería ser mayor que el tiempo de ida y vuelta (RTT) de la conexión; es decir, mayor que el tiempo que transcurre desde que se envía un segmento hasta que se recibe su reconocimiento. Si fuera de otra manera, se enviarían retransmisiones innecesarias. Pero, ¿cuánto mayor? y, ¿cómo se debería estimar el RTT por primera vez? ¿Debería asociarse un temporizador con cada uno de los segmentos no reconocidos? ¡Demasiadas preguntas! En esta sección vamos a basar nuestra exposición en el trabajo sobre TCP de [Jacobson 1988] y en las recomendaciones actuales del IETF para gestionar los temporizadores TCP [RFC 6298].

Estimación del tiempo de ida y vuelta

Comencemos nuestro estudio de la gestión del temporizador TCP viendo cómo estima TCP el tiempo de ida y vuelta entre el emisor y el receptor. Esto se lleva a cabo de la siguiente manera: el RTT de muestra, expresado como `RTTMuestra`, para un segmento es la cantidad de tiempo que transcurre desde que se envía el segmento (es decir, se pasa a IP) hasta que se recibe el correspondiente paquete de reconocimiento del segmento. En lugar de medir `RTTMuestra` para cada segmento transmitido, la mayor parte de las implementaciones TCP toman solo una medida de `RTTMuestra` cada vez. Es decir, en cualquier instante, `RTTMuestra` se estima a partir de uno solo de los segmentos transmitidos pero todavía no reconocidos, lo que nos proporciona un nuevo valor de `RTTMuestra` aproximadamente cada RTT segundos. Además, TCP nunca calcula `RTTMuestra` para un segmento que haya sido retransmitido; solo mide este valor para los segmentos que han sido transmitidos una vez [Karn 1987]. En uno de los problemas incluidos al final del capítulo se le pedirá que explique el por qué de este comportamiento.

Obviamente, los valores de `RTTMuestra` fluctuarán de un segmento a otro a causa de la congestión en los routers y a la variación de la carga en los sistemas terminales. A causa de esta fluctuación, cualquier valor de `RTTMuestra` dado puede ser atípico. Con el fin de estimar un RTT típico, es natural por tanto calcular algún tipo de promedio de los valores de `RTTMuestra`. TCP mantiene un valor promedio, denominado `RTTESTimado`, de los valores `RTTMuestra`. Para obtener un nuevo valor de `RTTMuestra`, TCP actualiza `RTTESTimado` según la fórmula siguiente:

$$\text{RTTESTimado} = (1 - \alpha) \cdot \text{RTTESTimado} + \alpha \cdot \text{RTTMuestra}$$

Hemos escrito esta fórmula como una instrucción de un lenguaje de programación (el nuevo valor de `RTTESTimado` es una combinación ponderada del valor anterior de `RTTESTimado` y del nuevo valor de `RTTMuestra`). El valor recomendado para α es $\alpha = 0,125$ (es decir, $1/8$) [RFC 6298], en cuyo caso la fórmula anterior se expresaría como sigue:

$$\text{RTTESTimado} = 0,875 \cdot \text{RTTESTimado} + 0,125 \cdot \text{RTTMuestra}$$

Observe que `RTTESTimado` es una media ponderada de los valores de `RTTMuestra`. Como se examina en uno de los problemas de repaso incluidos al final del capítulo, esta media ponderada asigna un mayor peso a las muestras recientes que a las más antiguas. Esto es lógico, ya que las muestras más recientes reflejan mejor la congestión que existe actualmente en la red. En estadística, una media como esta se denomina **media móvil exponencialmente ponderada (EWMA, Exponential Weighted Moving Average)**. El término “exponencial” aparece en EWMA porque el peso de un valor dado de `RTTMuestra` disminuye exponencialmente tan rápido como tienen lugar

EN LA PRÁCTICA

TCP proporciona un servicio de transferencia de datos fiable utilizando mensajes de reconocimiento positivos y temporizadores, de forma muy similar a como hemos visto en la Sección 3.4. TCP confirma los datos que ha recibido correctamente y retransmite segmentos cuando estos o sus correspondientes reconocimientos se piensa que se han perdido o se han corrompido. Ciertas versiones de TCP también disponen de un mecanismo NAK implícito con un mecanismo rápido de retransmisión TCP: la recepción de tres ACK duplicados para un determinado segmento sirve como un NAK implícito para el siguiente segmento, provocando la retransmisión de dicho segmento antes del fin de la temporización. TCP utiliza secuencias de números para permitir al receptor identificar los segmentos perdidos o duplicados. Al igual que en el caso de nuestro protocolo de transferencia de datos fiable, rdt3.0, TCP no puede por sí mismo saber si un cierto segmento, o su correspondiente ACK, se ha perdido, está corrompido o se ha retardado demasiado. En el emisor, la respuesta TCP será la misma: retransmitir el segmento en cuestión.

TCP también utiliza el procesamiento en cadena, permitiendo al emisor tener múltiples segmentos transmitidos pero aun no reconocidos en cualquier instante. Anteriormente hemos visto que el procesamiento en cadena puede mejorar enormemente la tasa de transferencia de una sesión cuando la relación entre el tamaño del segmento y el retardo de ida y vuelta es pequeña. El número específico de segmentos pendientes no reconocidos que un emisor puede tener se determina mediante los mecanismos de control de congestión y de control de flujo de TCP. El control de flujo de TCP se examina al final de esta sección y el mecanismo de control de congestión en la Sección 3.7. Por el momento, basta con que seamos conscientes de que el emisor TCP utiliza el procesamiento en cadena.

las actualizaciones. En los problemas de repaso se le pedirá que deduzca el término exponencial en RTTEstimado.

La Figura 3.32 muestra los valores RTTMuestra y RTTEstimado para $\alpha = 1/8$ en una conexión TCP entre `gaia.cs.umass.edu` (en Amherst, Massachusetts) y `fantasia.eurecom.fr` (en el sur de Francia). Evidentemente, las variaciones en RTTMuestra se suavizan en el cálculo de RTTEstimado.

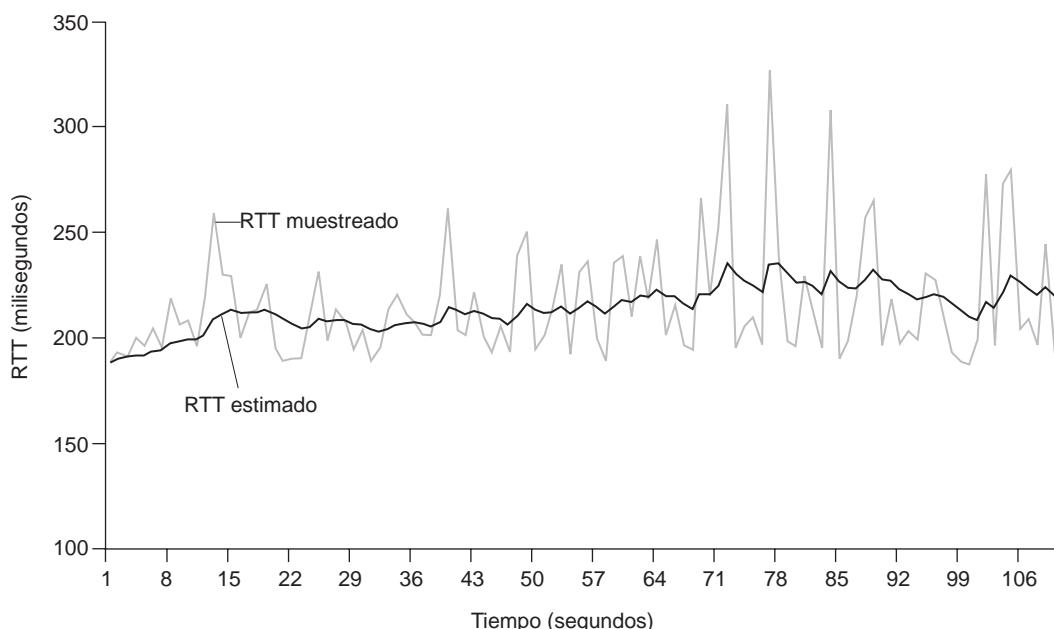


Figura 3.32 ♦ Muestreo de RTT y estimación de RTT.

Además de tener un estimado de RTT, también es importante disponer de una medida de la variabilidad de RTT. [RFC 6298] define la variación de RTT, RTTDesv, como una estimación de cuánto se desvía típicamente RTTMuestra de RTTEstimado:

$$\text{RTTDesv} = (1 - \beta) \cdot \text{RTTDesv} + \beta \cdot | \text{RTTMuestra} - \text{RTTEstimado} |$$

Observe que RTTDesv es una media EWMA de la diferencia entre RTTMuestra y RTTEstimado. Si los valores de RTTMuestra presentan una pequeña fluctuación, entonces RTTDesv será pequeño; por el contrario, si existe una gran fluctuación, RTTDesv será grande. El valor recomendado para β es 0,25.

Definición y gestión del intervalo de fin de temporización para la retransmisión

Dados los valores de RTTEstimado y RTTDesv, ¿qué valor debería utilizarse para el intervalo de fin de temporización de TCP? Evidentemente, el intervalo tendrá que ser mayor o igual que RTTEstimado, o se producirán retransmisiones innecesarias. Pero el intervalo de fin de temporización no debería ser mucho mayor que RTTEstimado; de otra manera, si un segmento se pierde, TCP no retransmitirá rápidamente el segmento, provocando retardos muy largos en la transferencia de datos. Por tanto, es deseable hacer el intervalo de fin de temporización igual a RTTEstimado más un cierto margen. El margen deberá ser más grande cuando la fluctuación en los valores de RTTMuestra sea grande y más pequeño cuando la fluctuación sea pequeña. El valor de RTTDesv deberá entonces tenerse en cuenta. Todas estas consideraciones se tienen en cuenta en el método TCP para determinar el intervalo de fin de temporización de las retransmisiones:

$$\text{IntervaloFinTemporizacion} = \text{RTTEstimado} + 4 \cdot \text{RTTDesv}$$

[RFC 6298] recomienda un valor inicial para IntervaloFinTemporización de 1 segundo. Además, cuando se produce un fin de temporización, el valor de IntervaloFinTemporización se duplica con el fin de evitar un fin de temporización prematuro para el segmento subsiguiente que enseguida será reconocido. Sin embargo, tan pronto como se recibe un segmento y se actualiza el valor de RTTEstimado, el IntervaloFinTemporización se calcula de nuevo utilizando la fórmula anterior.

3.5.4 Transferencia de datos fiable

Recuerde que el servicio de la capa de red de Internet (el servicio IP) no es fiable. IP no garantiza la entrega de los datagramas, no garantiza la entrega en orden de los datagramas y no garantiza la integridad de los datos contenidos en los datagramas. Con el servicio IP, los datagramas pueden desbordar los buffers de los routers y no llegar nunca a su destino, pueden llegar desordenados y los bits de un datagrama pueden corromperse (bacular de 0 a 1, y viceversa). Puesto que los segmentos de la capa de transporte son transportados a través de la red por los datagramas IP, estos segmentos de la capa de transporte pueden sufrir también estos problemas.

TCP crea un **servicio de transferencia de datos fiable** sobre el servicio de mejor esfuerzo pero no fiable de IP. El servicio de transferencia de datos fiable de TCP garantiza que el flujo de datos que un proceso extrae de su buffer de recepción TCP no está corrompido, no contiene huecos, ni duplicados y está en orden; es decir, el flujo de bytes es exactamente el mismo flujo que fue enviado por el sistema terminal existente en el otro extremo de la conexión. La forma en que TCP proporciona una transferencia de datos fiable implica muchos de los principios que hemos estudiado en la Sección 3.4.

En el anterior desarrollo que hemos realizado sobre las técnicas que proporcionan una transferencia de datos fiable, conceptualmente era fácil suponer que cada segmento transmitido pero aun no reconocido tenía asociado un temporizador individual. Aunque esto está bien en teoría, la gestión del temporizador puede requerir una sobrecarga considerable. Por tanto, los procedimientos

de gestión del temporizador TCP recomendados [RFC 6298] utilizan un *único* temporizador de retransmisión, incluso aunque haya varios segmentos transmitidos y aún no reconocidos. El protocolo TCP descrito en esta sección sigue esta recomendación de emplear un único temporizador.

Ahora vamos a ver cómo proporciona TCP el servicio de transferencia de datos fiable en dos pasos incrementales. En primer lugar, vamos a presentar una descripción extremadamente simplificada de un emisor TCP que solo emplea los fines de temporización para recuperarse de las pérdidas de segmentos; a continuación, veremos una descripción más completa que utiliza los mensajes de reconocimiento duplicados además de los fines de temporización. En la siguiente exposición suponemos que solo se están enviando datos en una única dirección, del host A al host B, y que el host A está enviando un archivo grande.

La Figura 3.33 presenta una descripción simplificada de un emisor TCP. Vemos que hay tres sucesos importantes relacionados con la transmisión y la retransmisión de datos en el emisor TCP: los datos recibidos desde la aplicación; el fin de temporización del temporizador y la recepción de paquetes ACK. Al producirse el primero de los sucesos más importantes, TCP recibe datos de la aplicación, encapsula los datos en un segmento y pasa el segmento a IP. Observe que cada segmento incluye un número de secuencia que es el número del primer byte de datos del segmento, dentro del flujo de datos, como se ha descrito en la Sección 3.5.2. Fíjese también en que si el temporizador no está ya funcionando para algún otro segmento, TCP lo inicia en el momento de pasar el segmento a IP (resulta útil pensar en el temporizador como si estuviera asociado con el segmento no reconocido más antiguo). El intervalo de caducidad para este temporizador es IntervaloFinTemporizacion, que se calcula a partir de RTTEstimado y RTTDesv, como se ha descrito en la Sección 3.5.3.

```
/* Suponga que el emisor no está restringido por los mecanismos de control de flujo ni de control de congestión de TCP, que el tamaño de los datos procedentes de la capa superior es menor que el MSS y que la transferencia de datos tiene lugar en un único sentido.*/
```

```
SigNumSec=NumeroSecuenciaInicial
BaseEmision=NumeroSecuenciaInicial

loop (siempre) {
    switch(suceso)

        suceso: datos recibidos de la aplicación de la capa superior
            crear segmento TCP con número de secuencia SigNumSec
            if (el temporizador no se está ejecutando actualmente)
                iniciar temporizador
            pasar segmento a IP
            SigNumSec=SigNumSec+longitud(datos)
            break;

        suceso: fin de temporización del temporizador
            retransmitir el segmento aún no reconocido con el
                número de secuencia más pequeño
            iniciar temporizador
            break;

        suceso: ACK recibido, con valor de campo ACK igual a y
            if (y > BaseEmision) {
                BaseEmision=y
                if (existen actualmente segmentos aún no reconocidos)
                    iniciar temporizador
            }
            break;

    } /* fin del bucle siempre */
```

Figura 3.33 ♦ Emisor TCP simplificado.

El segundo suceso importante es el fin de temporización. TCP responde a este suceso retransmitiendo el segmento que ha causado el fin de la temporización y, a continuación, reinicia el temporizador.

El tercer suceso importante que tiene que gestionar el emisor TCP es la llegada de un segmento de reconocimiento (ACK) procedente del receptor (más específicamente, un segmento que contenga un valor de campo ACK válido). Al ocurrir este suceso, TCP compara el valor ACK y con su variable `BaseEmision`. La variable de estado TCP `BaseEmision` es el número de secuencia del byte de reconocimiento más antiguo. (Por tanto, `BaseEmision-1` es el número de secuencia del último byte que se sabe que ha sido recibido correctamente y en orden en el receptor). Como hemos indicado anteriormente, TCP utiliza reconocimientos acumulativos, de modo que y confirma la recepción de todos los bytes anteriores al número de byte y. Si $y > \text{BaseEmision}$, entonces el ACK está confirmando uno o más de los segmentos no reconocidos anteriores. Así, el emisor actualiza su variable `BaseEmision` y reinicia el temporizador si actualmente aun existen segmentos no reconocidos.

Algunos escenarios interesantes

Acabamos de describir una versión enormemente simplificada de cómo TCP proporciona un servicio de transferencia de datos fiable. Pero incluso esta versión simplificada tiene sus sutilezas. Con el fin de clarificar cómo funciona este protocolo, vamos a analizar algunos escenarios sencillos. La Figura 3.34 describe el primero de ellos, en el que el host A envía un segmento al host B. Suponga que ese segmento tiene el número de secuencia 92 y contiene 8 bytes de datos. Después de enviar este segmento, el host A espera un segmento procedente de B con un número de reconocimiento de 100. Aunque el segmento de A se recibe en B, el paquete de reconocimiento de B a A se pierde. En este caso, se produce un suceso de fin de temporización y el host A retransmite el mismo segmento. Por supuesto, cuando el host B recibe la retransmisión, comprueba que de acuerdo con el número de secuencia el

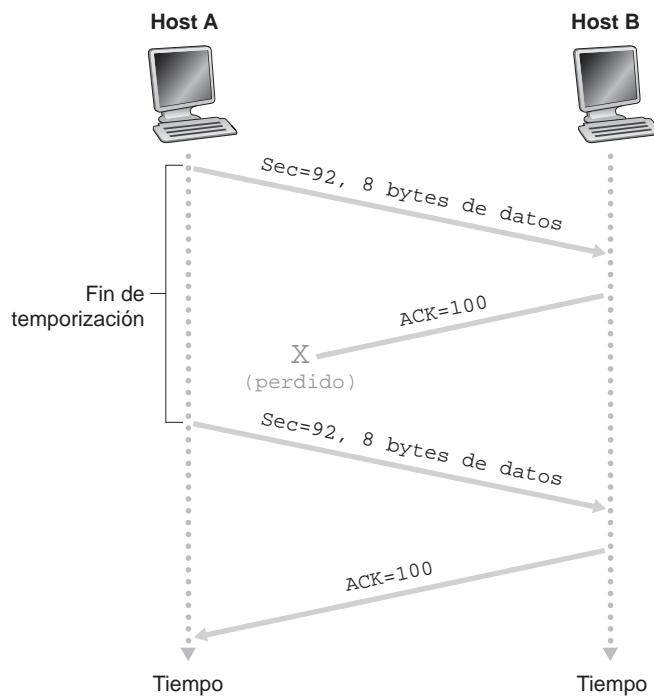


Figura 3.34 ♦ Retransmisión debida a la pérdida de un paquete de reconocimiento ACK.

segmento contiene datos que ya habían sido recibidos. Por tanto, TCP en el host B descartará los bytes del segmento retransmitido.

En el segundo escenario, mostrado en la Figura 3.35, el host A envía dos segmentos seguidos. El primer segmento tiene el número de secuencia 92 y 8 bytes de datos, y el segundo segmento tiene el número de secuencia 100 y 20 bytes de datos. Suponga que ambos segmentos llegan intactos a B, y que B envía dos mensajes de reconocimiento separados para cada uno de estos segmentos. El primero de estos mensajes tiene el número de reconocimiento 100 y el segundo el número 120. Suponga ahora que ninguno de estos mensajes llega al host A antes de que tenga lugar el fin de la temporización. Cuando se produce el fin de temporización, el host A reenvía el primer segmento con el número de secuencia 92 y reinicia el temporizador. Siempre y cuando el ACK correspondiente al segundo segmento llegue antes de que tenga lugar un nuevo fin de temporización, el segundo segmento no se retransmitirá.

En el tercer y último escenario, suponemos que el host A envía los dos segmentos del mismo modo que en el segundo ejemplo. El paquete de reconocimiento del primer segmento se pierde en la red, pero justo antes de que se produzca el fin de la temporización, el host A recibe un paquete de reconocimiento con el número de reconocimiento 120. Por tanto, el host A sabe que el host B ha recibido *todo* hasta el byte 119; por tanto, el host A no reenvía ninguno de los dos segmentos. Este escenario se ilustra en la Figura 3.36.

Duplicación del intervalo de fin de temporización

Examinemos ahora algunas de las modificaciones que aplican la mayor parte de las implementaciones TCP. La primera de ellas está relacionada con la duración del intervalo de fin de temporización después de que el temporizador ha caducado. En esta modificación, cuando tiene lugar un suceso de fin de temporización TCP retransmite el segmento aún no reconocido con el número de secuencia más pequeño, como se ha descrito anteriormente. Pero cada vez que TCP retransmite, define el

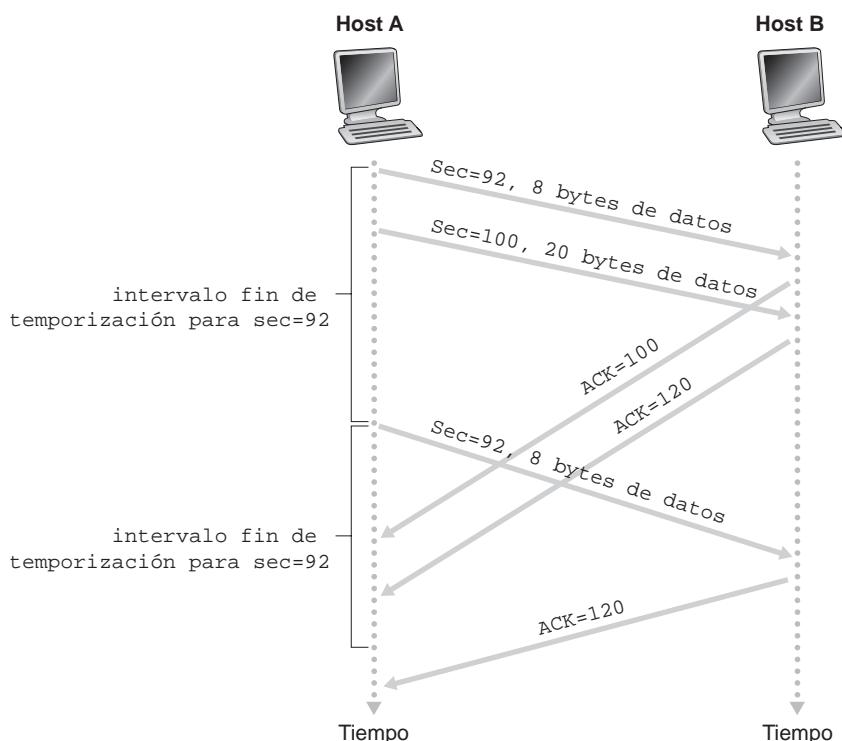


Figura 3.35 ♦ Segmento 100 no retransmitido.

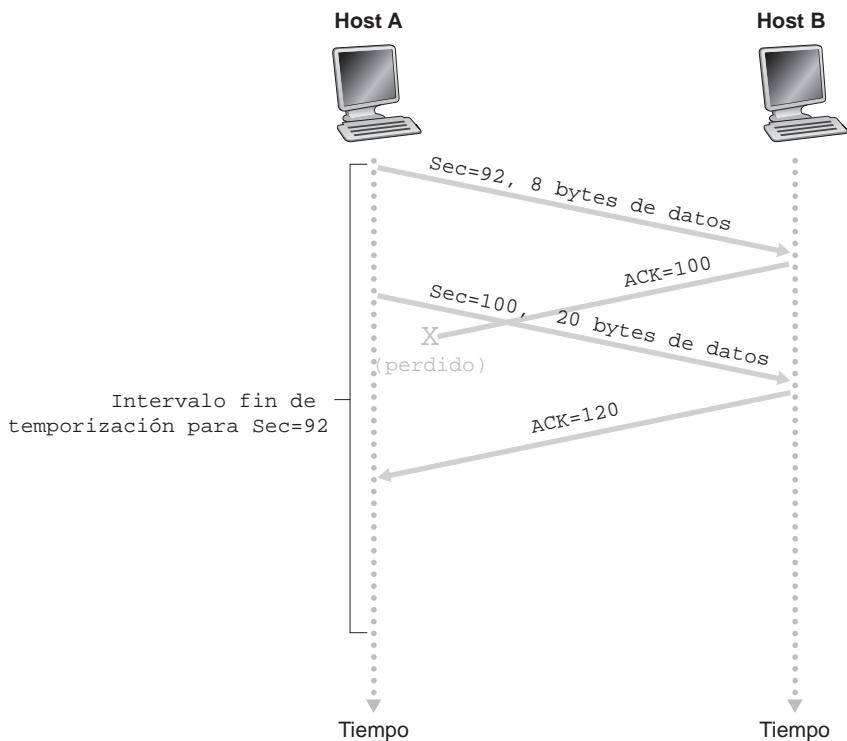


Figura 3.36 ♦ Un reconocimiento acumulativo evita la retransmisión del primer segmento.

siguiente intervalo de fin de temporización como dos veces el valor anterior, en lugar de obtenerlo a partir de los últimos valores de RTTEstimado y RTTDesv (como se ha descrito en la Sección 3.5.3). Por ejemplo, suponga que el IntervaloFinTemporización asociado con el segmento no reconocido más antiguo es 0,75 segundos cuando el temporizador caduca por primera vez. Entonces TCP retransmitirá este segmento y establecerá el nuevo intervalo en 1,5 segundos. Si el temporizador caduca de nuevo 1,5 segundos más tarde, TCP volverá a retransmitir este segmento, estableciendo el intervalo de caducidad en 3,0 segundos. Por tanto, los intervalos crecen exponencialmente después de cada retransmisión. Sin embargo, cuando el temporizador se inicia después de cualquiera de los otros dos sucesos (es decir, datos recibidos de la aplicación y recepción de un ACK), el IntervaloFinTemporización se obtiene a partir de los valores más recientes de RTTEstimado y RTTDesv.

Esta modificación proporciona una forma limitada de control de congestión (en la Sección 3.7 estudiaremos formas más exhaustivas de realizar el control de congestión en TCP). La caducidad del temporizador está causada muy probablemente por la congestión en la red, es decir, llegan demasiados paquetes a una (o más) colas de router a lo largo de la ruta entre el origen y el destino, haciendo que los paquetes sean descartados y/o sufran largos retardos de cola. Cuando existe congestión, si los orígenes continúan retransmitiendo paquetes persistentemente, la congestión puede empeorar. En lugar de ello, TCP actúa de forma más diplomática, haciendo que los emisores retransmitan después de intervalos cada vez más grandes. Cuando estudiamos CSMA/CD en el Capítulo 5, veremos que una idea similar se emplea en Ethernet .

Retransmisión rápida

Uno de los problemas con las retransmisiones generadas por los sucesos de fin de temporización es que el periodo de fin de temporización puede ser relativamente largo. Cuando se pierde un segmento, un periodo de fin de temporización largo fuerza al emisor a retardar el reenvío del paquete

perdido, aumentando el retardo terminal a terminal. Afortunadamente, el emisor puede a menudo detectar la pérdida de paquetes antes de que tenga lugar el suceso de fin de temporización, observando los ACK duplicados. Un **ACK duplicado** es un ACK que vuelve a reconocer un segmento para el que el emisor ya ha recibido un reconocimiento anterior. Para comprender la respuesta del emisor a un ACK duplicado, tenemos que entender en primer lugar por qué el receptor envía un ACK duplicado. La Tabla 3.2 resume la política de generación de mensajes ACK en el receptor TCP [RFC 5681]. Cuando un receptor TCP recibe un segmento con un número de secuencia que es mayor que el siguiente número de secuencia en orden esperado, detecta un hueco en el flujo de datos, es decir, detecta que falta un segmento. Este hueco podría ser el resultado de segmentos perdidos o reordenados dentro de la red. Dado que TCP no utiliza paquetes NAK, el receptor no puede devolver al emisor un mensaje de reconocimiento negativo explícito. En su lugar, simplemente vuelve a reconocer (es decir, genera un ACK duplicado) al último byte de datos en orden que ha recibido. Observe que la Tabla 3.2 contempla el caso de que el receptor no descarte los segmentos que no llegan en orden.

Puesto que un emisor suele enviar una gran número de segmentos seguidos, si se pierde uno de ellos, probablemente habrá muchos ACK duplicados seguidos. Si el emisor TCP recibe tres ACK duplicados para los mismos datos, toma esto como una indicación de que el segmento que sigue al segmento que ha sido reconocido tres veces se ha perdido (en los problemas de repaso, consideraremos la cuestión de por qué el emisor espera tres ACK duplicados, en lugar de un único duplicado). En el caso de que se reciban tres ACK duplicados, el emisor TCP realiza una **retransmisión rápida** [RFC 5681], reenviando el segmento que falta *antes* de que caduque el temporizador de dicho segmento. Esto se muestra en la Figura 3.37, en la que se pierde el segundo segmento y luego se retransmite antes de que caduque su temporizador. Para TCP con retransmisión rápida, el siguiente fragmento de código reemplaza al suceso de recepción de un ACK de la Figura 3.33:

```

sucedido: ACK recibido, con un valor de campo ACK de y
    if (y > BaseEmision) {
        BaseEmision=y
        if (existen segmentos pendientes de reconocimiento)
            iniciar temporizador
        }
        else /* un ACK duplicado para un segmento
               ya reconocido */
            incrementar el nº de mensajes ACK duplicados
            recibidos para y
            if (número de mensajes ACK duplicados recibidos
                para y==3)
                /* TCP con retransmisión rápida */
                reenviar el segmento con nº de secuencia y
    }
    break;

```

Anteriormente hemos mencionado que surgen muchos problemas sutiles cuando se implementa un mecanismo de fin de temporización/retransmisión en un protocolo real tal como TCP. Los procedimientos anteriores, que se han desarrollado como resultado de más de 20 años de experiencia con los temporizadores TCP, deberían convencerle de hasta qué punto son sutiles esos problemas.

Retroceder N o Repetición selectiva

Profundicemos algo más en nuestro estudio del mecanismo de recuperación de errores de TCP considerando la siguiente cuestión: ¿Es TCP un protocolo GBN o un protocolo SR? Recuerde que los reconocimientos de TCP son acumulativos y que los segmentos correctamente recibidos pero no en orden no son reconocidos individualmente por el receptor. En consecuencia, como se muestra en la Figura 3.33 (véase también la Figura 3.19), el emisor TCP solo necesita mantener

Suceso	Acción del receptor TCP
Llegada de un segmento en orden con el número de secuencia esperado. Todos los datos hasta el número de secuencia esperado ya han sido reconocidos.	ACK retardado. Esperar hasta durante 500 milisegundos la llegada de otro segmento en orden. Si el siguiente segmento en orden no llega en este intervalo, enviar un ACK.
Llegada de un segmento en orden con el número de secuencia esperado. Hay otro segmento en orden esperando la transmisión de un ACK.	Enviar inmediatamente un único ACK acumulativo, reconociendo ambos segmentos ordenados.
Llegada de un segmento desordenado con un número de secuencia más alto que el esperado. Se detecta un hueco.	Enviar inmediatamente un ACK duplicado, indicando el número de secuencia del siguiente byte esperado (que es el límite inferior del hueco).
Llegada de un segmento que completa parcial o completamente el hueco existente en los datos recibidos.	Enviar inmediatamente un ACK, suponiendo que el segmento comienza en el límite inferior del hueco.

Tabla 3.2 ♦ Recomendación para la generación de mensajes ACK en TCP [RFC 5681].

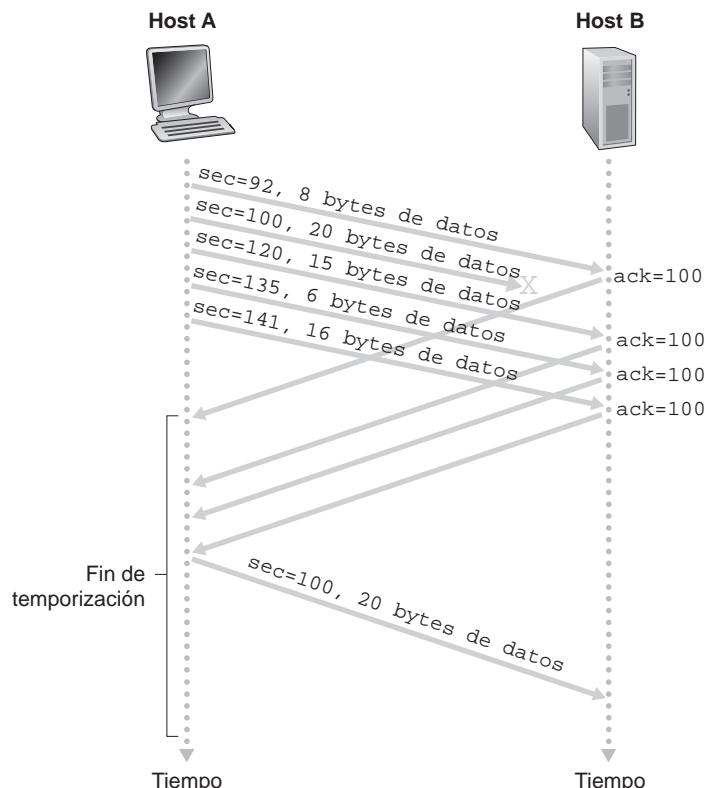


Figura 3.37 ♦ Retransmisión rápida: retransmisión del segmento que falta antes de que caduque el temporizador del segmento

el número de secuencia más pequeño de un byte transmitido pero no reconocido (`BaseEmision`) y el número de secuencia del siguiente byte que va a enviar (`SigNumSec`). En este sentido, TCP se parece mucho a un protocolo de tipo GBN. No obstante, existen algunas diferencias entre TCP y Retroceder N. Muchas implementaciones de TCP almacenan en buffer los segmentos recibidos correctamente pero no en orden [Stevens 1994]. Considere también lo que ocurre cuando el emisor envía una secuencia de segmentos $1, 2, \dots, N$, y todos ellos llegan en orden y sin errores al receptor. Suponga además que el paquete de reconocimiento para el paquete $n < N$ se pierde, pero los $N - 1$ paquetes de reconocimiento restantes llegan al emisor antes de que tengan lugar sus

respectivos fines de temporización. En este ejemplo, GBN retransmitiría no solo el paquete n , sino también todos los paquetes subsiguientes $n + 1, n + 2, \dots, N$. Por otro lado, TCP retransmitiría como mucho un segmento, el segmento n . Además, TCP no retransmitiría ni siquiera el segmento n si el reconocimiento para el segmento $n + 1$ llega antes del fin de temporización correspondiente al segmento n .

Una modificación propuesta para TCP es lo que se denomina **reconocimiento selectivo** [RFC 2018], que permite a un receptor TCP reconocer segmentos no ordenados de forma selectiva, en lugar de solo hacer reconocimientos acumulativos del último segmento recibido correctamente y en orden. Cuando se combina con la retransmisión selectiva (saltándose la retransmisión de segmentos que ya han sido reconocidos de forma selectiva por el receptor), TCP se comporta como nuestro protocolo SR selectivo. Por tanto, el mecanismo de recuperación de errores de TCP probablemente es mejor considerarlo como un híbrido de los protocolos GBN y SR.

3.5.5 Control de flujo

Recuerde que los hosts situados a cada lado de una conexión TCP disponen de un buffer de recepción para la conexión. Cuando la conexión TCP recibe bytes que son correctos y en secuencia, coloca los datos en el buffer de recepción. El proceso de aplicación asociado leerá los datos de este buffer, pero no necesariamente en el instante en que llegan. De hecho, la aplicación receptora puede estar ocupada con alguna otra tarea y puede incluso no leer los datos hasta mucho tiempo después de que estos hayan llegado. Si la aplicación es relativamente lenta en lo que respecta a la lectura de los datos, el emisor puede fácilmente desbordar el buffer de recepción de la conexión enviando muchos datos demasiado rápidamente.

TCP proporciona un **servicio de control de flujo** a sus aplicaciones para eliminar la posibilidad de que el emisor desborde el buffer del receptor. El control de flujo es por tanto un servicio de adaptación de velocidades (adapta la velocidad a la que el emisor está transmitiendo frente a la velocidad a la que la aplicación receptora está leyendo). Como hemos mencionado anteriormente, un emisor TCP también puede atascarse debido a la congestión de la red IP; esta forma de control del emisor se define como un mecanismo de **control de congestión**, un tema que exploraremos en detalle en las Secciones 3.6 y 3.7. Aunque las acciones tomadas por los controles de flujo y de congestión son similares (regular el flujo del emisor), obviamente se toman por razones diferentes. Lamentablemente, muchos autores utilizan los términos de forma indistinta, por lo que los lectores conocedores del tema deberían tratar de diferenciarlos. Examinemos ahora cómo proporciona TCP su servicio de control de flujo. En esta sección vamos a suponer que la implementación de TCP es tal que el receptor TCP descarta los segmentos que no llegan en orden.

TCP proporciona un servicio de control de flujo teniendo que mantiene el *emisor* una variable conocida como **ventana de recepción**. Informalmente, la ventana de recepción se emplea para proporcionar al emisor una idea de cuánto espacio libre hay disponible en el buffer del receptor. Puesto que TCP es una conexión full-duplex, el emisor de cada lado de la conexión mantiene una ventana de recepción diferente. Estudiemos la ventana de recepción en el contexto de una operación de transferencia de un archivo. Suponga que el host A está enviando un archivo grande al host B a través de una conexión TCP. El host B asigna un buffer de recepción a esta conexión; designamos al tamaño de este buffer como *BufferRecepcion*. De vez en cuando, el proceso de aplicación del host B lee el contenido del buffer. Definimos las siguientes variables:

- *UltimoByteLeido*: el número del último byte del flujo de datos del buffer leído por el proceso de aplicación del host B.
- *UltimoByteRecibido*: el número del último byte del flujo de datos que ha llegado procedente de la red y que se ha almacenado en el buffer de recepción del host B.

Puesto que en TCP no está permitido desbordar el buffer asignado, tenemos que:

$$\text{UltimoByteRecibido} - \text{UltimoByteLeido} \leq \text{BufferRecepcion}$$

La ventana de recepción, llamada `VentRecepcion`, se hace igual a la cantidad de espacio libre disponible en el buffer:

```
VentRecepcion = BufferRecepcion - [UltimoByteRecibido - UltimoByteLeido]
```

Dado que el espacio libre varía con el tiempo, `VentRecepcion` es una variable dinámica, la cual se ilustra en la Figura 3.38.

¿Cómo utiliza la conexión la variable `VentRecepcion` para proporcionar el servicio de control de flujo? El host B dice al host A la cantidad de espacio disponible que hay en el buffer de la conexión almacenando el valor actual de `VentRecepcion` en el campo ventana de recepción de cada segmento que envía a A. Inicialmente, el host B establece que `VentRecepcion = BufferRecepcion`. Observe que para poder implementar este mecanismo, el host B tiene que controlar diversas variables específicas de la conexión.

A su vez, el host A controla dos variables, `UltimoByteEnviado` y `UltimoByteReconocido`, cuyos significados son obvios. Observe que la diferencia entre estas dos variables, `UltimoByteEnviado - UltimoByteReconocido`, es la cantidad de datos no reconocidos que el host A ha enviado a través de la conexión. Haciendo que el número de datos no reconocidos sea inferior al valor de `VentRecepcion`, el host A podrá asegurarse de no estar desbordando el buffer de recepción en el host B. Por tanto, el host A se asegura, a todo lo largo del tiempo de vida de la conexión, de que:

$$\text{UltimoByteEnviado} - \text{UltimoByteReconocido} \leq \text{VentRecepcion}$$

Existe un pequeño problema técnico con este esquema. Para ver de qué se trata, suponga que el buffer de recepción del host B está lleno, de manera que `VentRecepcion = 0`. Después de anunciar al host A que `VentRecepcion = 0`, suponga también que B no tiene *nada* que enviar a A. Veamos qué es lo que ocurre. A medida que el proceso de aplicación del host B vacía el buffer, TCP no envía nuevos segmentos con valores nuevos `VentRecepcion` al host A; por supuesto, TCP envía un segmento al host A solo si tiene datos que enviar o si tiene que enviar un paquete de reconocimiento. Por tanto, el host A nunca es informado de que hay algo de espacio en el buffer de recepción del host B (el host A está bloqueado y no puede transmitir más datos). Para resolver este problema, la especificación TCP requiere al host A que continúe enviando segmentos con un byte de datos cuando la longitud de la ventana de recepción de B es cero. Estos segmentos serán reconocidos por el receptor. Finalmente, el buffer comenzará a vaciarse y los ACK contendrán un valor de `VentRecepcion` distinto de cero.

El sitio web del libro en <http://www.awl.com/kurose-ross> proporciona un applet Java interactivo que ilustra el funcionamiento de la ventana de recepción de TCP.

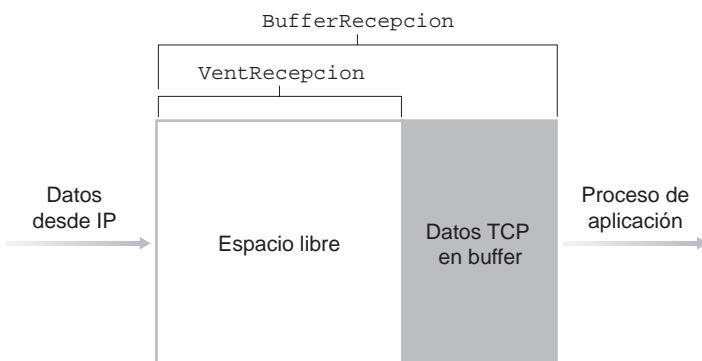


Figura 3.38 ♦ La ventana de recepción (`VentRecepcion`) y el buffer de recepción (`BufferRecepcion`).

Debemos comentar, una vez que hemos descrito el servicio de control de flujo de TCP, que UDP no proporciona ningún mecanismo de control de flujo y, en consecuencia, los segmentos pueden perderse en el receptor a causa del desbordamiento del buffer. Por ejemplo, considere la transmisión de una serie de segmentos UDP desde un proceso que se ejecuta en el host A a un proceso que se ejecuta en el host B. En una implementación típica de UDP, el protocolo UDP almacenará los segmentos en un buffer de tamaño finito que “precede” al correspondiente socket (es decir, la puerta de entrada al proceso). El proceso lee un segmento completo del buffer cada vez. Si el proceso no lee los segmentos del buffer lo suficientemente rápido, este se desbordará y los segmentos serán descartados.

3.5.6 Gestión de la conexión TCP

En esta subsección vamos a ver cómo se establece y termina una conexión TCP. Aunque este tema no parece particularmente emocionante, tiene una gran importancia, porque el establecimiento de una conexión TCP puede aumentar significativamente el retardo percibido (por ejemplo, cuando se navega por la Web). Además, muchos de los ataques de red más comunes, incluyendo el increíblemente popular ataque por inundación SYN, explotan las vulnerabilidades de la gestión de una conexión TCP. En primer lugar, veamos cómo se establece una conexión TCP. Suponga que hay un proceso en ejecución en un host (cliente) que desea iniciar una conexión con otro proceso que se ejecuta en otro host (servidor). El proceso de aplicación cliente informa en primer lugar al cliente TCP que desea establecer una conexión con un proceso del servidor. A continuación, el protocolo TCP en el cliente establece una conexión TCP con el protocolo TCP en el servidor de la siguiente manera:

- *Paso 1.* En primer lugar, TCP del lado del cliente envía un segmento TCP especial al TCP del lado servidor. Este segmento especial no contiene datos de la capa de aplicación. Pero uno de los bits indicadores de la cabecera del segmento (véase la Figura 3.29), el bit SYN, se pone a 1. Por esta razón, este segmento especial se referencia como un segmento SYN. Además, el cliente selecciona de forma aleatoria un número de secuencia inicial (`cliente_nsi`) y lo coloca en el campo número de secuencia del segmento TCP inicial SYN. Este segmento se encapsula dentro de un datagrama IP y se envía al servidor. Es importante que esta elección aleatoria del valor de `cliente_nsi` se haga apropiadamente con el fin de evitar ciertos ataques de seguridad [CERT 2001-09].
- *Paso 2.* Una vez que el datagrama IP que contiene el segmento SYN TCP llega al host servidor (¡suponiendo que llega!), el servidor extrae dicho segmento SYN del datagrama, asigna los buffers y variables TCP a la conexión y envía un segmento de conexión concedida al cliente TCP. (Veremos en el Capítulo 8 que la asignación de estos buffers y variables antes de completar el tercer paso del proceso de acuerdo en tres fases hace que TCP sea vulnerable a un ataque de denegación de servicio, conocido como ataque por inundación SYN.) Este segmento de conexión concedida tampoco contiene datos de la capa de aplicación. Sin embargo, contiene tres fragmentos de información importantes de la cabecera del segmento. El primero, el bit SYN se pone a 1. El segundo, el campo reconocimiento de la cabecera del segmento TCP se hace igual a `cliente_nsi+1`. Por último, el servidor elige su propio número de secuencia inicial (`servidor_nsi`) y almacena este valor en el campo número de secuencia de la cabecera del segmento TCP. Este segmento de conexión concedida está diciendo, en efecto, “He recibido tu paquete SYN para iniciar una conexión con tu número de secuencia inicial, `cliente_nsi`. Estoy de acuerdo con establecer esta conexión. Mi número de secuencia inicial es `servidor_nsi`”. El segmento de conexión concedida se conoce como **segmento SYNACK**.
- *Paso 3.* Al recibir el segmento SYNACK, el cliente también asigna buffers y variables a la conexión. El host cliente envía entonces al servidor otro segmento; este último segmento confirma el segmento de conexión concedida del servidor (el cliente hace esto almacenando el valor `servidor_nsi+1` en el campo de reconocimiento de la cabecera del segmento TCP). El

bit SYN se pone a cero, ya que la conexión está establecida. Esta tercera etapa del proceso de acuerdo en tres fases puede transportar datos del cliente al servidor dentro de la carga útil del segmento.

Una vez completados estos tres pasos, los hosts cliente y servidor pueden enviarse entre sí segmentos que contengan datos. En cada uno de estos segmentos futuros, el valor del bit SYN será cero. Observe que con el fin de establecer la conexión se envían tres paquetes entre los dos hosts, como se ilustra en la Figura 3.39. Por ello, este procedimiento de establecimiento de la conexión suele denominarse proceso de **acuerdo en tres fases**. En los problemas de repaso se exploran varios aspectos de este proceso de TCP (¿Por qué se necesitan los números de secuencia iniciales? ¿Por qué se necesita un proceso de acuerdo en tres fases en lugar de uno en dos fases?). Es interesante observar que un escalador y la persona que le asegura (que se encuentra por debajo del escalador y cuya tarea es sostener la cuerda de seguridad del mismo) utilizan un protocolo de comunicaciones con un proceso de acuerdo en tres fases que es idéntico al de TCP, para garantizar que ambas partes estén preparadas antes de que el escalador inicie el ascenso.

Todo lo bueno se termina y esto es aplicable también a una conexión TCP. Cualquiera de los dos procesos participantes en una conexión TCP pueden dar por terminada dicha conexión. Cuando una conexión se termina, los “recursos” (es decir, los buffers y las variables) de los hosts se liberan. Por ejemplo, suponga que el cliente decide cerrar la conexión, como se muestra en la Figura 3.40. El proceso de la aplicación cliente ejecuta un comando de cierre. Esto hace que el cliente TCP envíe un segmento especial TCP al proceso servidor. Este segmento especial contiene un bit indicador en la cabecera del segmento, el bit FIN (véase la Figura 3.29), puesto a 1. Cuando el servidor recibe este segmento, devuelve al cliente un segmento de reconocimiento. El servidor entonces envía su propio segmento de desconexión, que tiene el bit FIN puesto a 1. Por último, el cliente reconoce el segmento de desconexión del servidor. En esta situación, los recursos de ambos hosts quedan liberados.

Mientras se mantiene una conexión TCP, el protocolo TCP que se ejecuta en cada host realiza transiciones a través de los diversos **estados TCP**. La Figura 3.41 ilustra una secuencia típica de

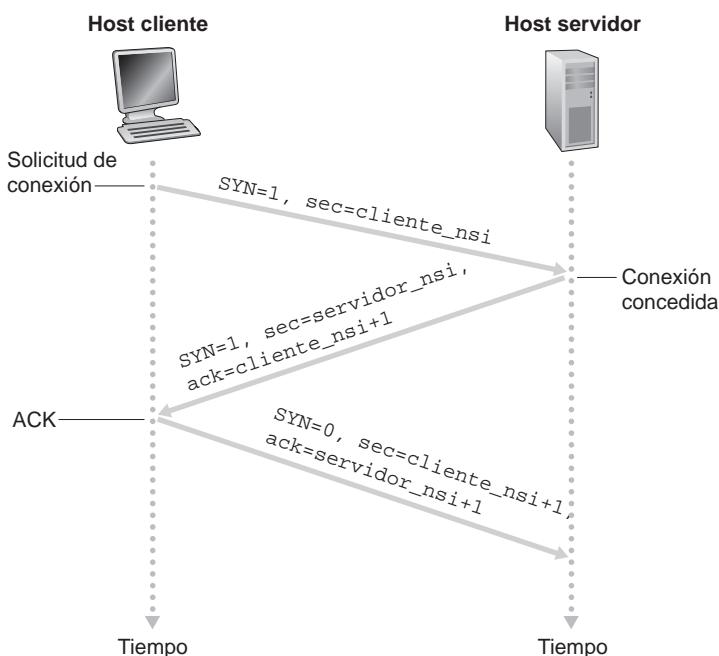


Figura 3.39 ♦ El proceso de acuerdo en tres fases de TCP: intercambio de segmentos.

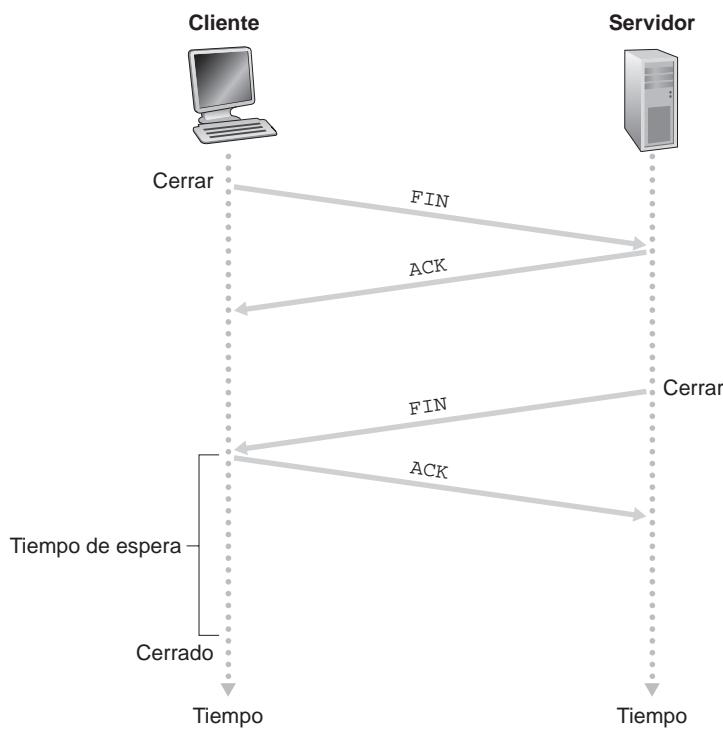


Figura 3.40 ♦ Cierre de una conexión TCP.

los estados TCP visitados por el *cliente* TCP, el cual se inicia en el estado CLOSED (cerrado). La aplicación en el lado del cliente inicia una nueva conexión (creando un objeto Socket en nuestros ejemplos de Java, así como en los ejemplos de Python del Capítulo 2). Esto hace que TCP en el cliente envíe un segmento SYN a TCP en el servidor. Después de haber enviado el segmento SYN, el cliente TCP entra en el estado SYN_SENT (SYN_enviado). Mientras se encuentra en este estado, el cliente TCP espera un segmento procedente del TCP servidor que incluya un reconocimiento del segmento anterior del cliente y que tenga el bit SYN puesto a 1. Una vez recibido ese segmento, el cliente TCP entra en el estado ESTABLISHED (establecido). Mientras está en este último estado, el cliente TCP puede enviar y recibir segmentos TCP que contengan datos de carga útil (es decir, datos generados por la aplicación).

Suponga que la aplicación cliente decide cerrar la conexión (tenga en cuenta que el servidor también podría decidir cerrar la conexión). Así, el cliente TCP envía un segmento TCP con el bit FIN puesto a 1 y entra en el estado FIN_WAIT_1 (FIN_espera_1). En este estado, el cliente TCP queda a la espera de un segmento TCP procedente del servidor que contenga un mensaje de reconocimiento. Cuando lo recibe, el cliente TCP pasa al estado FIN_WAIT_2. En este estado, el cliente espera a recibir otro segmento del servidor con el bit FIN puesto a 1; una vez recibido, el cliente TCP reconoce el segmento del servidor y pasa al estado TIME_WAIT, en el que puede reenviar al cliente TCP el reconocimiento final en caso de que el paquete ACK se pierda. El tiempo invertido en el estado TIME_WAIT es dependiente de la implementación, siendo sus valores típicos 30 segundos, 1 minuto y 2 minutos. Después de este tiempo de espera, la conexión se cierra y todos los recursos del lado del cliente (incluyendo los números de puerto) son liberados.

La Figura 3.42 ilustra la serie de estados que visita normalmente el TCP del lado del servidor, suponiendo que el cliente inicia el cierre de la conexión. Las transiciones se explican por sí mismas. En estos dos diagramas de transiciones de estados únicamente hemos mostrado cómo se establece y termina normalmente una conexión TCP. No hemos descrito lo que ocurre en determinados escenarios problemáticos; por ejemplo, cuando ambos lados de una conexión desean iniciar o

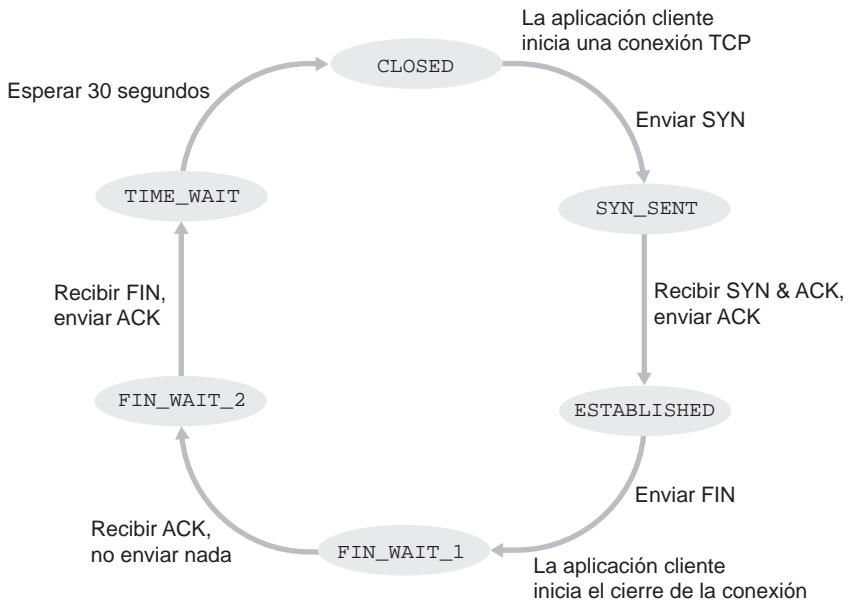


Figura 3.41 ♦ Secuencia típica de estados TCP visitados por un cliente TCP.

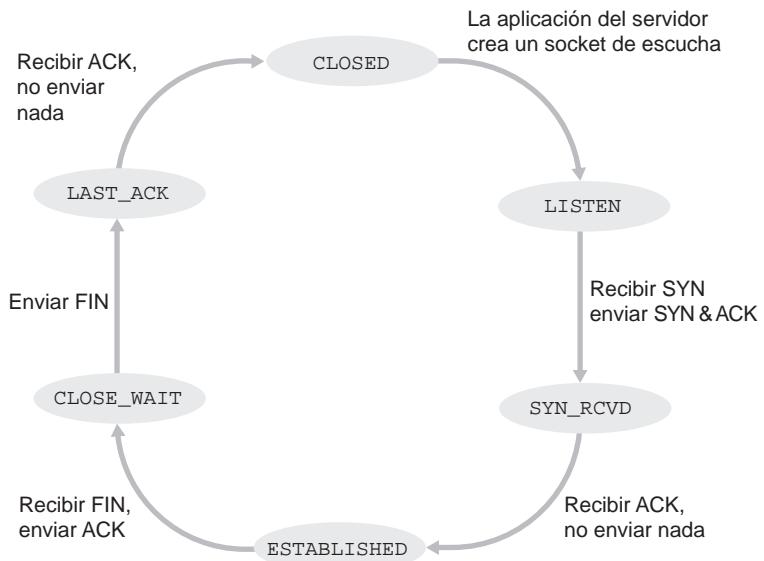


Figura 3.42 ♦ Secuencia típica de estados TCP visitados por un servidor TCP.

terminar al mismo tiempo la conexión. Si está interesado en este tema y en otros algo más avanzados sobre TCP, le animamos a consultar el exhaustivo libro de [Stevens 1994].

Hasta aquí, hemos supuesto que tanto el cliente como el servidor están preparados para comunicarse; es decir, que el servidor está escuchando en el puerto al que el cliente envía su segmento SYN. Consideremos lo que ocurre cuando un host recibe un segmento TCP cuyo número de puerto o cuya dirección IP de origen no se corresponde con ninguno de los sockets activos en el host. Por ejemplo, suponga que un host recibe un paquete TCP SYN cuyo puerto de destino es el número 80, pero el host no está aceptando conexiones en dicho puerto (es decir, no está ejecutando un servidor web en el puerto 80). Entonces, el host enviará al origen un segmento especial de

SEGURIDAD

EL ATAQUE POR INUNDACIÓN SYN

Hemos visto el proceso de acuerdo en tres fases de TCP en el que un servidor asigna e inicializa los buffers y variables de una conexión en respuesta a la recepción de un segmento SYN. A continuación, el servidor envía como respuesta un segmento SYNACK y espera al correspondiente segmento de reconocimiento (ACK) del cliente. Si el cliente no envía un segmento ACK para completar el tercero de los pasos del proceso de acuerdo en tres fases, al final (a menudo después de un minuto o más) el servidor terminará la conexión semiabierta y reclamará los recursos asignados.

Este protocolo de gestión de la conexión TCP establece la base para un ataque DoS clásico, conocido como **ataque por inundación SYN**. En este ataque, el atacante o atacantes envían un gran número de segmentos SYN TCP, sin completar el tercer paso del proceso de acuerdo. Con esta gran cantidad de segmentos SYN, los recursos de conexión del servidor pueden agotarse rápidamente a medida que se van asignando (aunque nunca se utilizan!) a conexiones semiabiertas. Con los recursos del servidor agotados, se niega el servicio a los clientes legítimos. Estos ataques por inundación SYN se encuentran entre los primeros ataques DoS documentados [CERT SYN 1996]. Afortunadamente, existe una defensa efectiva, denominada **cookies SYN** [RFC 4987], actualmente implantada en la mayoría de los principales sistemas operativos. Las cookies SYN funcionan del siguiente modo:

- Cuando el servidor recibe un segmento SYN, no sabe si ese segmento procede de un usuario legítimo o forma parte de un ataque por inundación SYN. Por tanto, el servidor en lugar de crear una conexión semiabierta TCP para ese segmento SYN, crea un número de secuencia TCP inicial que es una función compleja (una función hash) de las direcciones IP de origen y de destino y los números de puerto del segmento SYN, así como de un número secreto que únicamente conoce el servidor. Este número de secuencia inicial cuidadosamente confeccionado se denomina "cookie". El servidor tiene entonces que enviar al cliente un paquete SYNACK con este número de secuencia inicial especial. *Es importante que el servidor no recuerde la cookie ni ninguna otra información de estado correspondiente al paquete SYN.*
- Si el cliente es legítimo, entonces devolverá un segmento ACK. El servidor, al recibir este ACK, tiene que verificar que corresponde a algún SYN enviado anteriormente. ¿Cómo se hace esto si el servidor no mantiene memoria acerca de los segmentos SYN? Como ya habrá imaginado, se hace utilizando la cookie. Recuerde que para un segmento ACK legítimo, el valor contenido en el campo de reconocimiento es igual al número de secuencia inicial del segmento SYNACK (el valor de la cookie en este caso) más uno (véase la Figura 3.39). A continuación, el servidor puede ejecutar la misma función hash utilizando las direcciones IP de origen y de destino, los números de puerto en el segmento SYNACK (los cuales son los mismos que en el segmento SYN original) y el número secreto. Si el resultado de la función más uno es igual que el número de reconocimiento (cookie) del SYNACK del cliente, el servidor concluye que el ACK se corresponde con un segmento SYN anterior y, por tanto, lo valida. A continuación, el servidor crea una conexión completamente abierta junto con un socket.
- Por el contrario, si el cliente no devuelve un segmento ACK, entonces el segmento SYN original no causa ningún daño al servidor, ya que este no le ha asignado ningún recurso.

reinicio. Este segmento TCP tiene el bit indicador RST (véase la Sección 3.5.2) puesto a 1. Por tanto, cuando un host envía un segmento de reinicio, le está diciendo al emisor “No tengo un socket para ese segmento. Por favor, no reenvies el segmento.” Cuando un host recibe un paquete UDP en el que el número de puerto de destino no se corresponde con un socket UDP activo, el host envía un datagrama ICMP especial, como se explica en el Capítulo 5.

Ahora que ya tenemos unos buenos conocimientos sobre cómo se gestiona una conexión TCP, vamos a volver sobre la herramienta de exploración de puertos nmap y vamos a ver más

detalladamente cómo funciona. Para explorar un puerto TCP específico, por ejemplo, el puerto 6789, en un host objetivo, nmap enviará un segmento TCP SYN con el puerto de destino 6789 a dicho host. Pueden obtenerse entonces tres posibles resultados:

- *El host de origen recibe un segmento TCP SYNACK del host objetivo.* Dado que esto significa que hay una aplicación ejecutándose con TCP en el puerto 6789 del host objetivo, nmap devuelve “open” (puerto abierto).
- *El host de origen recibe un segmento TCP RST procedente del host objetivo.* Esto significa que el segmento SYN ha alcanzado el host objetivo, pero este no está ejecutando una aplicación con TCP en el puerto 6789. Pero el atacante sabrá como mínimo que el segmento destinado al puerto 6789 del host no está bloqueado por ningún cortafuegos existente en la ruta entre los hosts de origen y de destino. (Los cortafuegos se estudian en el Capítulo 8.)
- *El origen no recibe nada.* Probablemente, esto significa que el segmento SYN fue bloqueado por un cortafuegos intermedio y nunca llegó al host objetivo.

nmap es una potente herramienta que puede detectar “las brechas en el muro”, no solo en lo relativo a los puertos TCP abiertos, sino también a los puertos UDP abiertos, a los cortafuegos y sus configuraciones e incluso a las versiones de aplicaciones y sistemas operativos. La mayor parte de esta tarea se lleva a cabo manipulando los segmentos de gestión de la conexión TCP [Skoudis 2006]. Puede descargar nmap en la dirección www.nmap.org.

Con esto completamos nuestra introducción a los mecanismos de control de errores y de control de flujo de TCP. En la Sección 3.7, volveremos sobre TCP y estudiaremos más detalladamente el control de congestión de TCP. Sin embargo, antes de eso, vamos a dar un paso atrás y vamos a examinar los problemas de control de congestión en un contexto más amplio.

3.6 Principios del control de congestión

En la sección anterior hemos examinado tanto los principios generales como los específicos de los mecanismos de TCP utilizados para proporcionar un servicio de transferencia de datos fiable en lo que se refiere a la pérdida de paquetes. Anteriormente habíamos mencionado que, en la práctica, tales pérdidas son normalmente el resultado de un desbordamiento de los buffers de los routers a medida que la red se va congestionando. La retransmisión de paquetes por tanto se ocupa de un síntoma de la congestión de red (la pérdida de un segmento específico de la capa de transporte) pero no se ocupa de la causa de esa congestión de la red (demasiados emisores intentando transmitir datos a una velocidad demasiado alta). Para tratar la causa de la congestión de la red son necesarios mecanismos que regulen el flujo de los emisores en cuanto la congestión de red aparezca.

En esta sección consideraremos el problema del control de congestión en un contexto general, con el fin de comprender por qué la congestión es algo negativo, cómo la congestión de la red se manifiesta en el rendimiento ofrecido a las aplicaciones de la capa superior y cómo pueden aplicarse diversos métodos para evitar la congestión de la red o reaccionar ante la misma. Este estudio de carácter general del control de la congestión es apropiado porque, puesto que junto con la transferencia de datos fiable, se encuentra al principio de la lista de los diez problemas más importantes de las redes. En la siguiente sección se lleva a cabo un estudio detallado sobre el algoritmo de control de congestión de TCP.

3.6.1 Las causas y los costes de la congestión

Iniciemos este estudio de carácter general sobre el control de congestión examinando tres escenarios con una complejidad creciente en los que se produce congestión. En cada caso, veremos en primer lugar por qué se produce la congestión y el coste de la misma (en términos de recursos no utilizados

por completo y del bajo rendimiento ofrecido a los sistemas terminales). Todavía no vamos a entrar a ver cómo reaccionar ante una congestión, o cómo evitarla, simplemente vamos a poner el foco sobre la cuestión más simple de comprender: qué ocurre cuando los hosts incrementan su velocidad de transmisión y la red comienza a congestionarse.

Escenario 1: dos emisores, un router con buffers de capacidad ilimitada

Veamos el escenario de congestión más simple posible: dos hosts (A y B), cada uno de los cuales dispone de una conexión que comparte un único salto entre el origen y el destino, como se muestra en la Figura 3.43.

Supongamos que la aplicación del host A está enviando datos a la conexión (por ejemplo, está pasando datos al protocolo de la capa de transporte a través de un socket) a una velocidad media de λ_{in} bytes/segundo. Estos datos son originales en el sentido de que cada unidad de datos se envía solo una vez al socket. El protocolo del nivel de transporte subyacente es un protocolo simple. Los datos se encapsulan y se envían; no existe un mecanismo de recuperación de errores (como por ejemplo, las retransmisiones), ni se realiza un control de flujo ni un control de congestión. Ignorando la sobrecarga adicional debida a la adición de la información de cabecera de las capas de transporte e inferiores, la velocidad a la que el host A entrega tráfico al router en este primer escenario es por tanto λ_{in} bytes/segundo. El host B opera de forma similar, y suponemos, con el fin de simplificar, que también está enviando datos a una velocidad de λ_{in} bytes/segundo. Los paquetes que salen de los hosts A y B atraviesan un router y un enlace de salida compartido de capacidad R . El router tiene buffers que le permiten almacenar paquetes entrantes cuando la tasa de llegada de paquetes excede la capacidad del enlace de salida. En este primer escenario, suponemos que el router tiene un buffer con una cantidad de espacio infinita.

La Figura 3.44 muestra el rendimiento de la conexión del host A en este primer escenario. La gráfica de la izquierda muestra la **tasa de transferencia por conexión** (número de bytes por segundo en el receptor) como una función de la velocidad de transmisión de la conexión. Para una velocidad de transmisión comprendida entre 0 y $R/2$, la tasa de transferencia en el receptor es igual a la velocidad de transmisión en el emisor (todo lo que envía el emisor es recibido en el receptor con un retardo finito). Sin embargo, cuando la velocidad de transmisión es mayor que $R/2$, la tasa de transferencia es de solo $R/2$. Este límite superior de la tasa de transferencia es una consecuencia de compartir entre dos conexiones la capacidad del enlace. El enlace simplemente no puede proporcionar paquetes a un receptor a una velocidad de régimen permanente que sea mayor que $R/2$. Independientemente de lo altas que sean las velocidades de transmisión de los hosts A y B, nunca verán una tasa de transferencia mayor que $R/2$.

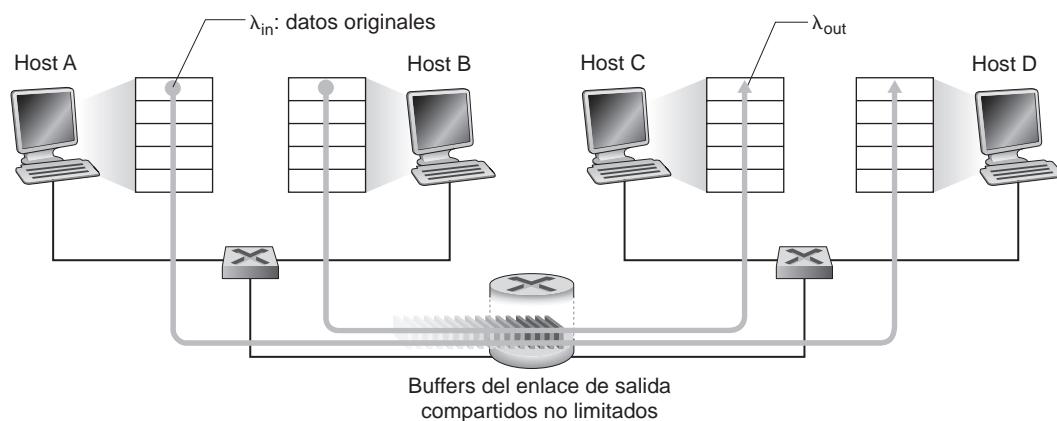


Figura 3.43 ♦ Escenario de congestión 1: dos conexiones que comparten un único salto con buffers de capacidad ilimitada.

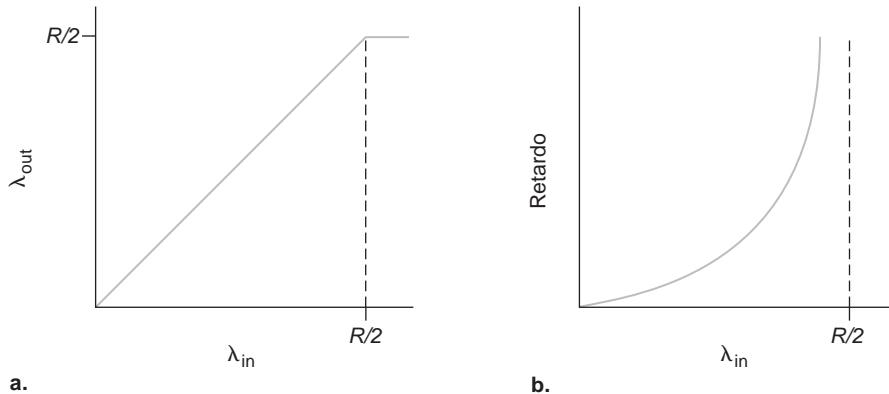


Figura 3.44 ♦ Escenario de congestión 1: tasa de transferencia y retardo en función de la velocidad de transmisión del host.

Alcanzar una tasa de transferencia por conexión de $R/2$ podría realmente parecer algo positivo, porque el enlace se utiliza completamente en suministrar paquetes a sus destinos. Sin embargo, la gráfica de la derecha de la Figura 3.44 muestra la consecuencia de operar cerca de la capacidad del enlace. A medida que la velocidad de transmisión se aproxima a $R/2$ (desde la izquierda), el retardo medio se hace cada vez más grande. Cuando la velocidad de transmisión excede de $R/2$, el número medio de paquetes en cola en el router no está limitado y el retardo medio entre el origen y el destino se hace infinito (suponiendo que las conexiones operan a esas velocidades de transmisión durante un periodo de tiempo infinito y que existe una cantidad infinita de espacio disponible en el buffer). Por tanto, aunque operar a una tasa de transferencia agregada próxima a R puede ser ideal desde el punto de vista de la tasa de transferencia, no es ideal en absoluto desde el punto de vista del retardo. *Incluso en este escenario (extremadamente) idealizado, ya hemos encontrado uno de los costes de una red congestionada: los grandes retardos de cola experimentados cuando la tasa de llegada de los paquetes se aproxima a la capacidad del enlace.*

Escenario 2: dos emisores y un router con buffers finitos

Ahora vamos a modificar ligeramente el escenario 1 de dos formas (véase la Figura 3.45). En primer lugar, suponemos que el espacio disponible en los buffers del router es finito. Una consecuencia de esta suposición aplicable en la práctica es que los paquetes serán descartados cuando lleguen a un buffer que ya esté lleno. En segundo lugar, suponemos que cada conexión es fiable. Si un paquete que contiene un segmento de la capa de transporte se descarta en el router, el emisor tendrá que retransmitirlo. Dado que los paquetes pueden retransmitirse, ahora tenemos que ser más cuidadosos al utilizar el término *velocidad de transmisión*. Específicamente, vamos a designar la velocidad a la que la aplicación envía los datos originales al socket como λ_{in} bytes/segundo. La velocidad a la que la capa de transporte envía segmentos (que contienen los datos originales y los datos retransmitidos) a la red la denotaremos como λ'_{in} bytes/segundo. En ocasiones, λ'_{in} se denomina **carga ofrecida** a la red.

El rendimiento de este segundo escenario dependerá en gran medida de cómo se realicen las retransmisiones. En primer lugar, considere el caso no realista en el que el host A es capaz de determinar de alguna manera (mágicamente) si el buffer en el router está libre o lleno y enviar entonces un paquete solo cuando el buffer esté libre. En este caso no se produciría ninguna pérdida, λ_{in} sería igual a λ'_{in} y la tasa de transferencia de la conexión sería igual a λ_{in} . Este caso se muestra en la Figura 3.46(a). Desde el punto de vista de la tasa de transferencia, el rendimiento es ideal: todo lo que se envía, se recibe. Observe que, en este escenario, la velocidad media de transmisión de host no puede ser mayor que $R/2$, ya que se supone que nunca tiene lugar una pérdida de paquetes.

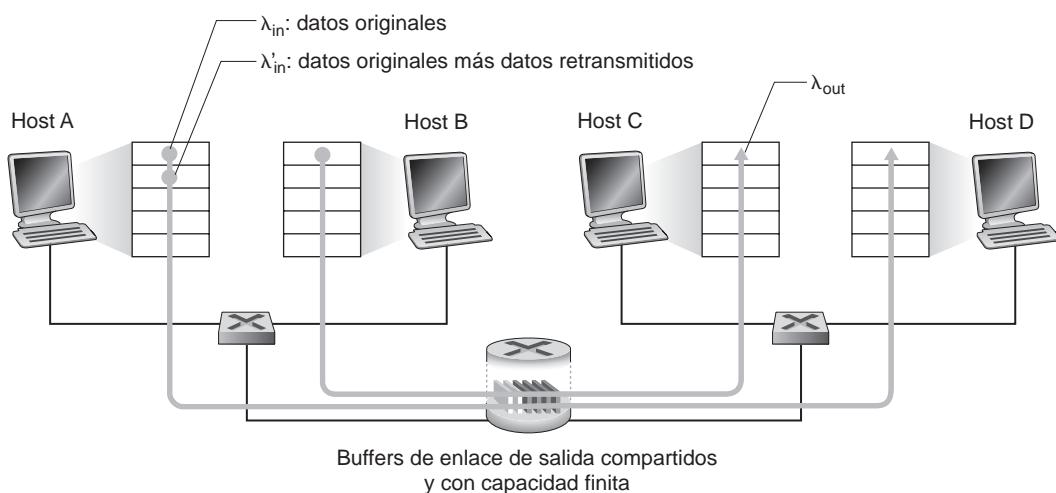


Figura 3.45 ♦ Escenario 2: dos hosts (con retransmisión) y un router con buffers con capacidad finita.

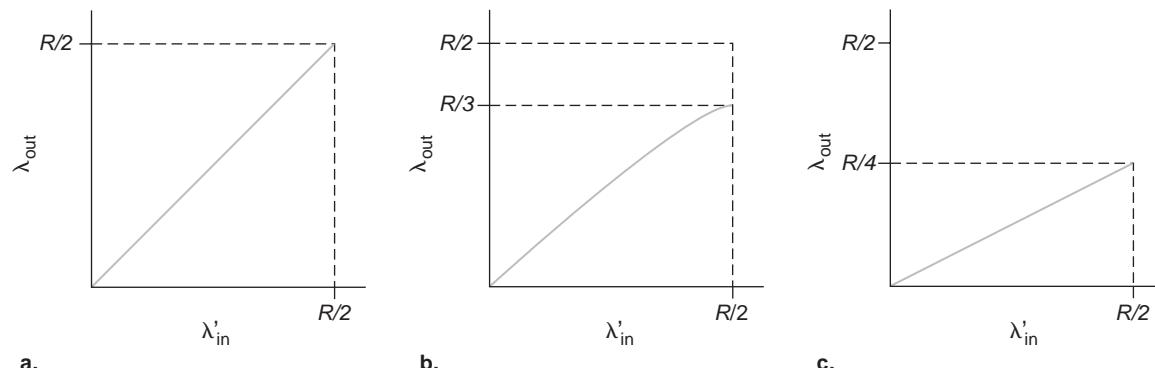


Figura 3.46 ♦ Escenario 2: rendimiento con buffers de capacidad finita.

Consideremos ahora un caso algo más realista en el que el emisor solo retransmite cuando sabe con seguridad que un paquete se ha perdido. De nuevo, esta es una suposición un poco exagerada; sin embargo, es posible que el host emisor tenga fijado su intervalo de fin de temporización en un valor lo suficientemente grande como para garantizar que un paquete que no ha sido reconocido es un paquete que se ha perdido. En este caso, el rendimiento puede ser similar al mostrado en la Figura 3.46(b). Para apreciar lo que está ocurriendo aquí, considere el caso en el que la carga ofrecida, λ'_{in} (la velocidad de transmisión de los datos originales más la de las retransmisiones), es igual a $R/2$. Según la Figura 3.46(b), para este valor de la carga ofrecida la velocidad a la que los datos son suministrados a la aplicación del receptor es $R/3$. Por tanto, de las $0,5R$ unidades de datos transmitidos, $0,333R$ bytes/segundo (como media) son datos originales y $0,166R$ bytes/segundo (como media) son datos retransmitidos. Tenemos aquí por tanto otro de los costes de una red congestionada: el emisor tiene que realizar retransmisiones para poder compensar los paquetes descartados (perdidos) a causa de un desbordamiento de buffer.

Por último, considere el caso en el que el emisor puede alcanzar el fin de la temporización de forma prematura y retransmitir un paquete que ha sido retardado en la cola pero que todavía no se ha perdido. En este caso, tanto el paquete de datos original como la retransmisión pueden llegar al receptor. Por supuesto, el receptor necesitará solo una copia de este paquete y descartará

la retransmisión. En este caso, el trabajo realizado por el router al reenviar la copia retransmitida del paquete original se desperdicia, ya que el receptor ya había recibido la copia original de ese paquete. El router podría haber hecho un mejor uso de la capacidad de transmisión del enlace enviando en su lugar un paquete distinto. *Por tanto, tenemos aquí otro de los costes de una red congestionada: las retransmisiones innecesarias del emisor causadas por retardos largos pueden llevar a que un router utilice el ancho de banda del enlace para reenviar copias innecesarias de un paquete.* La Figura 3.46(c) muestra la tasa de transferencia en función de la carga ofrecida cuando se supone que el router reenvía cada paquete dos veces (como media). Dado que cada paquete se reenvía dos veces, la tasa de transferencia tendrá un valor asintótico de $R/4$ cuando la carga ofrecida se aproxime a $R/2$.

Escenario 3: cuatro emisores, routers con buffers de capacidad finita y rutas con múltiples saltos

En este último escenario dedicado a la congestión de red tenemos cuatro hosts que transmiten paquetes a través de rutas solapadas con dos saltos, como se muestra en la Figura 3.47. De nuevo suponemos que cada host utiliza un mecanismo de fin de temporización/retransmisión para implementar un servicio de transferencia de datos fiable, que todos los hosts tienen el mismo valor de λ_{in} y que todos los enlaces de router tienen una capacidad de R bytes/segundo.

Consideremos la conexión del host A al host C pasando a través de los routers R1 y R2. La conexión A-C comparte el router R1 con la conexión D-B y comparte el router R2 con la conexión B-D. Para valores extremadamente pequeños de λ_{in} , es raro que el buffer se desborde (como en los dos escenarios de congestión anteriores), y la tasa de transferencia es aproximadamente igual a

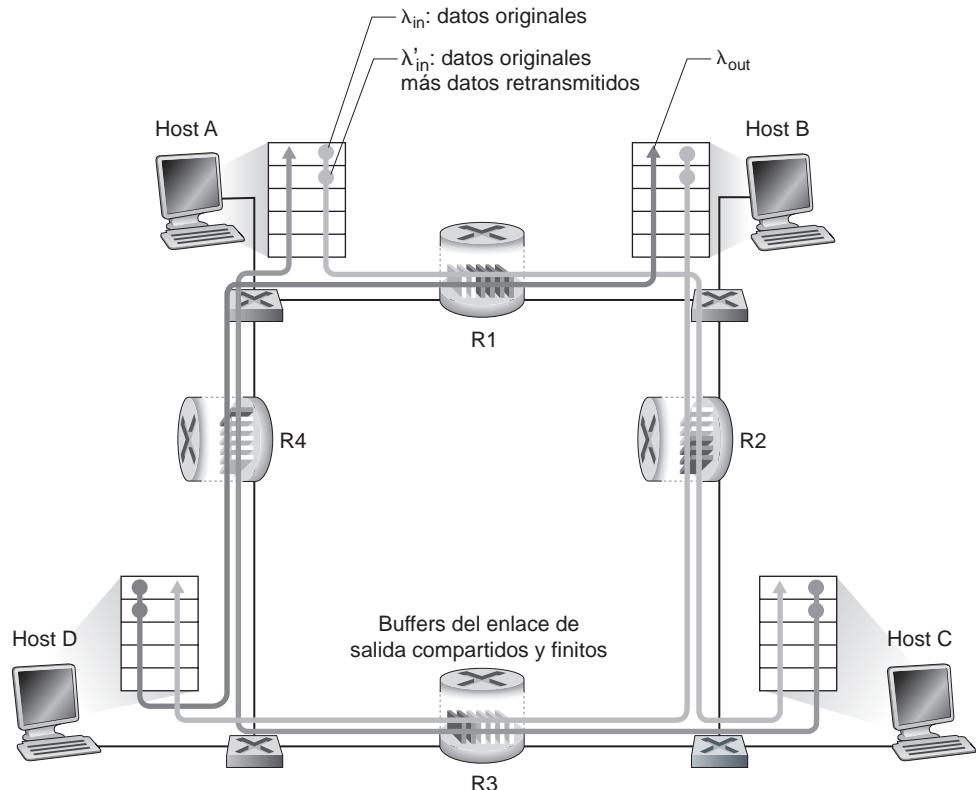


Figura 3.47 ♦ Cuatro emisores, routers con buffers finitos y rutas con varios saltos.

la carga ofrecida. Para valores ligeramente más grandes de λ_{in} , la correspondiente tasa de transferencia es también más grande, ya que se están transmitiendo más datos originales por la red y entregándose en su destino, y los desbordamientos siguen siendo raros. Por tanto, para valores pequeños de λ_{in} , un incremento de λ_{in} da lugar a un incremento de λ_{out} .

Una vez considerado el caso de tráfico extremadamente bajo, pasemos a examinar el caso en que λ_{in} (y por tanto λ'_{in}) es extremadamente grande. Sea el router R2. El tráfico de A–C que llega al router R2 (el que llega a R2 después de ser reenviado desde R1) puede tener una velocidad de llegada a R2 que es como máximo R , la capacidad del enlace de R1 a R2, independientemente del valor de λ_{in} . Si λ'_{in} es extremadamente grande en todas las conexiones (incluyendo la conexión B–D), entonces la velocidad de llegada del tráfico de B–D a R2 puede ser mucho mayor que la del tráfico de A–C. Puesto que los tráficos de A–C y B–D tienen que competir en el router R2 por el espacio limitado disponible en el buffer, la cantidad de tráfico de A–C que consiga atravesar con éxito R2 (es decir, que no se pierda por desbordamiento del buffer) será cada vez menor a medida que la carga ofrecida por la conexión B–D aumente. En el límite, a medida que la carga ofrecida se aproxima a infinito, un buffer vacío de R2 es llenado de forma inmediata por un paquete de B–D y la tasa de transferencia de la conexión A–C en R2 tiende a cero. Esto, a su vez, *implica que la tasa de transferencia terminal a terminal de A–C tiende a cero* en el límite correspondiente a una situación de tráfico intenso. Estas consideraciones dan lugar a la relación de compromiso entre la carga ofrecida y la tasa de transferencia que se muestra en la Figura 3.48.

La razón del eventual decrecimiento de la tasa de transferencia al aumentar la carga ofrecida es evidente cuando se considera la cantidad de trabajo desperdiciado realizado por la red. En el escenario descrito anteriormente en el que había una gran cantidad de tráfico, cuando un paquete se descartaba en un router de segundo salto, el trabajo realizado por el router del primer salto al encaminar un paquete al router del segundo salto terminaba siendo “desperdiciado”. Para eso, hubiera dado igual que el primer router simplemente hubiera descartado dicho paquete y hubiera permanecido inactivo, porque de todos modos el paquete no llegaría a su destino. Aún más, la capacidad de transmisión utilizada en el primer router para reenviar el paquete al segundo router podría haber sido mejor aprovechada si se hubiera empleado para transmitir un paquete diferente (por ejemplo, al seleccionar un paquete para transmitirlo, puede resultar mejor para un router dar la prioridad a los paquetes que ya hayan pasado por varios routers anteriormente). *Por tanto, aquí nos encontramos con otro de los costes de descartar un paquete a causa de la congestión de la red: cuando un paquete se descarta a lo largo de una ruta, la capacidad de transmisión empleada en cada uno de los enlaces anteriores para encaminar dicho paquete hasta el punto en el que se ha descartado termina por desperdiciarse.*

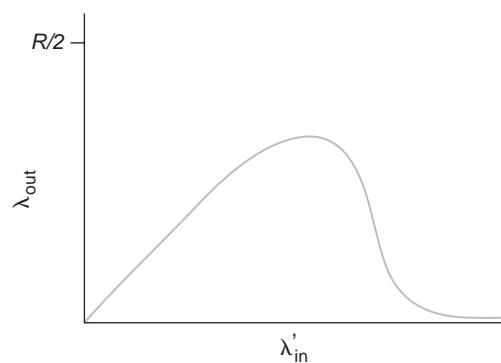


Figura 3.48 ♦ Escenario 3: rendimiento con buffers finitos y rutas con varios saltos.

3.6.2 Métodos para controlar la congestión

En la Sección 3.7 examinaremos en gran detalle el método específico de TCP para controlar la congestión. Ahora, vamos a identificar los dos métodos más comunes de control de congestión que se utilizan en la práctica y abordaremos las arquitecturas de red específicas y los protocolos de control de congestión que se integran en estos métodos.

En el nivel más general, podemos diferenciar entre las técnicas de control de congestión basándonos en si la capa de red proporciona alguna ayuda explícita a la capa de transporte con propósitos de controlar la congestión:

- *Control de congestión terminal a terminal.* En este método, la capa de red no proporciona soporte explícito a la capa de transporte para propósitos de control de congestión. Incluso la presencia de congestión en la red tiene que ser inferida por los sistemas terminales basándose únicamente en el comportamiento observado de la red (por ejemplo, la pérdida de paquetes y los retardos). En la Sección 3.7.1 veremos que TCP tiene que aplicar este método de control de congestión terminal a terminal, ya que la capa IP no proporciona ninguna realimentación a los hosts relativa a la congestión de la red. La pérdida de segmentos TCP (indicada por un fin de temporización o por la recepción de un triple paquete ACK duplicado) se toma como indicación de que existe congestión en la red, por lo que TCP reduce el tamaño de su ventana en consecuencia. También veremos una propuesta más reciente para abordar el control de congestión de TCP que utiliza valores de retardo de ida y vuelta crecientes como indicadores de que existe una mayor congestión de red.
- *Control de congestión asistido por la red.* En este método de control de congestión, los routers proporcionan una realimentación explícita al emisor y/o receptor informando del estado de congestión en la red. Esta realimentación puede ser tan simple como un único bit que indica que existe congestión en un enlace. Este método se aplicó en las tempranas arquitecturas de red SNA de IBM [Schwartz 1982] y DECnet de DEC [Jain 1989; Ramakrishnan 1990] y ATM [Black 1995]. También es posible proporcionar una realimentación de red más sofisticada. Por ejemplo, una forma del mecanismo de control de congestión de ABR (*Available Bit Rate*) en las redes ATM permite a un router informar al emisor de la velocidad máxima de transmisión de host que el router puede soportar en un enlace saliente. Como hemos mencionado anteriormente, las versiones predeterminadas de Internet de IP y TCP adoptan un método terminal a terminal para llevar a cabo el control de congestión. Sin embargo, veremos en la Sección 3.7.2 que, recientemente, IP y TCP pueden implementar de forma opcional un mecanismo de control de congestión asistido por la red.

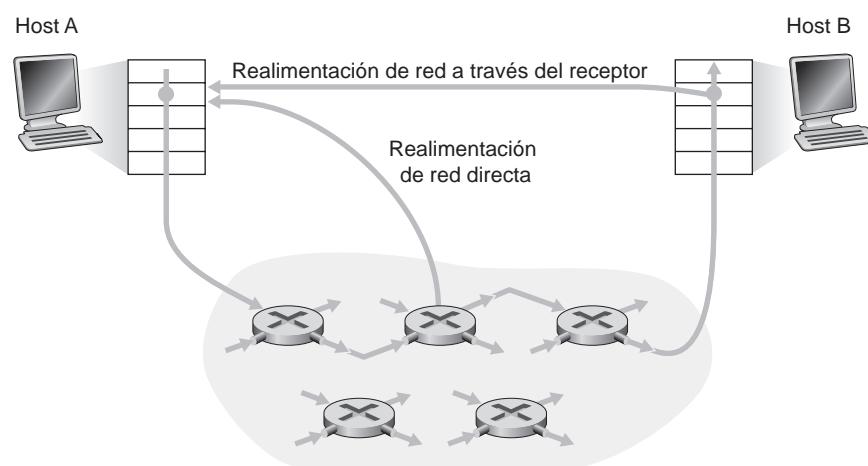


Figura 3.49 ♦ Dos rutas de realimentación de la información de la congestión asistida por la red.

En el mecanismo de control de congestión asistido por la red, la información acerca de la congestión suele ser realimentada de la red al emisor de una de dos formas, como se ve en la Figura 3.49. La realimentación directa puede hacerse desde un router de la red al emisor. Esta forma de notificación, normalmente, toma la forma de un paquete de asfixia o bloqueo (*choke packet*) (que esencialmente dice “¡Estoy congestionada!”). La segunda forma de notificación, más común, tiene lugar cuando un router marca/actualiza un campo de un paquete que se transmite del emisor al receptor para indicar que existe congestión. Después de recibir un paquete marcado, el receptor notifica al emisor la existencia de congestión. Observe que esta última forma de notificación tarda al menos un periodo igual al tiempo de ida y vuelta completo.

3.7 Mecanismo de control de congestión de TCP

En esta sección vamos a continuar con nuestro estudio de TCP. Como hemos visto en la Sección 3.5, TCP proporciona un servicio de transporte fiable entre dos procesos que se ejecutan en hosts diferentes. Otro componente clave de TCP es su mecanismo de control de congestión. Como hemos mencionado en la sección anterior, TCP tiene que utilizar un control de congestión terminal a terminal en lugar de un control de congestión asistido por la red, ya que la capa IP no proporciona una realimentación explícita a los sistemas terminales en lo tocante a la congestión de la red.

El método empleado por TCP consiste en que cada emisor limite la velocidad a la que transmite el tráfico a través de su conexión en función de la congestión de red percibida. Si un emisor TCP percibe que en la ruta entre él mismo y el destino apenas existe congestión, entonces incrementará su velocidad de transmisión; por el contrario, si el emisor percibe que existe congestión a lo largo de la ruta, entonces reducirá su velocidad de transmisión. Pero este método plantea tres cuestiones. En primer lugar, ¿cómo limita el emisor TCP la velocidad a la que envía el tráfico a través de su conexión? En segundo lugar, ¿cómo percibe el emisor TCP que existe congestión en la ruta entre él mismo y el destino? Y, tercero, ¿qué algoritmo deberá emplear el emisor para variar su velocidad de transmisión en función de la congestión percibida terminal a terminal?

Examinemos en primer lugar cómo un emisor TCP limita la velocidad a la que envía el tráfico a través de su conexión. En la Sección 3.5, hemos visto que cada lado de una conexión TCP consta de un buffer de recepción, un buffer de transmisión y varias variables (`UltimoByteLeido`, `VentRecepcion`, etc.). El mecanismo de control de congestión de TCP que opera en el emisor hace un seguimiento de una variable adicional, la **ventana de congestión**. Esta ventana, indicada como `VentCongestion`, impone una restricción sobre la velocidad a la que el emisor TCP puede enviar tráfico a la red. Específicamente, la cantidad de datos no reconocidos en un emisor no puede exceder el mínimo de entre `VentCongestion` y `VentRecepcion`, es decir:

$$\text{UltimoByteEnviado} - \text{UltimoByteReconocido} \leq \min\{\text{VentCongestion}, \text{VentRecepcion}\}$$

Con el fin de centrarnos en el mecanismo de control de congestión (en oposición al control de flujo), vamos a suponer que el buffer de recepción TCP es tan grande que la restricción de la ventana de recepción puede ignorarse; por tanto, la cantidad de datos no reconocidos por el emisor queda limitada únicamente por `VentCongestion`. Supondremos también que el emisor siempre tiene datos que enviar; es decir, todos los segmentos de la ventana de congestión son enviados.

La restricción anterior limita la cantidad de datos no reconocidos por el emisor y, por tanto, limita de manera indirecta la velocidad de transmisión del emisor. Para comprender esto imagine una conexión en la que tanto la pérdida de paquetes como los retardos de transmisión sean despreciables. En esta situación, lo que ocurre a grandes rasgos es lo siguiente: al inicio de cada periodo RTT, la restricción permite al emisor enviar `VentCongestion` bytes de datos a través de la conexión; al final del periodo RTT, el emisor recibe los paquetes ACK correspondientes a los datos. *Por tanto, la velocidad de transmisión del emisor es aproximadamente igual a `VentCongestion/RTT bytes/`*

segundo. Ajustando el valor de la ventana de congestión, el emisor puede ajustar la velocidad a la que transmite los datos a través de su conexión.

Veamos ahora cómo percibe un emisor TCP que existe congestión en la ruta entre él mismo y el destino. Definamos un “suceso de pérdida” en un emisor TCP como el hecho de que se produzca un fin de temporización o se reciban tres paquetes ACK duplicados procedentes del receptor (recuerde la exposición de la Sección 3.5.4 sobre el suceso de fin de temporización mostrado en la Figura 3.33 y la subsiguiente modificación para incluir la retransmisión rápida a causa de la recepción de tres paquetes ACK duplicados). Cuando existe una congestión severa, entonces uno o más de los buffers de los routers existentes a lo largo de la ruta pueden desbordarse, dando lugar a que un datagrama (que contenga un segmento TCP) sea descartado. A su vez, el datagrama descartado da lugar a un suceso de pérdida en el emisor (debido a un fin de temporización o a la recepción de tres paquetes ACK duplicados), el cual lo interpreta como una indicación de que existe congestión en la ruta entre el emisor y el receptor.

Ahora que ya hemos visto cómo se detecta la existencia de congestión en la red, vamos a considerar el mejor de los casos, cuando no existe congestión en la red, es decir, cuando no se producen pérdidas de paquetes. En este caso, el emisor TCP recibirá los paquetes de reconocimiento ACK correspondientes a los segmentos anteriormente no reconocidos. Como veremos, TCP interpretará la llegada de estos paquetes ACK como una indicación de que todo está bien (es decir, que los segmentos que están siendo transmitidos a través de la red están siendo entregados correctamente al destino) y empleará esos paquetes de reconocimiento para incrementar el tamaño de la ventana de congestión (y, por tanto, la velocidad de transmisión). Observe que si la velocidad de llegada de los paquetes ACK es lenta, porque por ejemplo la ruta terminal a terminal presenta un retardo grande o contiene un enlace con un ancho de banda pequeño, entonces el tamaño de la ventana de congestión se incrementará a una velocidad relativamente lenta. Por el contrario, si la velocidad de llegada de los paquetes de reconocimiento es alta, entonces el tamaño de la ventana de congestión se incrementará más rápidamente. Puesto que TCP utiliza los paquetes de reconocimiento para provocar (o temporizar) sus incrementos del tamaño de la ventana de congestión, se dice que TCP es **auto-temporizado**.

Conocido el *mecanismo* de ajuste del valor de `VentCongestion` para controlar la velocidad de transmisión, la cuestión crítica que nos queda es: *¿cómo* debería un emisor TCP determinar su velocidad de transmisión? Si los emisores TCP de forma colectiva transmiten a velocidades demasiado altas pueden congestionar la red, llevándola al tipo de colapso de congestión que hemos visto en la Figura 3.48. De hecho, la versión de TCP que vamos a estudiar a continuación fue desarrollada en respuesta al colapso de congestión observado en Internet [Jacobson 1988] en las versiones anteriores de TCP. Sin embargo, si los emisores TCP son demasiado cautelosos y transmiten la información muy lentamente, podrían infrautilizar el ancho de banda de la red; es decir, los emisores TCP podrían transmitir a velocidades más altas sin llegar a congestionar la red. Entonces, *¿cómo* pueden determinar los emisores TCP sus velocidades de transmisión de manera que no congestionen la red a la vez que hacen uso del todo el ancho de banda disponible? ¿Están los emisores TCP explícitamente coordinados, o existe un método distribuido en el que dichos emisores TCP puedan establecer sus velocidades de transmisión basándose únicamente en la información local? TCP responde a estas preguntas basándose en los siguientes principios:

- *Un segmento perdido implica congestión y, por tanto, la velocidad del emisor TCP debe reducirse cuando se pierde un segmento.* Recuerde que en la Sección 3.5.4 hemos visto que un suceso de fin de temporización o la recepción de cuatro paquetes de reconocimiento para un segmento dado (el paquete ACK original y los tres duplicados) se interpreta como una indicación implícita de que se ha producido un “suceso de pérdida” del segmento que sigue al segmento que ha sido reconocido cuatro veces, activando el proceso de retransmisión del segmento perdido. Desde el punto de vista del mecanismo de control de congestión, la cuestión es cómo el emisor TCP debe disminuir el tamaño de su ventana de congestión y, por tanto, su velocidad de transmisión, en respuesta a este suceso de pérdida inferido.

- Un segmento que ha sido reconocido indica que la red está entregando los segmentos del emisor al receptor y, por tanto, la velocidad de transmisión del emisor puede incrementarse cuando llega un paquete ACK correspondiente a un segmento que todavía no había sido reconocido. La llegada de paquetes de reconocimiento se interpreta como una indicación implícita de que todo funciona bien (los segmentos están siendo entregados correctamente del emisor al receptor y la red por tanto no está congestionada). Luego se puede aumentar el tamaño de la ventana de congestión.
- Tanteo del ancho de banda. Puesto que los paquetes ACK indican que la ruta entre el origen y el destino está libre de congestión y la pérdida de paquetes indica que hay una ruta congestionada, la estrategia de TCP para ajustar su velocidad de transmisión consiste en incrementar su velocidad en respuesta a la llegada de paquetes ACK hasta que se produce una pérdida, momento en el que reduce la velocidad de transmisión. El emisor TCP incrementa entonces su velocidad de transmisión para tantear la velocidad a la que de nuevo aparece congestión, retrocede a partir de ese punto y comienza de nuevo a tantear para ver si ha variado la velocidad a la que comienza de nuevo a producirse congestión. El comportamiento del emisor TCP es quizás similar a la del niño que pide (y consigue) una y otra vez golosinas hasta que se le dice “¡No!”, momento en el que da marcha atrás, pero enseguida comienza otra vez a pedir más golosinas. Observe que la red no proporciona una indicación explícita del estado de congestión (los paquetes ACK y las pérdidas se utilizan como indicadores implícitos) y que cada emisor TCP reacciona a la información local de forma asíncrona con respecto a otros emisores TCP.

Ahora que ya conocemos los fundamentos del mecanismo de control de congestión de TCP, estamos en condiciones de pasar a estudiar los detalles del famoso **algoritmo de control de congestión de TCP**, que fue descrito por primera vez en el libro de [Jacobson 1988] y que está estandarizado en el documento [RFC 5681]. El algoritmo consta de tres componentes principales: (1) arranque lento (*slow start*), (2) evitación de la congestión (*congestion avoidance*) y (3) recuperación rápida (*fast recovery*). Los dos primeros componentes son obligatorios en TCP, diferenciándose en la forma en que aumentan el tamaño de la ventana de congestión en respuesta a los paquetes ACK recibidos. Vamos a ver brevemente que el arranque lento incrementa el tamaño de la ventana de congestión más rápidamente (¡contrariamente a lo que indica su nombre!) que la evitación de la congestión. El componente recuperación rápida es recomendable, aunque no obligatorio, para los emisores TCP.

Arranque lento

Cuando se inicia una conexión TCP, el valor de la ventana de congestión (`VentCongestion`) normalmente se inicializa con un valor pequeño igual a 1 MSS (tamaño máximo de segmento) [RFC 3390], que da como resultado una velocidad de transmisión inicial aproximadamente igual a MSS/RTT . Por ejemplo, si $\text{MSS} = 500$ bytes y $\text{RTT} = 200$ milisegundos, la velocidad de transmisión inicial será aproximadamente de 20 kbps. Puesto que el ancho de banda disponible para el emisor TCP puede ser mucho más grande que el valor de MSS/RTT , al emisor TCP le gustaría poder determinar rápidamente la cantidad de ancho de banda disponible. Por tanto, en el estado de **arranque lento**, el valor de `VentCongestion` se establece en 1 MSS y se incrementa 1 MSS cada vez que se produce el primer reconocimiento de un segmento transmitido. En el ejemplo de la Figura 3.50, TCP envía el primer segmento a la red y espera el correspondiente paquete ACK. Cuando llega dicho paquete de reconocimiento, el emisor TCP incrementa el tamaño de la ventana de congestión en 1 MSS y transmite dos segmentos de tamaño máximo. Estos segmentos son entonces reconocidos y el emisor incrementa el tamaño de la ventana de congestión en 1 MSS por cada uno de los segmentos de reconocimiento, generando así una ventana de congestión de 4 MSS, etc. Este proceso hace que la velocidad de transmisión se duplique en cada periodo RTT. Por tanto, la velocidad de transmisión inicial de TCP es baja, pero crece exponencialmente durante esa fase de arranque lento.

Pero, ¿cuándo debe finalizar este crecimiento exponencial? El algoritmo del arranque lento proporciona varias respuestas a esta cuestión. En primer lugar, si se produce un suceso de pérdida de

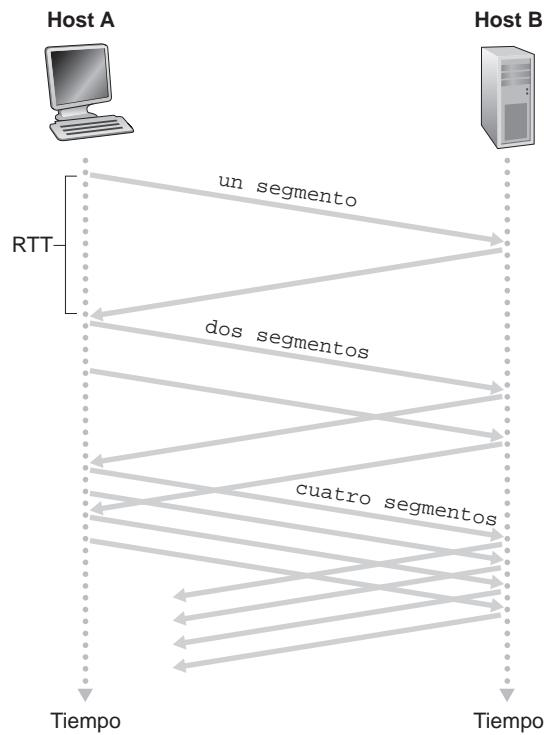


Figura 3.50 ♦ Fase de arranque lento de TCP.

paquete (es decir, si hay congestión) señalado por un fin de temporización, el emisor TCP establece el valor de `VentCongestion` en 1 e inicia de nuevo un proceso de arranque lento. También define el valor de una segunda variable de estado, `umbralAL`, que establece el umbral del arranque lento en `VentCongestion/2`, la mitad del valor del tamaño de la ventana de congestión cuando se ha detectado que existe congestión. La segunda forma en la que la fase de arranque lento puede terminar está directamente ligada al valor de `umbralAL`. Dado que `umbralAL` es igual a la mitad del valor que `VentCongestion` tenía cuando se detectó congestión por última vez, puede resultar algo imprudente continuar duplicando el valor de `VentCongestion` cuando se alcanza o sobrepasa el valor de `umbral`. Por tanto, cuando el valor de `VentCongestion` es igual a `umbralAL`, la fase de arranque lento termina y las transacciones TCP pasan al modo de evitación de la congestión. Como veremos, TCP incrementa con más cautela el valor de `VentCongestion` cuando está en el modo de evitación de la congestión. La última forma en la que puede terminar la fase de arranque lento es si se detectan tres paquetes ACK duplicados, en cuyo caso TCP realiza una retransmisión rápida (véase la Sección 3.5.4) y entra en el estado de recuperación rápida, que veremos más adelante. El comportamiento de TCP en la fase de arranque lento se resume en la descripción de la FSM del control de congestión de TCP de la Figura 3.51. El algoritmo de arranque lento tiene sus raíces en [Jacobson 1988]; en [Jain 1986] se proponía, de forma independiente, un método similar al algoritmo de arranque lento.

Evitación de la congestión

Al entrar en el estado de evitación de la congestión, el valor de `VentCongestion` es aproximadamente igual a la mitad de su valor en el momento en que se detectó congestión por última vez (podemos estar, por tanto, al borde de la congestión). En consecuencia, en lugar de duplicar el valor de `VentCongestion` para cada RTT, TCP adopta un método más conservador e incrementa el valor de `VentCongestion` solamente en un MSS cada RTT [RFC 5681]. Esto puede llevarse

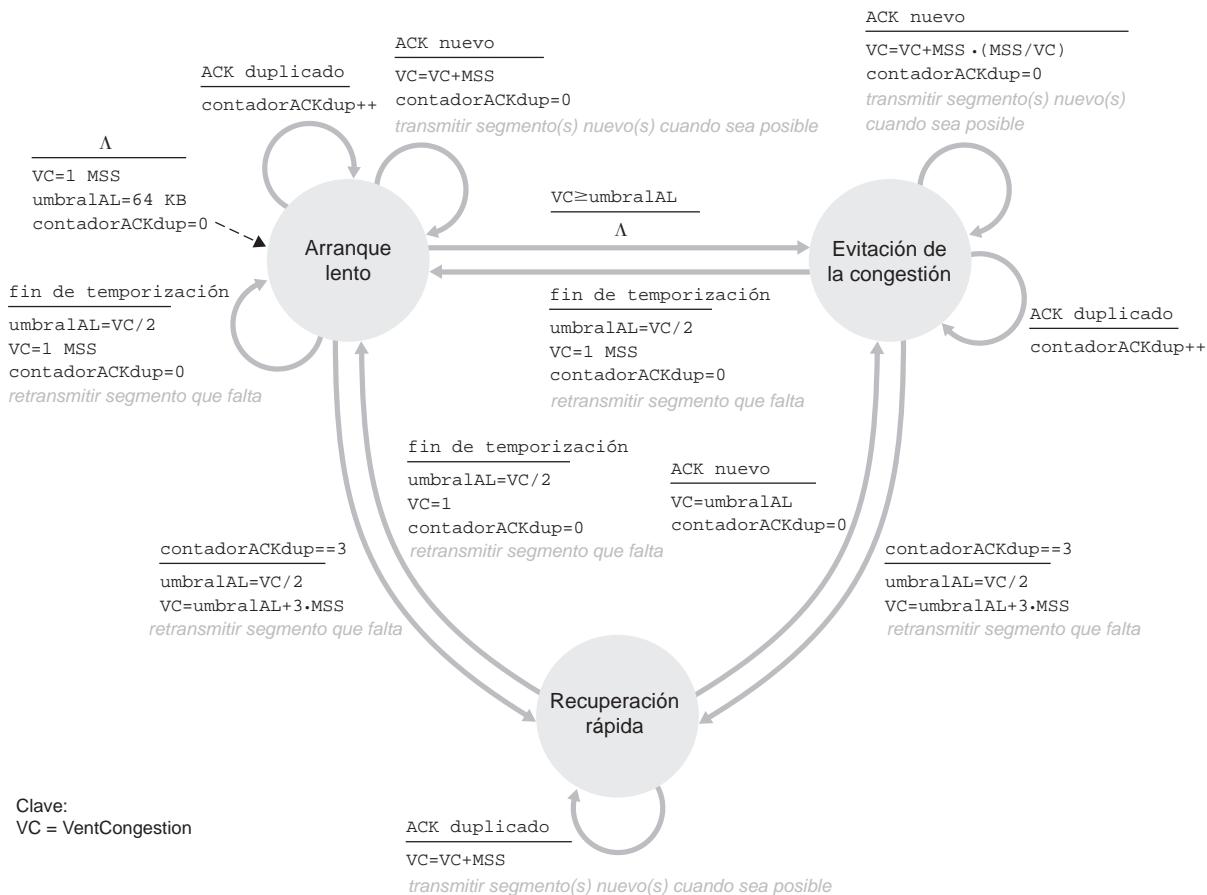


Figura 3.51 ♦ Descripción de la máquina de estados finitos del mecanismo de control de congestión de TCP.

a cabo de varias maneras. Un método habitual consiste en que un emisor TCP aumenta el valor de VentCongestion en MSS bytes (MSS/VentCongestion) cuando llega un nuevo paquete de reconocimiento. Por ejemplo, si MSS es igual a 1.460 bytes y VentCongestion es igual a 14.600 bytes, entonces se enviarán 10 segmentos en un RTT. Cada ACK que llega (suponiendo un ACK por segmento) incrementa el tamaño de la ventana de congestión en 1/10 MSS y, por tanto, el valor del tamaño de la ventana de congestión habrá aumentado en un MSS después de los ACK correspondientes a los 10 segmentos que hayan sido recibidos.

Pero, ¿en qué momento debería detenerse el crecimiento lineal (1 MSS por RTT) en el modo de evitación de la congestión? El algoritmo de evitación de la congestión de TCP se comporta del mismo modo que cuando tiene lugar un fin de temporización. Como en el caso del modo de arranque lento, el valor de VentCongestion se fija en 1 MSS y el valor de umbralAL se actualiza haciéndose igual a la mitad del valor de VentCongestion cuando se produce un suceso de pérdida de paquete. Sin embargo, recuerde que también puede detectarse una pérdida de paquete a causa de la llegada de tres ACK duplicados. En este caso, la red continúa entregando segmentos del emisor al receptor (como señala la recepción de paquetes ACK duplicados). Por tanto, el comportamiento de TCP ante este tipo de pérdida debería ser menos drástico que ante una pérdida de paquete indicada por un fin de temporización: TCP divide entre dos el valor de VentCongestion (añadiendo 3 MSS como forma de tener en cuenta los tres ACK duplicados que se han recibido) y configura el valor de umbralAL



Nota de video

Examen del comportamiento de TCP

para que sea igual a la mitad del valor que `VentCongestion` tenía cuando se recibieron los tres ACK duplicados. A continuación, se entra en el estado de recuperación rápida.

Recuperación rápida

En la fase de recuperación rápida, el valor de `VentCongestion` se incrementa en 1 MSS por cada ACK duplicado recibido correspondiente al segmento que falta y que ha causado que TCP entre en el estado de recuperación rápida. Cuando llega un ACK para el segmento que falta, TCP entra de nuevo en el estado de evitación de la congestión después de disminuir el valor de `VentCongestion`. Si se produce un fin de temporización, el mecanismo de recuperación rápida efectúa una transición al estado de arranque lento después de realizar las mismas acciones que en los modos de arranque lento y de evitación de la congestión: el valor de `VentCongestion` se establece en 1 MSS

EN LA PRÁCTICA

DIVISIÓN TCP: OPTIMIZACIÓN DEL RENDIMIENTO DE LOS SERVICIOS EN LA NUBE

Para servicios en la nube como los de búsqueda, correo electrónico y redes sociales, resulta deseable proporcionar un alto nivel de capacidad de respuesta, idealmente proporcionando a los usuarios la ilusión de que esos servicios se están ejecutando dentro de sus propios sistemas terminales (incluyendo sus teléfonos inteligentes). Esto puede constituir todo un desafío, ya que los usuarios están ubicados a menudo a mucha distancia de los centros de datos responsables de servir el contenido dinámico asociado con los servicios en la nube. De hecho, si el sistema terminal está lejos de un centro de datos, entonces el RTT será grande, lo que podría dar lugar a un tiempo de respuesta excesivo, debido al arranque lento de TCP.

Como caso de estudio, considere el retardo a la hora de recibir la respuesta a una consulta de búsqueda. Normalmente, el servidor necesita tres ventanas TCP durante el arranque lento para suministrar la respuesta [Pathak 2010]. Por tanto, el tiempo desde que un sistema terminal inicia una conexión TCP, hasta que recibe el último paquete de la respuesta, es de aproximadamente $4 \cdot \text{RTT}$ (un RTT para establecer la conexión TCP y tres RTT para las tres ventanas de datos), más el tiempo de procesamiento en el centro de datos. Estos retardos debidos al RTT pueden dar como resultado un retraso perceptible a la hora de devolver los resultados de las búsquedas, para un porcentaje significativo de las consultas. Además, puede haber un significativo porcentaje de pérdidas de paquetes en las redes de acceso, lo que provocaría retransmisiones TCP y retardos aún más grandes.

Una forma de mitigar este problema y mejorar la capacidad de respuesta percibida por el usuario consiste en (1) desplegar los servidores de *front-end* más cerca de los usuarios y (2) utilizar la **división TCP** (*TCP splitting*), descomponiendo la conexión TCP en el servidor de *front-end*. Con la división TCP, el cliente establece una conexión TCP con el *front-end* cercano, y el *front-end* mantiene una conexión TCP persistente con el centro de datos, con una ventana de congestión TCP muy grande [Tariq 2008, Pathak 2010, Chen 2011]. Con esta técnica, el tiempo de respuesta pasa a ser, aproximadamente, $4 \cdot \text{RTT}_{\text{FE}} + \text{RTT}_{\text{BE}} + \text{tiempo de procesamiento}$, donde RTT_{FE} es el tiempo de ida y vuelta entre el cliente y el servidor de *front-end* y RTT_{BE} es el tiempo de ida y vuelta entre el servidor de *front-end* y el centro de datos (servidor de *back-end*). Si el servidor de *front-end* está próximo al cliente, entonces este tiempo de respuesta es aproximadamente igual a RTT más el tiempo de procesamiento, ya que RTT_{FE} es despreciable y RTT_{BE} es aproximadamente RTT. Resumiendo, la técnica de división TCP permite reducir el retardo de red, aproximadamente, de 4·RTT a RTT, mejorando significativamente la capacidad de respuesta percibida por el usuario, en especial para aquellos usuarios que estén lejos de su centro de datos más próximo. La división TCP también ayuda a reducir los retardos de retransmisión TCP provocados por las pérdidas de paquetes en las redes de acceso. Google y Akamai han hecho un amplio uso de sus servidores CDN en las redes de acceso (recuerde nuestra exposición en la Sección 2.6) para llevar a cabo la división TCP para los servicios en la nube que soportan [Chen 2011].

y el valor de `umbralAL` se hace igual a la mitad del valor que tenía `VentCongestion` cuando tuvo lugar el suceso de pérdida.

El mecanismo de recuperación rápida es un componente de TCP recomendado, aunque no obligatorio [RFC 5681]. Es interesante resaltar que una versión anterior de TCP, conocida como **TCP Tahoe**, establece incondicionalmente el tamaño de la ventana de congestión en 1 MSS y entra en el estado de arranque lento después de un suceso de pérdida indicado por un fin de temporización o por la recepción de tres ACK duplicados. La versión más reciente de TCP, **TCP Reno**, incorpora la recuperación rápida.

La Figura 3.52 ilustra la evolución de la ventana de congestión de TCP para Reno y Tahoe. En esta figura, inicialmente el umbral es igual a 8 MSS. Durante los ocho primeros ciclos de transmisión, Tahoe y Reno realizan acciones idénticas. La ventana de congestión crece rápidamente de forma exponencial durante la fase de arranque lento y alcanza el umbral en el cuarto ciclo de transmisión. A continuación, la ventana de congestión crece linealmente hasta que se produce un suceso de tres ACK duplicados, justo después del octavo ciclo de transmisión. Observe que el tamaño de la ventana de congestión es igual a $12 \cdot MSS$ cuando se produce el suceso de pérdida. El valor de `umbralAL` se hace entonces igual a $0,5 \cdot \text{ventCongestion} = 6 \cdot MSS$. En TCP Reno, el tamaño de la ventana de congestión es puesto a `VentCongestion = 9 · MSS` y luego crece linealmente. En TCP Tahoe, la ventana de congestión es igual a 1 MSS y crece exponencialmente hasta que alcanza el valor de `umbralAL`, punto a partir del cual crece linealmente.

La Figura 3.51 presenta la descripción completa de la máquina de estados finitos de los algoritmos del mecanismo de control de congestión de TCP: arranque lento, evitación de la congestión y recuperación rápida. En la figura también se indica dónde pueden producirse transmisiones de nuevos segmentos y dónde retransmisiones de segmentos. Aunque es importante diferenciar entre las retransmisiones/control de errores de TCP y el control de congestión de TCP, también lo es apreciar cómo estos dos aspectos de TCP están estrechamente vinculados.

Control de congestión de TCP: retrospectiva

Una vez vistos los detalles de las fases de arranque lento, de evitación de la congestión y de recuperación rápida, merece la pena retroceder un poco para clarificar las cosas. Ignorando la fase inicial de arranque lento en la que se establece la conexión y suponiendo que las pérdidas están indicadas por la recepción de tres ACK duplicados en lugar de por fines de temporización, el control de congestión de TCP consiste en un crecimiento lineal (aditivo) de `VentCongestion` a razón de 1 MSS por RTT, seguido de un decrecimiento multiplicativo (división entre dos) del tamaño de la ventana, `VentCongestion`, cuando se reciben tres ACK duplicados. Por esta razón, suele decirse que el control de congestión de TCP es una forma de **crecimiento aditivo y decrecimiento multiplicativo**.

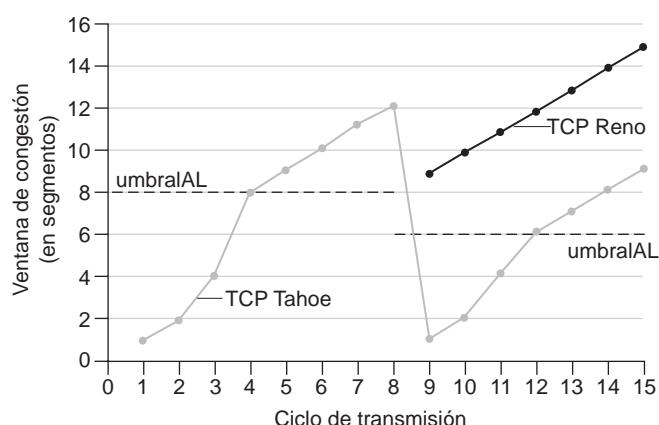


Figura 3.52 ♦ Evolución de la ventana de congestión de TCP (Tahoe y Reno).

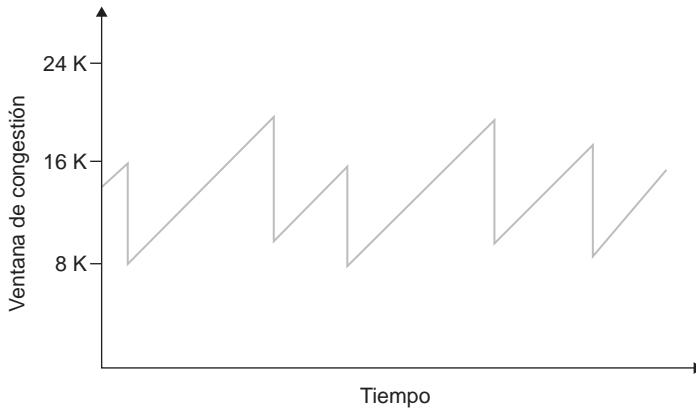


Figura 3.53 ♦ Control de congestión con crecimiento aditivo y decrecimiento multiplicativo.

(**AIMD, Additive-Increase, Multiplicative-Decrease**) de control de congestión. El control de congestión AIMD presenta un comportamiento en forma de “diente de sierra”, como se muestra en la Figura 3.53, lo que también ilustra nuestra anterior intuición de que TCP “va tanteando” el ancho de banda. (TCP aumenta linealmente el tamaño de su ventana de congestión, y por tanto su velocidad de transmisión, hasta que tiene lugar la recepción de tres ACK duplicados. A continuación, divide entre dos su ventana de congestión, pero vuelve después a crecer linealmente, tanteando para ver si hay disponible ancho de banda adicional.)

Como hemos mencionado anteriormente, la mayor parte de las implementaciones TCP actuales emplean el algoritmo Reno [Padhye 2001]. Se han propuesto muchas variantes del algoritmo Reno [RFC 3782; RFC 2018]. El algoritmo TCP Vegas [Brakmo 1995; Ahn 1995] intenta evitar la congestión manteniendo una buena tasa de transferencia. La idea básica del algoritmo Vegas es (1) detectar la congestión en los routers existentes entre el origen y el destino *antes* de que se pierda un paquete y (2) reducir la velocidad linealmente cuando se detecta una pérdida inminente de paquetes. La pérdida inminente de un paquete se predice observando el RTT. Cuanto mayor es el RTT de los paquetes, mayor es la congestión en los routers. A finales de 2015, la implementación Linux Ubuntu de TCP proporcionaba arranque lento, evitación de congestión, recuperación rápida, retransmisión rápida y SACK, de manera predeterminada; también se proporcionan algoritmos alternativos de control de congestión, como TCP Vegas y BIC [Xu 2004]. En [Afanasyev 2010] se proporciona un resumen de las muchas variantes de TCP.

El algoritmo AIMD de TCP fue desarrollado basándose en un enorme trabajo de ingeniería y de experimentación con los mecanismos de control de congestión en redes reales. Diez años después del desarrollo de TCP, los análisis teóricos mostraron que el algoritmo de control de congestión de TCP sirve como un algoritmo de optimización asíncrona distribuido que da como resultado la optimización simultánea de diversos aspectos importantes, tanto de las prestaciones proporcionadas al usuario como del rendimiento de la red [Kelly 1998]. Desde entonces se han desarrollado muchos aspectos teóricos del control de congestión [Srikant 2004].

Descripción macroscópica de la tasa de transferencia de TCP

Visto el comportamiento en diente de sierra de TCP, resulta natural preguntarse cuál es la tasa de transferencia media (es decir, la velocidad media) de una conexión TCP de larga duración. En este análisis vamos a ignorar las fases de arranque lento que tienen lugar después de producirse un fin de temporización (estas fases normalmente son muy cortas, ya que el emisor sale de ellas rápidamente de forma exponencial). Durante un intervalo concreto de ida y vuelta, la velocidad a la que TCP envía datos es función del tamaño de la ventana de congestión y del RTT actual.

Cuando el tamaño de la ventana es w bytes y el tiempo actual de ida y vuelta es RTT segundos, entonces la velocidad de transmisión de TCP es aproximadamente igual a w/RTT . TCP comprueba entonces si hay ancho de banda adicional incrementando w en 1 MSS cada RTT hasta que se produce una pérdida. Sea W el valor de w cuando se produce una pérdida. Suponiendo que RTT y W son aproximadamente constantes mientras dura la conexión, la velocidad de transmisión de TCP varía entre $W/(2 \cdot RTT)$ y W/RTT .

Estas suposiciones nos llevan a un modelo macroscópico extremadamente simplificado del comportamiento en régimen permanente de TCP. La red descarta un paquete de la conexión cuando la velocidad aumenta hasta W/RTT ; la velocidad entonces se reduce a la mitad y luego aumenta MSS/RTT cada RTT hasta que de nuevo alcanza el valor W/RTT . Este proceso se repite una y otra vez. Puesto que la tasa de transferencia (es decir, la velocidad) de TCP aumenta linealmente entre los dos valores extremos, tenemos

$$\text{tasa de transferencia media de una conexión} = \frac{0,75 \cdot W}{RTT}$$

Utilizando este modelo altamente idealizado para la dinámica del régimen permanente de TCP, también podemos deducir una expresión interesante que relaciona la tasa de pérdidas de una conexión con su ancho de banda disponible [Mahdavi 1997]. Dejamos esta demostración para los problemas de repaso. Un modelo más sofisticado que, según se ha comprobado, concuerda empíricamente con los datos medidos es el que se describe en [Padhye 2000].

TCP en rutas con un alto ancho de banda

Es importante darse cuenta de que el control de congestión de TCP ha ido evolucionando a lo largo de los años y todavía sigue evolucionando. Puede leer un resumen sobre las variantes de TCP actuales y una exposición acerca de la evolución de TCP en [Floyd 2001, RFC 5681, Afanasyev 2010]. Lo que era bueno para Internet cuando la mayoría de las conexiones TCP transportaban tráfico SMTP, FTP y Telnet no es necesariamente bueno para la Internet actual, en la que domina HTTP, o para una futura Internet que proporcione servicios que hoy ni siquiera podemos imaginar.

La necesidad de una evolución continua de TCP puede ilustrarse considerando las conexiones TCP de alta velocidad necesarias para las aplicaciones de computación reticular (*grid-computing*) y de computación en la nube. Por ejemplo, considere una conexión TCP con segmentos de 1.500 bytes y un RTT de 100 milisegundos y suponga que deseamos enviar datos a través de esta conexión a 10 Gbps. Siguiendo el documento [RFC 3649], observamos que utilizar la fórmula anterior para la tasa de transferencia de TCP, con el fin de alcanzar una tasa de transferencia de 10 Gbps, nos daría un tamaño medio de la ventana de congestión de 83.333 segmentos, que son *muchos* segmentos, lo que despierta el temor a que uno de esos 83.333 segmentos en tránsito pueda perderse. ¿Qué ocurriría si se produjeran pérdidas? O, dicho de otra manera, ¿qué fracción de los segmentos transmitidos podría perderse que permitiera al algoritmo de control de congestión de TCP de la Figura 3.51 alcanzar la velocidad deseada de 10 Gbps? En las cuestiones de repaso de este capítulo, mostraremos cómo derivar una fórmula que expresa la tasa de transferencia de una conexión TCP en función de la tasa de pérdidas (L), el tiempo de ida y vuelta (RTT) y el tamaño máximo de segmento (MSS):

$$\text{tasa de transferencia media de una conexión} = \frac{1,22 \cdot MSS}{RTT \sqrt{L}}$$

Con esta fórmula, podemos ver que para alcanzar una tasa de transferencia de 10 Gbps, el actual algoritmo de control de congestión de TCP solo puede tolerar una probabilidad de pérdida de segmentos de $2 \cdot 10^{-10}$ (o, lo que es equivalente, un suceso de pérdida por cada 5.000.000.000 segmentos), lo cual es una tasa muy baja. Esta observación ha llevado a una serie de investigadores a buscar nuevas versiones de TCP que estén diseñadas específicamente para estos entornos de alta

velocidad; consulte [Jin 2004; Kelly 2003; Ha 2008; RFC 7323] para obtener información sobre estos trabajos.

3.7.1 Equidad

Considere ahora K conexiones TCP, cada una de ellas con una ruta terminal a terminal diferente, pero atravesando todas ellas un enlace de cuello de botella con una velocidad de transmisión de R bps (con *enlace de cuello de botella* queremos decir que, para cada conexión, todos los restantes enlaces existentes a lo largo de la ruta de la conexión no están congestionados y tienen una capacidad de transmisión grande comparada con la capacidad de transmisión del enlace de cuello de botella). Suponga que cada conexión está transfiriendo un archivo de gran tamaño y que no existe tráfico UDP atravesando el enlace de cuello de botella. Se dice que un mecanismo de control de congestión es *equitativo* si la velocidad media de transmisión de cada conexión es aproximadamente igual a R/K ; es decir, cada conexión obtiene la misma cuota del ancho de banda del enlace.

¿Es el algoritmo AIMD de TCP equitativo, teniendo en cuenta que diferentes conexiones TCP pueden iniciarse en instantes distintos y, por tanto, pueden tener distintos tamaños de ventana en un instante determinado? [Chiu 1989] proporciona una explicación elegante e intuitiva de por qué el control de congestión de TCP converge para proporcionar la misma cuota de ancho de banda de un enlace de cuello de botella a las conexiones TCP que compiten por el ancho de banda.

Consideremos el caso simple de dos conexiones TCP que comparten un mismo enlace, cuya velocidad de transmisión es R , como se muestra en la Figura 3.54. Suponemos que las dos conexiones tienen el mismo MSS y el mismo RTT (por lo que si tienen el mismo tamaño de ventana de congestión, entonces tienen la misma tasa de transferencia). Además, tienen que enviar una gran cantidad de datos y ninguna otra conexión TCP ni datagrama UDP atraviesan este enlace compartido. Asimismo, vamos a ignorar la fase de arranque lento de TCP y vamos a suponer que las conexiones TCP están operando en el modo de evitación de la congestión (AIMD) durante todo el tiempo.

La gráfica de la Figura 3.55 muestra la tasa de transferencia de las dos conexiones TCP. Si TCP está diseñado para que las dos conexiones comparten equitativamente el ancho de banda del enlace, entonces la tasa de transferencia alcanzada debería caer a lo largo de la flecha que sale del origen con un ángulo de 45 grados (cuota equitativa de ancho de banda). Idealmente, la suma de las dos tasas de transferencia tiene que ser igual a R (evidentemente, que cada conexión reciba una cuota equitativa de la capacidad del enlace, pero igual a cero, no es una situación deseable). Por tanto, el objetivo debería ser que las tasas de transferencia alcanzadas se encuentren en algún punto próximo a la intersección de la línea de cuota equitativa de ancho de banda con la línea de utilización del ancho de banda completo mostradas en la Figura 3.55.

Suponga que los tamaños de ventana de TCP son tales que, en un instante determinado, las conexiones 1 y 2 alcanzan las tasas de transferencia indicadas por el punto A de la Figura 3.55. Puesto que la cantidad de ancho de banda del enlace conjunto consumido por las dos conexiones es menor que R , no se producirá ninguna pérdida y ambas conexiones incrementarán sus tamaños de ventana en

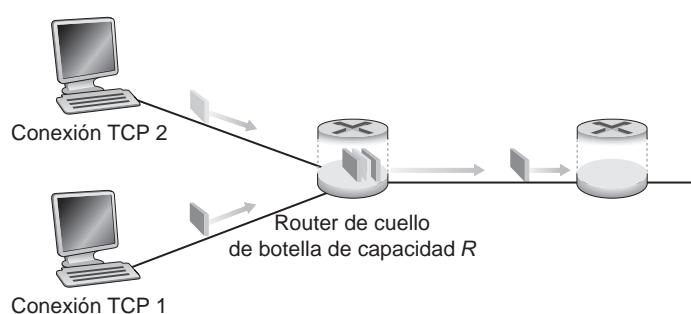


Figura 3.54 ♦ Dos conexiones TCP que comparten un mismo enlace cuello de botella.

1 MSS por RTT como resultado del algoritmo de evitación de la congestión de TCP. Por tanto, la tasa de transferencia conjunta de las dos conexiones sigue la línea de 45 grados (incremento equitativo para ambas conexiones) partiendo del punto A. Finalmente, el ancho de banda del enlace consumido conjuntamente por las dos conexiones será mayor que R , por lo que terminará produciéndose una pérdida de paquetes. Suponga que las conexiones 1 y 2 experimentan una pérdida de paquetes cuando alcanzan las tasas de transferencia indicadas por el punto B. Entonces las conexiones 1 y 2 reducen el tamaño de sus ventanas en un factor de dos. Las tasas de transferencia resultantes se encuentran por tanto en el punto C, a medio camino de un vector que comienza en B y termina en el origen. Puesto que el ancho de banda conjunto utilizado es menor que R en el punto C, de nuevo las dos conexiones incrementan sus tasas de transferencia a lo largo de la línea de 45 grados partiendo de C. Finalmente, terminará por producirse de nuevo una pérdida de paquetes, por ejemplo, en el punto D, y las dos conexiones otra vez reducirán el tamaño de sus ventanas en un factor de dos, y así sucesivamente. Compruebe que el ancho de banda alcanzado por ambas conexiones fluctúa a lo largo de la línea que indica una cuota equitativa del ancho de banda. Compruebe también que las dos conexiones terminarán convergiendo a este comportamiento, independientemente de su posición inicial dentro de ese espacio bidimensional. Aunque en este escenario se han hecho una serie de suposiciones ideales, permite proporcionar una idea intuitiva de por qué TCP hace que el ancho de banda se reparta de forma equitativa entre las conexiones.

En este escenario ideal hemos supuesto que solo las conexiones TCP atraviesan el enlace de cuello de botella, que las conexiones tienen el mismo valor de RTT y que solo hay una conexión TCP asociada con cada pareja origen-destino. En la práctica, estas condiciones normalmente no se dan y las aplicaciones cliente-servidor pueden por tanto obtener cuotas desiguales del ancho de banda del enlace. En particular, se ha demostrado que cuando varias conexiones comparten un cuello de botella común, aquellas sesiones con un valor de RTT menor son capaces de apropiarse más rápidamente del ancho de banda disponible en el enlace, a medida que este va liberándose (es decir, abren más rápidamente sus ventanas de congestión) y por tanto disfrutan de una tasa de transferencia más alta que aquellas conexiones cuyo valor de RTT es más grande [Lakshman 1997].

Equidad y UDP

Acabamos de ver cómo el control de congestión de TCP regula la velocidad de transmisión de una aplicación mediante el mecanismo de la ventana de congestión. Muchas aplicaciones multimedia,

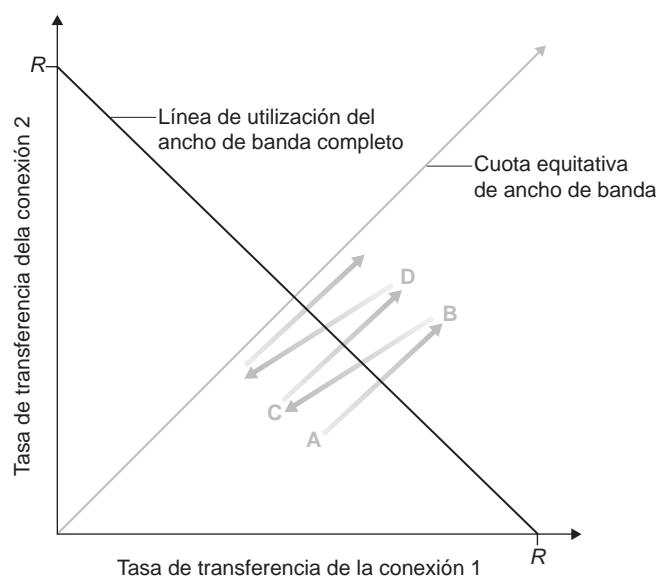


Figura 3.55 ♦ Tasa de transferencia alcanzada por las conexiones TCP 1 y 2.

como las videoconferencias y la telefonía por Internet, a menudo no se ejecutan sobre TCP precisamente por esta razón (no desean que su velocidad de transmisión se regule, incluso aunque la red esté muy congestionada). En lugar de ello, estas aplicaciones prefieren ejecutarse sobre UDP, que no incorpora un mecanismo de control de la congestión. Al ejecutarse sobre UDP, las aplicaciones pueden entregar a la red sus datos de audio y de vídeo a una velocidad constante y, ocasionalmente, perder paquetes, en lugar de reducir sus velocidades a niveles “equitativos” y no perder ningún paquete. Desde la perspectiva de TCP, las aplicaciones multimedia que se ejecutan sobre UDP no son equitativas (no cooperan con las demás conexiones ni ajustan sus velocidades de transmisión apropiadamente). Dado que el control de congestión de TCP disminuye la velocidad de transmisión para hacer frente a un aumento de la congestión (y de las pérdidas) y los orígenes de datos UDP no lo hacen, puede darse el caso de que esos orígenes UDP terminen por expulsar al tráfico TCP. Un área actual de investigación es el desarrollo de mecanismos de control de congestión para Internet que impidan que el tráfico UDP termine por reducir a cero la tasa de transferencia de Internet [Floyd 1999; Floyd 2000; Kohler 2006; RFC 4340].

Equidad y conexiones TCP en paralelo

Pero aunque se pudiera forzar al tráfico UDP a comportarse equitativamente, el problema de la equidad todavía no estaría completamente resuelto. Esto es porque no hay nada que impida a una aplicación basada en TCP utilizar varias conexiones en paralelo. Por ejemplo, los navegadores web a menudo utilizan varias conexiones TCP en paralelo para transferir los distintos objetos contenidos en una página web (en la mayoría de los navegadores puede configurarse el número exacto de conexiones múltiples). Cuando una aplicación emplea varias conexiones en paralelo obtiene una fracción grande del ancho de banda de un enlace congestionado. Por ejemplo, considere un enlace cuya velocidad es R que soporta nueve aplicaciones entrantes cliente-servidor, utilizando cada una de las aplicaciones una conexión TCP. Si llega una nueva aplicación y también emplea una conexión TCP, entonces cada conexión tendrá aproximadamente la misma velocidad de transmisión de $R/10$. Pero si esa nueva aplicación utiliza 11 conexiones TCP en paralelo, entonces la nueva aplicación obtendrá una cuota no equitativa de más de $R/2$. Dado que el tráfico web es el dominante en Internet, las conexiones múltiples en paralelo resultan bastante comunes.

3.7.2 Notificación explícita de congestión (ECN): control de congestión asistido por la red

Desde la estandarización inicial del arranque lento y de la evitación de la congestión a finales de la década de 1980 [RFC 1122], TCP ha implementado el tipo de control de congestión de extremo a extremo que hemos estudiado en la Sección 3.7.1: un emisor TCP no recibe ninguna indicación explícita de congestión por parte de la capa de red, deduciendo en su lugar la existencia de congestión a partir de la pérdida de paquetes observada. Más recientemente, se han propuesto, implementado e implantado extensiones tanto a IP como a TCP [RFC 3168] que permiten a la red señalizar explícitamente la congestión a un emisor y un receptor TCP. Este tipo de control de congestión asistido por la red se conoce con el nombre de **notificación explícita de congestión (ECN, Explicit Congestion Notification)**. Como se muestra en la Figura 3.56, están implicados los protocolos TCP e IP.

En la capa de red, se utilizan para ECN dos bits (lo que da un total de cuatro posibles valores) del campo Tipo de Servicio de la cabecera del datagrama IP (de la que hablaremos en la Sección 4.3). Una de las posibles configuraciones de los bits ECN es usada por los routers para indicar que están experimentando congestión. Esta indicación de congestión es transportada entonces en dicho datagrama IP hasta el host de destino, que a su vez informa al host emisor, como se muestra en la Figura 3.56. RFC 3168 no proporciona una definición de cuándo un router está congestionado; esa decisión es una opción de configuración proporcionada por el fabricante del router y sobre la cual decide el operador de la red. Sin embargo, RFC 3168 sí que recomienda que la indicación de

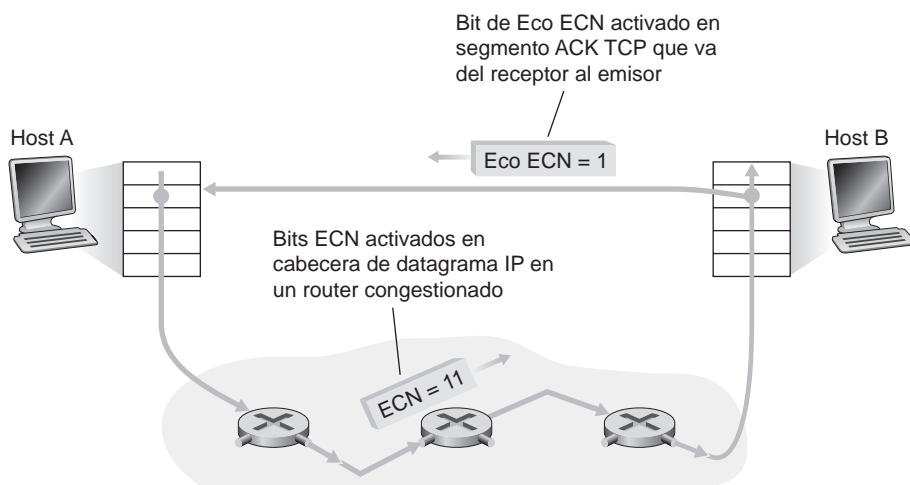


Figura 3.56 ♦ Notificación explícita de congestión: control de congestión asistido por la red.

congestión ECN solo se active cuando exista una congestión persistente. Una segunda configuración de los bits ECN es utilizada por el host emisor para informar a los routers de que el emisor y el receptor son compatibles con ECN, y por tanto capaces de llevar a cabo acciones en respuesta a las indicaciones de congestión de la red proporcionadas por ECN.

Como se muestra en la Figura 3.56, cuando la capa TCP en el host receptor detecta una indicación ECN de congestión en un datagrama recibido, informa de esa situación de congestión a la capa TCP del host emisor, activando el bit ECE (*Explicit Congestion Notification Echo*, bit de Eco ECN; véase la Figura 3.29) en un segmento ACK TCP enviado por el receptor al emisor. Al recibir un ACK con una indicación ECE de congestión, el emisor TCP reacciona a su vez dividiendo por dos la ventana de congestión, igual que lo haría al perderse un segmento mientras se usa retransmisión rápida, y activa el bit CWR (*Congestion Windows Reduced*, ventana de congestión reducida) en la cabecera del siguiente segmento TCP transmitido por el emisor hacia el receptor.

Además de TCP, también otros protocolos de la capa de transporte pueden hacer uso de la señalización ECN de la capa de red. El protocolo DCCP (*Datagram Congestion Control Protocol*, protocolo de control de congestión de datagramas) [RFC 4340] proporciona un servicio no fiable tipo UDP con control de congestión y baja sobrecarga administrativa, que emplea ECN. DCTCP (Data Center TCP, TCP para centros de datos) [Alizadeh 2010], que es una versión de TCP diseñada específicamente para redes de centros de datos, también utiliza ECN.

3.8 Resumen

Hemos comenzado este capítulo estudiando los servicios que un protocolo de la capa de transporte puede proporcionar a las aplicaciones de red. En uno de los extremos, el protocolo de capa de transporte puede ser muy simple y ofrecer un servicio poco sofisticado a las aplicaciones, poniendo a su disposición únicamente una función de multiplexación/demultiplexación para los procesos que se están comunicando. El protocolo UDP de Internet es un ejemplo de ese tipo de protocolo de la capa de transporte poco sofisticado. En el otro extremo, un protocolo de la capa de transporte puede proporcionar a las aplicaciones diversos tipos de garantías, como por ejemplo la de entrega fiable de los datos, garantías sobre los retardos y garantías concernientes al ancho de banda. De todos modos,

los servicios que un protocolo de transporte puede proporcionar están a menudo restringidos por el modelo de servicio del protocolo subyacente de la capa de red. Si el protocolo de la capa de red no puede proporcionar garantías sobre los retardos o de ancho de banda a los segmentos de la capa de transporte, entonces el protocolo de la capa de transporte no puede proporcionar garantías de retardo ni de ancho de banda a los mensajes intercambiados por los procesos.

En la Sección 3.4 hemos visto que un protocolo de la capa de transporte puede proporcionar una transferencia fiable de los datos incluso aunque el protocolo de red subyacente sea no fiable. Allí vimos que son muchas las sutilezas implicadas en la provisión de una transferencia fiable de los datos, pero que dicha tarea puede llevarse a cabo combinando cuidadosamente los paquetes de reconocimiento, los temporizadores, las retransmisiones y los números de secuencia.

Aunque hemos hablado de la transferencia fiable de los datos en este capítulo, debemos tener presente que esa transferencia fiable puede ser proporcionada por los protocolos de las capas de enlace, de red, de transporte o de aplicación. Cualquiera de las cuatro capas superiores de la pila de protocolos puede implementar los reconocimientos, los temporizadores, las retransmisiones y los números de secuencia, proporcionando así un servicio de transferencia de datos fiable a la capa que tiene por encima. De hecho, a lo largo de los años, los ingenieros e informáticos han diseñado e implementado de manera independiente protocolos de las capas de enlace, de red, de transporte y de aplicación que proporcionan una transferencia de datos fiable (aunque muchos de estos protocolos han desaparecido silenciosamente de la escena).

En la Sección 3.5 hemos examinado en detalle TCP, el protocolo fiable y orientado a la conexión de la capa de transporte de Internet. Hemos visto que TCP es bastante complejo, incluyendo técnicas de gestión de la conexión, de control de flujo y de estimación del tiempo de ida y vuelta, además de una transferencia fiable de los datos. De hecho, TCP es bastante más complejo de lo que nuestra descripción deja entrever; hemos dejado fuera de nuestra exposición, intencionadamente, diversos parches, correcciones y mejoras de TCP que están ampliamente implementadas en distintas versiones de dicho protocolo. De todos modos, todas estas complejidades están ocultas a ojos de la aplicación de red. Si el cliente en un host quiere enviar datos de forma fiable a un servidor implementado en otro host, se limita a abrir un socket TCP con el servidor y a bombear datos a través de dicho socket. Afortunadamente, la aplicación cliente-servidor es completamente inconsciente de toda la complejidad de TCP.

En la Sección 3.6 hemos examinado el control de congestión desde una perspectiva amplia, mientras que en la Sección 3.7 hemos mostrado cómo se implementa ese mecanismo de control de congestión en TCP. Allí vimos que el control de congestión es obligatorio para la buena salud de la red. Sin él, una red se puede colapsar fácilmente, sin que al final puedan transportarse datos de terminal a terminal. En la Sección 3.7 vimos que TCP implementa un mecanismo de control de congestión terminal a terminal que incrementa de forma aditiva su tasa de transmisión cuando se evalúa que la ruta seguida por la conexión TCP está libre de congestión, mientras que esa tasa de transmisión se reduce multiplicativamente cuando se producen pérdidas de datos. Este mecanismo también trata de proporcionar a cada conexión TCP que pasa a través de un enlace congestionado una parte equitativa del ancho de banda del enlace. También hemos examinado con cierta profundidad el impacto que el establecimiento de la conexión TCP y el lento arranque de la misma tienen sobre la latencia. Hemos observado que, en muchos escenarios importantes, el establecimiento de la conexión y el arranque lento contribuyen significativamente al retardo terminal a terminal. Conviene recalcar una vez más que, aunque el control de congestión de TCP ha ido evolucionando a lo largo de los años, continúa siendo un área intensiva de investigación y es probable que continúe evolucionando en los próximos años.

El análisis realizado en este capítulo acerca de los protocolos específicos de transporte en Internet se ha centrado en UDP y TCP, que son los dos “caballos de batalla” de la capa de transporte de Internet. Sin embargo, dos décadas de experiencia con estos dos protocolos han permitido identificar una serie de casos en los que ninguno de los dos resulta ideal. Los investigadores han dedicado, por tanto, grandes esfuerzos al desarrollo de protocolos adicionales de la capa de transporte, varios de los cuales son actualmente estándares propuestos por IETF.

El Protocolo de control de congestión para datagramas (DCCP, *Datagram Congestion Control Protocol*) [RFC 4340] proporciona un servicio no fiable con baja carga administrativa y orientado a mensajes, similar a UDP, pero que cuenta con un tipo de control de congestión seleccionado por la aplicación y que es compatible con TCP. Si una aplicación necesita una transferencia de datos fiable o semifiable, entonces el control de congestión sería implementado dentro de la propia aplicación, quizás utilizando los mecanismos que hemos estudiado en la Sección 3.4. DCCP está previsto para utilizarlo en aplicaciones tales como los flujos multimedia (véase el Capítulo 9) que pueden jugar con los compromisos existentes entre los requisitos de temporización y de fiabilidad en la entrega de datos, pero que quieran a la vez poder responder a situaciones de congestión en la red.

El protocolo QUIC (*Quick UDP Internet Connections*) de Google [Iyengar 2016], implementado en el navegador Chromium de Google, proporciona fiabilidad por medio de las retransmisiones, así como corrección de errores, configuración rápida de la conexión y un algoritmo de control de congestión basado en la velocidad que trata de ser compatible con TCP, todo ello implementado como un protocolo de la capa de aplicación por encima de UDP. A principios de 2015, Google informó de que aproximadamente la mitad de todas las solicitudes enviadas desde Chrome a los servidores Google se sirven a través de QUIC.

DCTCP (*Data Center TCP*) [Alizadeh 2010] es una versión de TCP diseñada específicamente para las redes de centros de datos y utiliza ECN para proporcionar un mejor soporte a la mezcla de flujos con tiempos de vida cortos y largos que caracterizan a las cargas de trabajo de los centros de datos.

El Protocolo de transmisión para control de flujos (SCTP, *Stream Control Transmission Protocol*) [RFC 4960, RFC 3286] es un protocolo fiable orientado a mensajes, que permite multiplexar diferentes “flujos” de nivel de aplicación a través de una única conexión SCTP (una técnica conocida con el nombre de “*multi-streaming*”). Desde el punto de vista de la fiabilidad, los diferentes flujos que comparten la conexión se gestionan de forma separada, de modo que la pérdida de paquetes en uno de los flujos no afecte a la entrega de los datos en los otros. SCTP también permite transferir datos a través de dos rutas de salida cuando un host está conectado a dos o más redes; también existe la posibilidad de la entrega opcional de datos fuera de orden, así como otra serie de características interesantes. Los algoritmos de control de flujo y de control de congestión de SCTP son prácticamente los mismos que en TCP.

El protocolo de Control de tasa compatible con TCP (TFRC, *TCP-Friendly Rate Control*) [RFC 5348] es un protocolo de control de congestión más que un protocolo completo de la capa de transporte. TFRC especifica un mecanismo de control de congestión que podría ser utilizado en algún otro protocolo de transporte, como DCCP (de hecho, uno de los dos protocolos seleccionables por la aplicación existentes en DCCP es TFRC). El objetivo de TFRC es suavizar el comportamiento típico en “diente de sierra” (véase la Figura 3.53) que se experimenta en el control de congestión de TCP, al mismo tiempo que se mantiene una tasa de transmisión a largo plazo “razonablemente” próxima a la TCP. Con una tasa de transmisión de perfil más suave que TCP, TFRC está bien adaptado a aplicaciones multimedia tales como la telefonía IP o los flujos multimedia, en donde es importante mantener ese perfil suave de la tasa de transmisión. TFRC es un protocolo “basado en ecuaciones” que utiliza la tasa medida de pérdida de paquetes como entrada para una ecuación [Padhye 2000] que permite estimar cuál sería la tasa de transferencia TCP si una sesión TCP experimentara dicha tasa de pérdidas. Entonces, dicha tasa de transferencia se adopta como objetivo de tasa de transmisión para TFRC.

Solo el futuro nos dirá si DCCP, SCTP, QUIC o TFRC serán adoptados ampliamente o no. Aunque estos protocolos proporcionan claramente una serie de capacidades mejoradas respecto a TCP y UDP, estos dos protocolos han demostrado ser a lo largo de los años “lo suficientemente buenos”. El que un “mejor” protocolo termine venciendo a otro que es “suficientemente bueno” dependerá de una compleja mezcla de aspectos técnicos, sociales y empresariales.

En el Capítulo 1 hemos visto que una red de computadoras puede dividirse entre lo que se denomina la “frontera de la red” y el “núcleo de la red”. La frontera de la red cubre todo lo que sucede en los sistemas terminales. Habiendo ya cubierto la capa de aplicación y la capa de transporte,

nuestro análisis de la frontera de la red está completo, así que ha llegado el momento de explorar el núcleo de la red. Comenzaremos nuestro viaje en los dos capítulos siguientes, donde estudiaremos la capa red, y seguiremos en el Capítulo 6 dedicado a la capa de enlace.

Problemas y cuestiones de repaso

Capítulo 3 Cuestiones de repaso

SECCIONES 3.1–3.3

- R1. Suponga que la capa de red proporciona el siguiente servicio: la capa de red del host de origen acepta un segmento con un tamaño máximo de 1.200 bytes y una dirección de host de destino de la capa de transporte. La capa de red garantiza la entrega del segmento a la capa de transporte en el host de destino. Suponga que en el host de destino pueden ejecutarse muchos procesos de aplicaciones de red.
 - a. Diseñe el protocolo de la capa de transporte más simple posible que entregue los datos de la aplicación al proceso deseado en el host de destino. Suponga que el sistema operativo del host de destino ha asignado un número de puerto de 4 bytes a cada proceso de aplicación en ejecución.
 - b. Modifique este protocolo de manera que proporcione una “dirección de retorno” al proceso de destino.
 - c. En sus protocolos, ¿la capa de transporte “tiene que hacer algo” en el núcleo de la red de computadoras?
- R2. Imagine una sociedad en la que todo el mundo perteneciera a una familia de seis miembros, todas las familias vivieran en su propia casa, cada casa tuviera una dirección única y cada persona de cada casa tuviera un nombre único. Imagine que esa sociedad dispone de un servicio de correos que transporta las cartas desde una vivienda de origen hasta una vivienda de destino. El servicio de correos requiere que (i) la carta se introduzca en un sobre y que (ii) la dirección de la casa de destino (y nada más) esté claramente escrita en el sobre. Suponga también que en cada familia hay un delegado que tiene asignada la tarea de recoger y distribuir las cartas a los restantes miembros de la familia. Las cartas no necesariamente proporcionan una indicación acerca de los destinatarios.
 - a. Partiendo de la solución del Problema R1, describa un protocolo que el delegado de la familia pueda utilizar para entregar las cartas de un miembro de la familia emisora a un miembro de la familia receptora.
 - b. En su protocolo, ¿el servicio de correos tiene que abrir el sobre y examinar la carta para proporcionar este servicio?
- R3. Considere una conexión TCP entre el host A y el host B. Suponga que los segmentos TCP que viajan del host A al host B tienen un número de puerto de origen x y un número de puerto de destino y . ¿Cuáles son los números de puerto de origen y de destino para los segmentos que viajan del host B al host A?
- R4. Describa por qué un desarrollador de aplicaciones puede decidir ejecutar una aplicación sobre UDP en lugar de sobre TCP.
- R5. ¿Por qué razón el tráfico de voz y de vídeo suele enviarse sobre TCP en lugar de sobre UDP en la Internet de hoy día? (*Sugerencia:* la respuesta que estamos buscando no tiene nada que ver con el mecanismo de control de congestión de TCP.)
- R6. ¿Es posible que una aplicación disfrute de una transferencia de datos fiable incluso si se ejecuta sobre UDP? En caso afirmativo, explique cómo.

- R7. Sea un proceso del host C que tiene un socket UDP con el número de puerto 6789. Suponga también que los hosts A y B envían cada uno de ellos un segmento UDP al host C con el número de puerto de destino 6789. ¿Serán dirigidos ambos segmentos al mismo socket del host C? En caso afirmativo, ¿cómo sabrá el proceso del host C que estos dos segmentos proceden de dos hosts distintos?
- R8. Suponga que un servidor web se ejecuta en el puerto 80 del host C. Suponga también que este servidor web utiliza conexiones persistentes y que actualmente está recibiendo solicitudes de dos hosts diferentes, A y B. ¿Están siendo enviadas todas las solicitudes al mismo socket del host C? Si están siendo pasadas a través de sockets diferentes, ¿utilizan ambos sockets el puerto 80? Explique y justifique su respuesta.

SECCIÓN 3.4

- R9. En los protocolos rdt estudiados, ¿por qué necesitábamos introducir números de secuencia?
- R10. En los protocolos rdt estudiados, ¿por qué necesitábamos introducir temporizadores?
- R11. Suponga que el retardo de ida y vuelta entre el emisor y el receptor es constante y conocido por el emisor. ¿Se necesitaría en este caso un temporizador en el protocolo rdt 3.0, suponiendo que los paquetes pueden perderse? Explique su respuesta.
- R12. Visite el applet de Java *Go-Back-N* en el sitio web del libro.
- Haga que el emisor envíe cinco paquetes y luego detenga la animación antes de que cualquiera de los cinco paquetes alcance su destino. A continuación, elimine el primer paquete y reanude la animación. Describa lo que ocurre.
 - Repita el experimento, pero ahora deje que el primer paquete alcance su destino y elimine el primer paquete de reconocimiento. Describa lo que ocurre.
 - Para terminar, pruebe a enviar seis paquetes. ¿Qué ocurre?
- R13. Repita el problema R12, pero ahora utilizando el applet de Java con repetición selectiva (SR). ¿En qué se diferencian los protocolos SR y GBN?

SECCIÓN 3.5

- R14. ¿Verdadero o falso?
- El host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. Suponga que el host B no tiene datos que enviar al host A. El host B no enviará paquetes de reconocimiento al host A porque el host B no puede superponer esos reconocimientos sobre los datos.
 - El tamaño de la ventana de recepción de TCP `VentCongestion` nunca varía mientras dura la conexión.
 - Suponga que el host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. El número de bytes no reconocidos que A envía no puede exceder el tamaño del buffer del receptor.
 - Suponga que el host A está enviando al host B un archivo de gran tamaño a través de una conexión TCP. Si el número de secuencia de un segmento en esta conexión es m , entonces el número de secuencia del siguiente segmento necesariamente tiene que ser $m + 1$.
 - El segmento TCP contiene un campo en su cabecera para `VentRecepcion`.
 - Suponga que el último RTT muestra en una conexión TCP es igual a 1 segundo. El valor actual del `IntervaloFinTemporización` para la conexión será necesariamente ≥ 1 segundo.
 - Suponga que el host A envía al host B un segmento con el número de secuencia 38 y 4 bytes de datos a través de una conexión TCP. En este mismo segmento el número de reconocimiento necesariamente tiene que ser 42.

- R15. Suponga que el host A envía dos segmentos TCP seguidos al host B a través de una conexión TCP. El primer segmento tiene el número de secuencia 90 y el segundo tiene el número de secuencia 110.
- ¿Cuántos datos hay en el primer segmento?
 - Suponga que el primer segmento se pierde pero el segundo llega a B. En el paquete de reconocimiento que el host B envía al host A, ¿cuál será el número de reconocimiento?
- R16. Considere el ejemplo de la conexión Telnet de la Sección 3.5. Unos pocos segundos después de que el usuario escriba la letra ‘C’, escribe la letra ‘R’. Después de escribir la letra ‘R’, ¿cuántos segmentos se envían y qué valores se almacenan en los campos número de secuencia y número de reconocimiento de los segmentos?

SECCIÓN 3.7

- R17. Suponga que existen dos conexiones TCP en un cierto enlace de cuello de botella con una velocidad de R bps. Ambas conexiones tienen que enviar un archivo de gran tamaño (en la misma dirección a través del enlace de cuello de botella). Las transmisiones de los archivos se inician en el mismo instante. ¿Qué velocidad de transmisión podría proporcionar TCP a cada una de las conexiones?
- R18. ¿Verdadero o falso? En el control de congestión de TCP, si el temporizador del emisor caduca, el valor de $umbral_{AL}$ se hace igual a la mitad de su valor anterior.
- R19. En la exposición acerca de la división TCP del recuadro de la Sección 3.7, se establecía que el tiempo de respuesta con la división TCP es aproximadamente igual a $4 \cdot RTT_{FE} + RTT_{BE} +$ el tiempo de procesamiento. Justifique esta afirmación.

Problemas

- P1. Suponga que el cliente A inicia una sesión Telnet con el servidor S. Aproximadamente en el mismo instante, el cliente B también inicia una sesión Telnet con el servidor S. Proporcione los posibles números de puerto de origen y de destino para:
- Los segmentos enviados de A a S.
 - Los segmentos enviados de B a S.
 - Los segmentos enviados de S a A.
 - Los segmentos enviados de S a B.
 - Si A y B son hosts diferentes, ¿es posible que el número de puerto de origen en los segmentos que van de A a S sea el mismo que en los segmentos que van de B a S?
 - ¿Qué ocurre si A y B son el mismo host?
- P2. Considere la Figura 3.5. ¿Cuáles son los valores de los puertos de origen y de destino en los segmentos que fluyen desde el servidor de vuelta a los procesos cliente? ¿Cuáles son las direcciones IP de los datagramas de la capa de red que transportan los segmentos de la capa de transporte?
- P3. UDP y TCP utilizan el complemento a 1 para calcular sus sumas de comprobación. Suponga que tiene los tres bytes de 8 bits siguientes: 01010011, 01100110, 01110100. ¿Cuál es el complemento a 1 de la suma de estos bytes? (Observe que aunque UDP y TCP emplean palabras de 16 bits para calcular la suma de comprobación, en este problema le pedimos que considere sumas de 8 bits). Explique cómo funciona. ¿Por qué UDP utiliza el complemento a 1 de la suma; es decir, por qué no simplemente emplea la suma? Con el esquema del complemento a 1, ¿cómo detecta el receptor los errores? ¿Es posible que un error de un solo bit no sea detectado? ¿Qué ocurre si hay 2 bits erróneos?

- P4. a. Suponga que tiene los 2 bytes siguientes: 01011100 y 01100101. ¿Cuál es el complemento a 1 de la suma de estos 2 bytes?
- b. Suponga que tiene los 2 bytes siguientes: 11011010 y 01100101. ¿Cuál es el complemento a 1 de la suma de estos 2 bytes?
- c. Para los bytes del apartado (a), proporcione un ejemplo en el que un bit cambie de valor en cada uno de los 2 bytes y aún así el complemento a 1 no varíe.
- P5. Suponga que el receptor UDP calcula la suma de comprobación de Internet para el segmento UDP recibido y comprueba que se corresponde con el valor almacenado en el campo de suma de comprobación. ¿Puede el receptor estar completamente seguro de que no hay ningún bit erróneo? Explique su respuesta.
- P6. Recuerde el motivo de corregir el protocolo rdt 2.1. Demuestre que el receptor mostrado en la Figura 3.57 y el emisor mostrado en la Figura 3.11 pueden llegar a entrar en un estado de bloqueo tal que cada uno de ellos esté esperando a que se produzca un suceso que no ocurrirá nunca.
- P7. En el protocolo rdt 3.0, los paquetes ACK que fluyen del receptor al emisor no tienen números de secuencia (aunque tienen un campo ACK que contiene el número de secuencia del paquete que están reconociendo). ¿Por qué estos paquetes ACK no requieren números de secuencia?
- P8. Dibuje la máquina de estados finitos correspondiente al lado receptor del protocolo rdt 3.0.
- P9. Dibuje un esquema que muestre la operación del protocolo rdt 3.0 cuando los paquetes de datos y los paquetes de reconocimiento están corrompidos. Utilice un esquema similar al mostrado en la Figura 3.16.
- P10. Sea un canal que puede perder paquetes pero del que se conoce su retardo máximo. Modifique el protocolo rdt 2.1 para incluir los fines de temporización y las retransmisiones del emisor. Argumente de manera informal por qué su protocolo puede comunicarse correctamente a través de este canal.

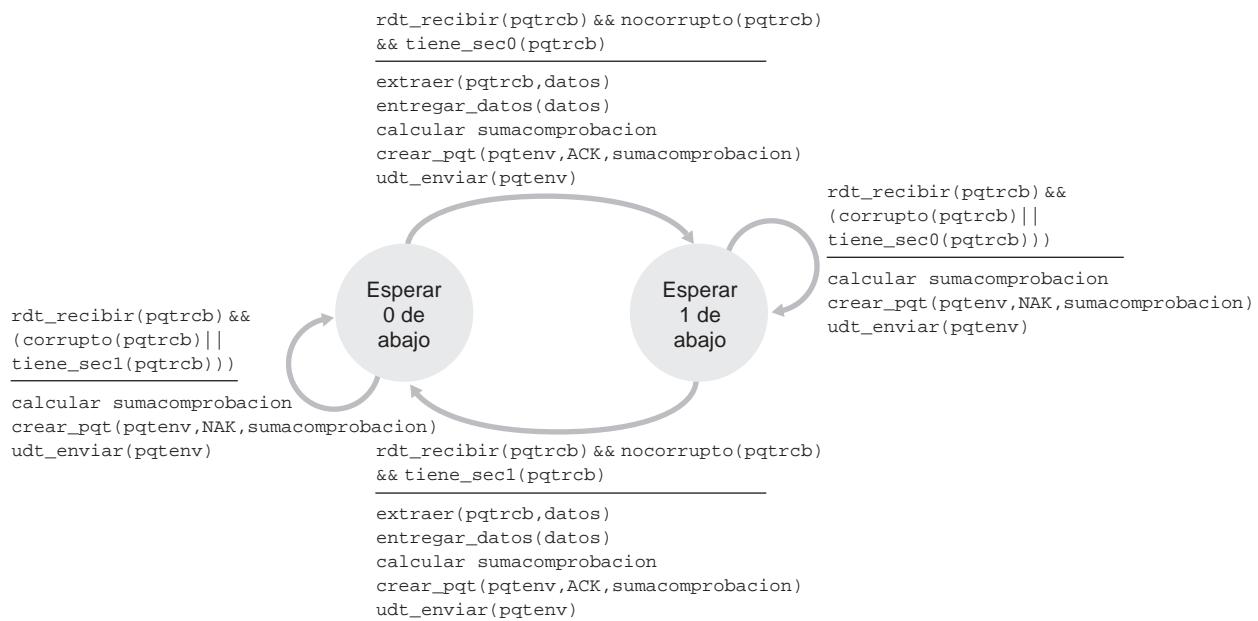


Figura 3.57 ♦ Un receptor incorrecto para el protocolo rdt 2.1.

- P11. Considere el receptor `rdt2.2` mostrado en la Figura 3.14 y la creación de un paquete nuevo en la transición a sí mismo (esto es, la transición del estado de vuelta hacia sí mismo) en los estados Esperar 0 de abajo y Esperar 1 de abajo: `pqtenv=crear_pqt(ACK, 1, sumacomprobacion)` y `pqtenv=crear_pqt(ACK, 0, sumacomprobacion)`. ¿Funcionaría correctamente el protocolo si se eliminara esta acción de la transición al mismo estado en el estado Esperar 1 de abajo? Justifique su respuesta. ¿Qué ocurriría si se elimina este suceso de la transición del estado Esperar 0 de abajo a sí mismo? [Sugerencia: en este último caso, considere lo que ocurriría si el primer paquete que va del emisor al receptor estuviera corrompido.]
- P12. El lado del emisor de `rdt3.0` simplemente ignora (es decir, no realiza ninguna acción) todos los paquetes recibidos que contienen un error o que presentan un valor erróneo en el campo número de reconocimiento (`acknum`) de un paquete de reconocimiento. Suponga que, en tales circunstancias, `rdt3.0` simplemente retransmite el paquete de datos actual. ¿Funcionaría en estas condiciones el protocolo? (Sugerencia: piense en lo que ocurriría si solo hubiera errores de bit; no se producen pérdidas de paquetes pero sí pueden ocurrir sucesos de fin prematuro de la temporización. Considere cuántas veces se envía el paquete n , cuando n tiende a infinito.)
- P13. Considere el protocolo `rdt 3.0`. Dibuje un diagrama que muestre que si la conexión de red entre el emisor y el receptor puede reordenar los mensajes (es decir, que dos mensajes que se propagan por el medio físico existente entre el emisor y el receptor pueden ser reordenados), entonces el protocolo de bit alternante no funcionará correctamente (asegúrese de identificar claramente el sentido en el que no funcionará correctamente). En el diagrama debe colocar el emisor a la izquierda y el receptor a la derecha, con el eje de tiempos en la parte inferior de la página y deberá mostrar el intercambio de los mensajes de datos (D) y de reconocimiento (A). No olvide indicar el número de secuencia asociado con cualquier segmento de datos o de reconocimiento.
- P14. Considere un protocolo de transferencia de datos fiable que solo utiliza paquetes de reconocimiento negativo. Imagine que el emisor envía datos con muy poca frecuencia. ¿Sería preferible un protocolo con solo emplea paquetes NAK a uno que utilice paquetes ACK? ¿Por qué? Suponga ahora que el emisor tiene muchos datos que transmitir y que la conexión terminal a terminal experimenta muy pocas pérdidas. En este segundo caso, ¿sería preferible un protocolo que solo emplee paquetes NAK a otro que utilice paquetes ACK? ¿Por qué?
- P15. Considere el ejemplo mostrado en la Figura 3.17. ¿Cuál tiene que ser el tamaño de la ventana para que la tasa de utilización del canal sea mayor del 98 por ciento? Suponga que el tamaño de un paquete es de 1.500 bytes, incluyendo tanto los campos de cabecera como los datos.
- P16. Suponga que una aplicación utiliza el protocolo `rdt 3.0` como su protocolo de la capa de transporte. Como el protocolo de parada y espera tiene una tasa de utilización del canal muy baja (como se ha demostrado en el ejemplo de conexión que atraviesa el país de costa a costa), los diseñadores de esta aplicación permiten al receptor devolver una serie (más de dos) de ACK 0 y ACK 1 alternantes incluso si los correspondientes datos no han llegado al receptor. ¿Debería este diseño de aplicación aumentar la tasa de utilización del canal? ¿Por qué? ¿Existe algún problema potencial con esta técnica? Explique su respuesta.
- P17. Suponga que tenemos dos entidades de red, A y B, que están conectadas mediante un canal bidireccional perfecto (es decir, cualquier mensaje enviado será recibido correctamente; el canal no corromperá, no perderá y no reordenará los paquetes). A y B se entregan mensajes de datos entre sí de manera alternante: en primer lugar, A debe entregar un mensaje a B, después B entrega un mensaje a A; a continuación, A entregará un mensaje a B, y así sucesivamente. Si una entidad se encuentra en un estado en el que no debería enviar un mensaje al otro lado y se produce un suceso como `rdt_enviar(datos)` procedente de arriba puede simplemente ignorarlo con una llamada a `rdt_inhabilitar_enviar(datos)`, que informa a la capa superior de que actualmente no es posible enviar datos. [Nota: esta simplificación se hace para no tener que preocuparse por el almacenamiento de datos en buffer].

Diseñe una especificación de la máquina de estados finitos (FSM) para este protocolo (una máquina FSM para A y una máquina FSM para B). Fíjese en que en este caso no tiene que preocuparse por el mecanismo de fiabilidad; el punto clave de esta cuestión es crear la especificación de una máquina FSM que refleje el comportamiento sincronizado de las dos entidades. Deberá emplear los siguientes sucesos y acciones, los cuales tienen el mismo significado que en el protocolo rdt1.0 de la Figura 3.9: `rdt_enviar(datos)`, `pqt=crear_pqt(datos)`, `udt_enviar(pqt)`, `rdt_recibir(pqt)`, `extraer (pqt,datos)`, `entregar_datos(datos)`. Asegúrese de que el protocolo refleja la alternancia estricta de envío entre A y B. Asegúrese también de indicar los estados iniciales para A y B en sus descripciones de la FSM.

- P18. En el protocolo SR genérico que hemos estudiado en la Sección 3.4.4, el emisor transmite un mensaje tan pronto como está disponible (si se encuentra dentro de la ventana) sin esperar a recibir un paquete de reconocimiento. Suponga ahora que deseamos disponer de un protocolo SR que envíe mensajes de dos en dos. Es decir, el emisor enviará una pareja de mensajes y enviará la siguiente pareja de mensajes solo cuando sepa que los dos mensajes de la primera pareja se han recibido correctamente.

Suponga que el canal puede perder mensajes pero no corromperlos ni tampoco reordenarlos. Diseñe un protocolo de control de errores para un servicio de transferencia de mensajes fiable y unidireccional. Proporcione una descripción de las máquinas de estados finitos del emisor y del receptor. Describa el formato de los paquetes intercambiados por el emisor y el receptor, y viceversa. Si utiliza alguna llamada a procedimiento distinta de las empleadas en la Sección 3.4 (por ejemplo, `udt_enviar()`, `iniciar_temporizador()`, `rdt_recibir()`, etc.), defina claramente las acciones que realizan. Proporcione un ejemplo (una gráfica temporal del emisor y del receptor) que muestre cómo este protocolo se recupera de la pérdida de paquetes.

- P19. Considere un escenario en el que el host A desea enviar simultáneamente paquetes a los hosts B y C. El host A está conectado a B y C a través de un canal de multidifusión (*broadcast*) (un paquete enviado por A es transportado por el canal tanto a B como a C). Suponga que el canal de multidifusión que conecta A, B y C puede perder y corromper de manera independiente los paquetes (es decir, puede ocurrir, por ejemplo, que un paquete enviado desde A llegue correctamente a B, pero no a C). Diseñe un protocolo de control de errores similar a un protocolo de parada y espera que permita transferir paquetes de forma fiable de A a B y C, de manera que A no obtendrá nuevos datos de la capa superior hasta que sepa que tanto B como C han recibido correctamente el paquete actual. Proporcione las descripciones de las máquinas de estados finitos de A y C. (*Sugerencia:* la FSM de B será prácticamente la misma que la de C.) Proporcione también una descripción del formato o formatos de paquete utilizados.

- P20. Considere un escenario en el que el host A y el host B desean enviar mensajes al host C. Los hosts A y C están conectados mediante un canal que puede perder y corromper (pero no reordenar) los mensajes. Los hosts B y C están conectados a través de otro canal (independiente del canal que conecta a A y C) que tiene las mismas propiedades. La capa de transporte del host C tiene que alternar la entrega de los mensajes que A y B tienen que pasar a la capa superior (es decir, primero entrega los datos de un paquete de A y luego los datos de un paquete de B, y así sucesivamente). Diseñe un protocolo de control de errores de tipo parada y espera para transferir de forma fiable los paquetes de A y B a C, con una entrega alternante en el host C, como hemos descrito anteriormente. Proporcione las descripciones de las FSM de A y C. (*Sugerencia:* la FSM de B será prácticamente la misma que la de A.) Proporcione también una descripción del formato o formatos de paquete utilizados.

- P21. Suponga que tenemos dos entidades de red, A y B. B tiene que enviar a A un conjunto de mensajes de datos, cumpliendo los siguientes convenios. Cuando A recibe una solicitud de la capa superior para obtener el siguiente mensaje de datos (D) de B, A tiene que enviar un mensaje de solicitud (R) a B a través del canal que va de A a B. Solo cuando B recibe

un mensaje R puede devolver un mensaje de datos (D) a A a través del canal de B a A. A tiene que entregar exactamente una copia de cada mensaje D a la capa superior. Los mensajes R se pueden perder (pero no corromper) en el canal de A a B; los mensajes D, una vez enviados, siempre son correctamente entregados. El retardo a lo largo de ambos canales es desconocido y variable.

Diseñe (proporcione una descripción de la FSM de) un protocolo que incorpore los mecanismos apropiados para compensar las pérdidas del canal de A a B e implemente el paso de los mensajes a la capa superior de la entidad A, como se ha explicado anteriormente. Utilice solo aquellos mecanismos que sean absolutamente necesarios.

- P22. Sea un protocolo GBN con un tamaño de ventana de emisor de 4 y un rango de números de secuencia de 1.024. Suponga que en el instante t el siguiente paquete en orden que el receptor está esperando tiene el número de secuencia k . Suponga que el medio de transmisión no reordena los mensajes. Responda a las siguientes cuestiones:
- ¿Cuáles son los posibles conjuntos de números de secuencia que pueden estar dentro de la ventana del emisor en el instante t ? Justifique su respuesta.
 - ¿Cuáles son todos los valores posibles del campo ACK en todos los posibles mensajes que están actualmente propagándose de vuelta al emisor en el instante t ? Justifique su respuesta.
- P23. Considere los protocolos GBN y SR. Suponga que el tamaño del espacio de números de secuencia es k . ¿Cuál es la máxima ventana de emisor permitida que evitará la ocurrencia de problemas como los indicados en la Figura 3.27 para cada uno de estos protocolos?
- P24. Responda verdadero o falso a las siguientes preguntas y justifique brevemente sus respuestas:
- Con el protocolo SR, el emisor puede recibir un ACK para un paquete que se encuentra fuera de su ventana actual.
 - Con GBN, el emisor puede recibir un ACK para un paquete que se encuentra fuera de su ventana actual.
 - El protocolo de bit alternante es igual que el protocolo SR pero con un tamaño de ventana en el emisor y en el receptor igual a 1.
 - El protocolo de bit alternante es igual que el protocolo GBN pero con un tamaño de ventana en el emisor y en el receptor igual a 1.
- P25. Hemos dicho que una aplicación puede elegir UDP como protocolo de transporte porque UDP ofrece a la aplicación un mayor grado de control (que TCP) en lo relativo a qué datos se envían en un segmento y cuándo.
- ¿Por qué una aplicación tiene más control sobre qué datos se envían en un segmento?
 - ¿Por qué una aplicación tiene más control sobre cuándo se envía el segmento?
- P26. Se desea transferir un archivo de gran tamaño de L bytes del host A al host B. Suponga un MSS de 536 bytes.
- ¿Cuál es el valor máximo de L tal que los números de secuencia de TCP no se agoten? Recuerde que el campo número de secuencia de TCP tiene 4 bytes.
 - Para el valor de L que haya obtenido en el apartado (a), calcule el tiempo que tarda en transmitirse el archivo. Suponga que a cada segmento se añade un total de 66 bytes para la cabecera de la capa de transporte, de red y de enlace de datos antes de enviar el paquete resultante a través de un enlace a 155 Mbps. Ignore el control de flujo y el control de congestión de modo que A pueda bombar los segmentos seguidos y de forma continuada.
- P27. Los hosts A y B están comunicándose a través de una conexión TCP y el host B ya ha recibido de A todos los bytes hasta el byte 126. Suponga que a continuación el host A envía dos segmentos seguidos al host B. El primer y el segundo segmentos contienen, respectivamente, 80 y 40 bytes de datos. En el primer segmento, el número de secuencia es 127, el número del

puerto de origen es 302 y el número de puerto de destino es 80. El host B envía un paquete de reconocimiento cuando recibe un segmento del host A.

- a. En el segundo segmento enviado del host A al B, ¿cuáles son el número de secuencia, el número del puerto de origen y el número del puerto de destino?
 - b. Si el primer segmento llega antes que el segundo segmento, ¿cuál es el número de reconocimiento, el número del puerto de origen y el número del puerto de destino en el ACK correspondiente al primer segmento?
 - c. Si el segundo segmento llega antes que el primero, ¿cuál es el número de reconocimiento en el ACK correspondiente a este segmento?
 - d. Suponga que los dos segmentos enviados por A llegan en orden a B. El primer paquete de reconocimiento se pierde y el segundo llega después de transcurrido el primer intervalo de fin de temporización. Dibuje un diagrama de temporización que muestre estos segmentos y todos los restantes segmentos y paquetes de reconocimiento enviados. (Suponga que no se producen pérdidas de paquetes adicionales.) Para cada uno de los segmentos que incluya en su diagrama, especifique el número de secuencia y el número de bytes de datos; para cada uno de los paquetes de reconocimiento que añada, proporcione el número de reconocimiento.
- P28. Los hosts A y B están directamente conectados mediante un enlace a 100 Mbps. Existe una conexión TCP entre los dos hosts y el host A está enviando al host B un archivo de gran tamaño a través de esta conexión. El host A puede enviar sus datos de la capa de aplicación a su socket TCP a una velocidad tan alta como 120 Mbps pero el host B solo puede leer los datos almacenados en su buffer de recepción TCP a una velocidad máxima de 50 Mbps. Describa el efecto del control de flujo de TCP.
- P29. En la Sección 3.5.6 se han estudiado las cookies SYN.
- a. ¿Por qué es necesario que el servidor utilice un número de secuencia inicial especial en SYNACK?
 - b. Suponga que un atacante sabe que un host objetivo utiliza cookies SYN. ¿Puede el atacante crear conexiones semi-abiertas o completamente abiertas enviando simplemente un paquete ACK al host objetivo? ¿Por qué?
 - c. Suponga que un atacante recopila una gran cantidad de números de secuencia iniciales enviados por el servidor. ¿Puede el atacante hacer que el servidor cree muchas conexiones completamente abiertas enviando paquetes ACK con esos números de secuencia iniciales? ¿Por qué?
- P30. Considere la red mostrada en el escenario 2 de la Sección 3.6.1. Suponga que ambos hosts emisores A y B tienen definidos valores de fin de temporización fijos.
- a. Demuestre que aumentar el tamaño del buffer finito del router puede llegar a hacer que se reduzca la tasa de transferencia (λ_{out}).
 - b. Suponga ahora que ambos hosts ajustan dinámicamente su valores de fin de temporización (como lo hace TCP) basándose en el retardo del buffer del router. ¿Incrementar el tamaño del buffer ayudaría a incrementar la tasa de transferencia? ¿Por qué?
- P31. Suponga que los cinco valores de RTTMuestra medidos (véase la Sección 3.5.3) son 106 ms, 120 ms, 140 ms, 90 ms y 115 ms. Calcule el valor de RTTEstimado después de obtener cada uno de estos valores de RTTMuestra, utilizando un valor de $\alpha = 0,125$ y suponiendo que el valor de RTTEstimado era 100 ms justo antes de obtener el primero de estas cinco muestras. Calcule también el valor de RTTDesv después de obtener cada muestra, suponiendo un valor de $\beta = 0,25$ y que el valor de RTTDesv era de 5 ms justo antes de obtener la primera de estas cinco muestras. Por último, calcule el IntervaloFinTemporización de TCP después de obtener cada una de estas muestras.

- P32. Considere el procedimiento de TCP para estimar RTT. Suponga que $\alpha = 0,1$. Sea $RTTMuestra_1$ la muestra de RTT más reciente, $RTTMuestra_2$ la siguiente muestra de RTT más reciente, y así sucesivamente.
- Para una conexión TCP determinada, suponga que han sido devueltos cuatro paquetes de reconocimiento con las correspondientes muestras de RTT, $RTTMuestra_4$, $RTTMuestra_3$, $RTTMuestra_2$ y $RTTMuestra_1$. Exprese $RTTESTimado$ en función de las cuatro muestras de RTT.
 - Generalize la fórmula para n muestras de RTT.
 - En la fórmula del apartado (b), considere que n tiende a infinito. Explique por qué este procedimiento de cálculo del promedio se conoce como media móvil exponencial.
- P33. En la Sección 3.5.3, se ha estudiado la estimación de RTT en TCP. ¿Por qué cree que TCP evita medir $RTTMuestra$ para los segmentos retransmitidos?
- P34. ¿Cuál es la relación entre la variable `EnviarBase` de la Sección 3.5.4 y la variable `UltimoByteRecibido` de la Sección 3.5.5?
- P35. ¿Cuál es la relación entre la variable `UltimoByteRecibido` de la Sección 3.5.5 y la variable `y` de la Sección 3.5.4?
- P36. En la Sección 3.5.4 hemos visto que TCP espera hasta que ha recibido tres ACK duplicados antes de realizar una retransmisión rápida. ¿Por qué cree que los diseñadores de TCP han decidido no realizar una retransmisión rápida después de recibir el primer ACK duplicado correspondiente a un segmento?
- P37. Compare GBN, SR y TCP (sin paquetes ACK retardados). Suponga que los valores de fin de temporización de los tres protocolos son los suficientemente grandes como para que 5 segmentos de datos consecutivos y sus correspondientes ACK puedan ser recibidos (si no se producen pérdidas en el canal) por el host receptor (host B) y el host emisor (host A), respectivamente. Suponga que el host A envía 5 segmentos de datos al host B y que el segundo segmento (enviado desde A) se pierde. Al final, los 5 segmentos de datos han sido recibidos correctamente por el host B.
- ¿Cuántos segmentos ha enviado en total el host A y cuántos ACK ha enviado en total el host B? ¿Cuáles son sus números de secuencia? Responda a esta pregunta para los tres protocolos.
 - Si los valores de fin de temporización para los tres protocolos son mucho mayores que 5 RTT, ¿qué protocolo entregará correctamente los cinco segmentos de datos en el menor intervalo de tiempo?
- P38. En la descripción de TCP de la Figura 3.53, el valor del umbral, $umbralAL$, se define como $umbral=VentCongestion/2$ en varios sitios y el valor de $umbralAL$ se hace igual a la mitad del tamaño de la ventana cuando se produce un suceso de pérdida. ¿Tiene que ser la velocidad a la que el emisor está transmitiendo cuando se produce un suceso de pérdida aproximadamente igual a $VentCongestion$ segmentos por RTT? Explique su respuesta. Si su respuesta es no, ¿puede sugerir una forma diferente en la que se podría fijar el valor de $umbralAL$?
- P39. Considere la Figura 3.46(b). Si λ'_{in} aumenta por encima de $R/2$, ¿puede λ_{out} incrementarse por encima de $R/3$? Explique su respuesta. Considere ahora la Figura 3.46(c). Si λ'_{in} aumenta por encima de $R/2$, ¿puede λ_{out} aumentar por encima de $R/4$ suponiendo que un paquete será reenviado dos veces como media desde el router al receptor? Explique su respuesta.
- P40. Considere la Figura 3.58. Suponiendo que TCP Reno es el protocolo que presenta el comportamiento mostrado en la figura, responda a las siguientes preguntas. En todos los casos, deberá proporcionar una breve explicación que justifique su respuesta.
- Identifique los intervalos de tiempo cuando TCP está operando en modo de arranque lento.



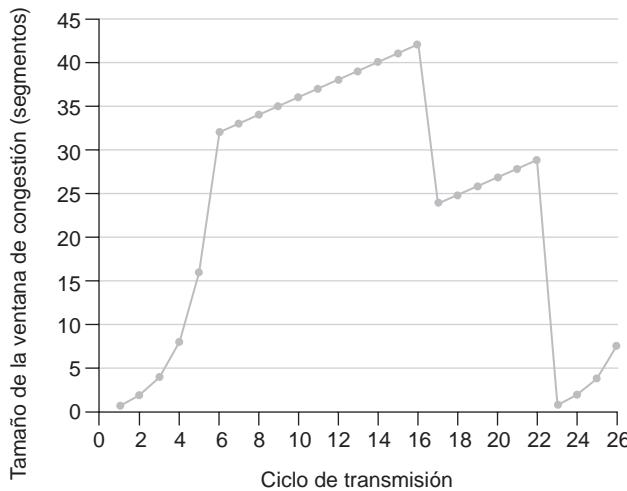


Figura 3.58 ♦ Tamaño de ventana de TCP en función del tiempo.

- b. Identifique los intervalos de tiempo cuando TCP está operando en el modo de evitación de la congestión.
 - c. Despues del ciclo de transmisión 16, ¿se detecta la pérdida de segmento mediante tres ACK duplicados o mediante un fin de temporización?
 - d. Despues del ciclo de transmisión 22, ¿se detecta la pérdida de segmento mediante tres ACK duplicados o mediante un fin de temporización?
 - e. ¿Cuál es el valor inicial de $\text{umbral}_{\text{AL}}$ en el primer ciclo de transmisión?
 - f. ¿Cuál es el valor de $\text{umbral}_{\text{AL}}$ transcurridos 18 ciclos de transmisión?
 - g. ¿Cuál es el valor de $\text{umbral}_{\text{AL}}$ transcurridos 24 ciclos de transmisión?
 - h. ¿Durante cuál ciclo de transmisión se envía el segmento 70?
 - i. Suponiendo que se detecta una pérdida de paquete despues del ciclo de transmisión 26 a causa de la recepción de un triple ACK duplicado, ¿cuáles serán los valores del tamaño de la ventana de congestión y de $\text{umbral}_{\text{AL}}$?
 - j. Suponga que se utiliza TCP Tahoe (en lugar de TCP Reno) y que se han recibido triples ACK duplicados en el ciclo de transmisión 16. ¿Cuáles serán los valores del tamaño de la ventana de congestión y de $\text{umbral}_{\text{AL}}$ en el ciclo de transmisión 19?
 - k. Suponga otra vez que se utiliza TCP Tahoe y que se produce un suceso de fin de temporización en el ciclo de transmisión 22. ¿Cuántos paquetes han sido enviados entre los ciclos de transmisión 17 a 22, ambos inclusive?
- P41. Utilice la Figura 3.55, que ilustra la convergencia del algoritmo AIMD de TCP. Suponga que en lugar de un decrecimiento multiplicativo, TCP disminuye el tamaño de la ventana en una cantidad constante. ¿Convergería el algoritmo AIAD resultante hacia un algoritmo de cuota equitativa? Justifique su respuesta utilizando un diagrama similar al de la Figura 3.55.
- P42. En la Sección 3.5.4, hemos explicado que el intervalo de fin de temporización se duplica despues de un suceso de fin de temporización. Este mecanismo es una forma de control de congestión. ¿Por qué TCP necesita un mecanismo de control de congestión basado en ventana (como hemos estudiado en la Sección 3.7) además de un mecanismo de duplicación del intervalo de fin de temporización?
- P43. El host A está enviando un archivo de gran tamaño al host B a través de una conexión TCP. En esta conexión nunca se pierden paquetes y los temporizadores nunca caducan. La velocidad

de transmisión del enlace que conecta el host A con Internet es R bps. Suponga que el proceso del host A es capaz de enviar datos a su socket TCP a una velocidad de S bps, donde $S = 10 \cdot R$. Suponga también que el buffer de recepción de TCP es lo suficientemente grande como para almacenar el archivo completo y que el buffer emisor solo puede almacenar un porcentaje del archivo. ¿Qué impide al proceso del host A pasar datos de forma continua a su socket TCP a una velocidad de S bps? ¿El mecanismo de control de flujo de TCP, el mecanismo de control de congestión de TCP o alguna otra cosa? Razone su respuesta.

- P44. Se envía un archivo de gran tamaño de un host a otro a través de una conexión TCP sin pérdidas.
- Suponga que TCP utiliza el algoritmo AIMD para su control de congestión sin fase de arranque lento. Suponiendo que `VentCongestion` aumenta 1 MSS cada vez que se recibe un lote de paquetes ACK y suponiendo que los intervalos RTT son aproximadamente constantes, ¿Cuánto tiempo tarda `VentCongestion` en aumentar de 6 MSS a 12 MSS (si no se producen sucesos de pérdida de paquetes)?
 - ¿Cuál es la tasa de transferencia media (en función de MSS y RTT) para esta conexión hasta llegar al periodo RTT número 6?
- P45. Recuerde la descripción macroscópica de la tasa de transferencia de TCP. En el periodo de tiempo que va desde que la velocidad de la conexión varía entre $W/(2 \cdot RTT)$ y W/RTT , solo se pierde un paquete (justo al final del periodo).
- Demuestre que la tasa de pérdidas (fracción de paquetes perdidos) es igual a:

$$L = \text{tasa de pérdidas} = \frac{1}{\frac{3}{8}W^2 + \frac{3}{4}W}$$

- Utilice el resultado anterior para demostrar que si una conexión tiene una tasa de pérdidas igual a L , entonces su tasa promedio es aproximadamente igual a

$$\approx \frac{1,22 \cdot MSS}{RTT \sqrt{L}}$$

- P46. Considere una única conexión TCP (Reno) que emplea un enlace a 10Mbps que no almacena en buffer ningún dato. Suponga que este enlace es el único enlace congestionado entre los hosts emisor y receptor. Suponga también que el emisor TCP tiene que enviar al receptor un archivo de gran tamaño y que el buffer de recepción del receptor es mucho más grande que la ventana de congestión. Haremos además las siguientes suposiciones: el tamaño de segmento TCP es de 1.500 bytes, el retardo de propagación de ida y vuelta de esta conexión es igual a 150 milisegundos y esta conexión TCP siempre se encuentra en la fase de evitación de la congestión, es decir, ignoramos la fase de arranque lento.
- ¿Cuál es el tamaño máximo de ventana (en segmentos) que esta conexión TCP puede alcanzar?
 - ¿Cuáles son el tamaño medio de ventana (en segmentos) y la tasa de transferencia media (en bps) de esta conexión TCP?
 - ¿Cuánto tiempo tarda esta conexión TCP en alcanzar de nuevo su tamaño de ventana máximo después de recuperarse de una pérdida de paquete?
- P47. Continuando con el escenario descrito en el problema anterior, suponga que el enlace a 10Mbps puede almacenar en buffer un número finito de segmentos. Razone por qué para que el enlace esté siempre ocupado enviando datos, deberíamos seleccionar un tamaño de buffer que sea al menos igual al producto de la velocidad del enlace C y el retardo de propagación de ida y vuelta entre el emisor y el receptor.

- P48. Repita el Problema 46, pero sustituyendo el enlace a 10 Mbps por un enlace a 10 Gbps. Observe que en la respuesta al apartado (c) habrá demostrado que se tarda mucho tiempo en que el tamaño de la ventana de congestión alcance su máximo después de recuperarse de una pérdida de paquete. Diseñe una solución que resuelva este problema.
- P49. Sea T (medido en RTT) el intervalo de tiempo que una conexión TCP tarda en aumentar el tamaño de su ventana de congestión de $W/2$ a W , donde W es el tamaño máximo de la ventana de congestión. Demuestre que T es una función de la tasa de transferencia media de TCP.
- P50. Considere un algoritmo AIMD de TCP simplificado en el que el tamaño de la ventana de congestión se mide en número de segmentos, no en bytes. En la fase de incremento aditivo, el tamaño de la ventana de congestión se incrementa en un segmento cada RTT. En la fase de decrecimiento multiplicativo, el tamaño de la ventana de congestión se reduce a la mitad (si el resultado no es un entero, redondee al entero más próximo). Suponga que dos conexiones TCP, C_1 y C_2 , comparten un enlace congestionado cuya velocidad es de 30 segmentos por segundo. Suponemos que tanto C_1 como C_2 están en la fase de evitación de la congestión. El intervalo RTT de la conexión C_1 es igual a 50 milisegundos y el de la conexión C_2 es igual a 100 milisegundos. Suponemos que cuando la velocidad de los datos en el enlace excede la velocidad del enlace, todas las conexiones TCP experimentan pérdidas de segmentos de datos.
- Si en el instante t_0 el tamaño de la ventana de congestión de ambas conexiones, C_1 y C_2 , es de 10 segmentos, ¿cuáles serán los tamaños de dichas ventanas de congestión después de transcurridos 1.000 milisegundos?
 - ¿Obtendrán estas dos conexiones, a largo plazo, la misma cuota de ancho de banda del enlace congestionado? Explique su respuesta.
- P51. Continúe con la red descrita en el problema anterior, pero ahora suponga que las dos conexiones TCP, C_1 y C_2 , tienen el mismo intervalo RTT de 100 milisegundos. Supongamos que en el instante t_0 , el tamaño de la ventana de congestión de C_1 es de 15 segmentos pero el tamaño de la ventana de congestión de C_2 es igual a 10 segmentos.
- ¿Cuáles serán los tamaños de las ventanas de congestión después de transcurridos 2.200 ms?
 - ¿Obtendrán estas dos conexiones, a largo plazo, la misma cuota de ancho de banda del enlace congestionado?
 - Decimos que dos conexiones están sincronizadas si ambas conexiones alcanzan su tamaño de ventana máximo al mismo tiempo y alcanzan su tamaño mínimo de ventana también al mismo tiempo. ¿Terminarán con el tiempo sincronizándose estas dos conexiones? En caso afirmativo, ¿cuáles son sus tamaños máximos de ventana?
 - ¿Ayudará esta sincronización a mejorar la tasa de utilización del enlace compartido? ¿Por qué? Esboce alguna idea para evitar esta sincronización.
- P52. Veamos una modificación del algoritmo de control de congestión de TCP. En lugar de utilizar un incremento aditivo podemos emplear un incremento multiplicativo. Un emisor TCP incrementa su tamaño de ventana según una constante pequeña positiva a ($0 < a < 1$) cuando recibe un ACK válido. Halle la relación funcional existente entre la tasa de pérdidas L y el tamaño máximo de la ventana de congestión W . Demuestre que para esta conexión TCP modificada, independientemente de la tasa media de transferencia de TCP, una conexión TCP siempre invierte la misma cantidad de tiempo en incrementar el tamaño de su ventana de congestión de $W/2$ a W .
- P53. En nuestra exposición sobre el futuro de TCP de la Sección 3.7 hemos destacado que para alcanzar una tasa de transferencia de 10 Gbps, TCP solo podría tolerar una probabilidad de pérdida de segmentos de $2 \cdot 10^{-10}$ (o lo que es equivalente, un suceso de pérdida por cada 5.000.000.000 segmentos). Indique de dónde se obtienen los valores $2 \cdot 10^{-10}$ (1 por cada 5.000.000) para los valores de RTT y MSS dados en la Sección 3.7. Si TCP tuviera que dar soporte a una conexión a 100 Gbps, ¿qué tasa de pérdidas sería tolerable?

- P54. En nuestra exposición sobre el control de congestión de TCP de la Sección 3.7, implícitamente hemos supuesto que el emisor TCP siempre tiene datos que enviar. Consideremos ahora el caso en que el emisor TCP envía una gran cantidad de datos y luego en el instante t_1 se queda inactivo (puesto que no tiene más datos que enviar). TCP permanece inactivo durante un periodo de tiempo relativamente largo y en el instante t_2 quiere enviar más datos. ¿Cuáles son las ventajas y las desventajas de que TCP tenga que utilizar los valores de `VentCongestion` y `umbralAL` de t_1 cuando comienza a enviar datos en el instante t_2 ? ¿Qué alternativa recomendaría? ¿Por qué?
- P55. En este problema vamos a investigar si UDP o TCP proporcionan un cierto grado de autenticación del punto terminal.
- Considera un servidor que recibe una solicitud dentro de un paquete UDP y responde a la misma dentro de un paquete UDP (por ejemplo, como en el caso de un servidor DNS). Si un cliente con la dirección IP X suplanta su dirección con la dirección Y, ¿a dónde enviará el servidor su respuesta?
 - Suponga que un servidor recibe un SYN con la dirección IP de origen Y, y después de responder con un SYNACK, recibe un ACK con la dirección IP de origen Y y con el número de reconocimiento correcto. Suponiendo que el servidor elige un número de secuencia inicial aleatorio y que no existe ningún atacante interpuerto (*man-in-the-middle*), ¿puede el servidor estar seguro de que el cliente está en la dirección Y (y no en alguna otra dirección X que esté intentando suplantar a Y)?
- P56. En este problema, vamos a considerar el retardo introducido por la fase de arranque lento de TCP. Se tiene un cliente y un servidor web directamente conectados mediante un enlace a velocidad R . Suponga que el cliente desea extraer un objeto cuyo tamaño es exactamente igual a $15 S$, donde S es el tamaño máximo de segmento (MSS). Sea RTT el tiempo de transmisión de ida y vuelta entre el cliente y el servidor (suponemos que es constante). Ignorando las cabeceras del protocolo, determine el tiempo necesario para recuperar el objeto (incluyendo el tiempo de establecimiento de la conexión TCP) si
- $4 S/R > S/R + RTT > 2 S/R$
 - $S/R + RTT > 4 S/R$
 - $S/R > RTT$

Tareas de programación

Implementación de un protocolo de transporte fiable

En esta tarea de programación tendrá que escribir el código para la capa de transporte del emisor y el receptor con el fin de implementar un protocolo de transferencia de datos fiable simple. Hay disponibles dos versiones de esta práctica de laboratorio: la versión del protocolo de bit alternante y la versión del protocolo GBN. Esta práctica de laboratorio le resultará entretenida y su implementación diferirá muy poco de lo que se necesita en una situación real.

Puesto que probablemente no dispone de máquinas autónomas (con un sistema operativo que pueda modificar), su código tendrá que ejecutarse en un entorno simulado hardware/software. Sin embargo, la interfaz de programación proporcionada a sus rutinas (el código que efectuará las llamadas a sus entidades desde las capas superior e inferior) es muy similar a la que se utiliza en un entorno UNIX real. (De hecho, las interfaces software descritas en esta tarea de programación son mucho más realistas que los emisores y receptores con bucles infinitos que se describen en muchos textos.) También se simula el arranque y la detención de temporizadores, y las interrupciones de los temporizadores harán que se active su rutina de tratamiento de temporizadores.

La tarea completa de laboratorio, así como el código que tendrá que compilar con su propio código está disponible en el sitio web del libro en www.pearsonhighered.com/cs-resources.

Práctica de laboratorio con Wireshark: exploración de TCP

En esta práctica de laboratorio tendrá que utilizar su navegador web para acceder a un archivo almacenado en un servidor web. Como en las prácticas de laboratorio con Wireshark anteriores, tendrá que utilizar Wireshark para capturar los paquetes que lleguen a su computadora. A diferencia de las prácticas anteriores, *también* podrá descargar una traza de paquetes que Wireshark puede leer del servidor web del que haya descargado el archivo. En esta traza del servidor encontrará los paquetes que fueron generados a causa de su propio acceso al servidor web. Analizará las trazas del lado del cliente y de lado del servidor para explorar los aspectos de TCP. En particular, tendrá que evaluar el rendimiento de la conexión TCP entre su computadora y el servidor web. Tendrá que trazar el comportamiento de la ventana de TCP e inferirá la pérdida de paquetes, las retransmisiones, el comportamiento del control de flujo y del control de congestión y el tiempo de ida y vuelta estimado.

Como con el resto de las prácticas de laboratorio con Wireshark, la descripción completa de esta práctica está disponible en el sitio web del libro en www.pearsonhighered.com/cs-resources.

Práctica de laboratorio con Wireshark: exploración de UDP

En esta corta práctica de laboratorio, realizará una captura y un análisis de paquetes de su aplicación favorita que utilice UDP (por ejemplo, DNS o una aplicación multimedia como Skype). Como hemos visto en la Sección 3.3, UDP es un protocolo de transporte simple. En esta práctica de laboratorio tendrá que investigar los campos de cabecera del segmento UDP, así como el cálculo de la suma de comprobación.

Como con el resto de las prácticas de laboratorio con Wireshark, la descripción completa de esta práctica está disponible en el sitio web del libro en www.pearsonhighered.com/cs-resources.

UNA ENTREVISTA CON...

Van Jacobson

Van Jacobson trabaja actualmente en Google y fue previamente investigador en PARC. Antes de eso, fue co-fundador y Director Científico de Packet Design. Y antes aun, fue Director Científico en Cisco. Antes de unirse a Cisco, dirigió el Grupo de Investigación de Redes en el Lawrence Berkeley National Laboratory y dio clases en las universidades de Berkeley y Stanford en California. Van recibió el premio SIGCOMM de la ACM en 2001 por su sobresaliente contribución profesional al campo de las redes de comunicaciones, así como el premio Kobayashi del IEEE en 2002 por "contribuir a la comprensión de la congestión de red y a desarrollar mecanismos de control de congestión que han permitido la exitosa expansión de Internet". Fue elegido como miembro de la Academia Nacional de Ingeniería de los Estados Unidos en 2004.



Describa uno o dos de los proyectos más interesantes en los que haya trabajado durante su carrera. ¿Cuáles fueron los principales desafíos?

La escuela nos enseña muchas formas de encontrar respuestas. En todos los problemas interesantes en los que he trabajado, el desafío consistía en encontrar la pregunta correcta. Cuando Mike Karels y yo empezamos a analizar el tema de la congestión TCP, nos pasamos meses observando trazas de protocolos y paquetes, preguntándonos “¿Por qué está fallando?”. Un día, en la oficina de Mike, uno de nosotros dijo “La razón por la que no consigo ver por qué falla, es porque ni siquiera comprendo por qué llegó a funcionar”. Resultó que esa era la pregunta correcta, y nos obligó a inventar el “reloj ack” que hace que TCP funcione. Despues de eso, el resto fue sencillo.

Hablando en términos más generales, ¿cuál cree que es el futuro de las redes y de Internet?

Para la mayoría de la gente, Internet es la Web. Los expertos en redes sonríen educadamente, porque sabemos que la Web no es otra cosa que una aplicación que se ejecuta sobre Internet, pero... ¿y si la gente tuviera razón? El objetivo de Internet es permitir la conversación entre parejas de hosts. El de la Web es la producción y el consumo distribuidos de información. La “propagación de información” es una visión muy general de la comunicación, de la cual las “conversaciones de dos” constituyen un diminuto subconjunto. Necesitamos pasar al ámbito más general. Las redes, hoy en día, tienen que ver con los medios de difusión (radios, redes PON, etc.), simulando ser un cable punto a punto. Eso es espantosamente ineficiente. Por todo el mundo se intercambian terabytes por segundo de datos mediante memorias USB o teléfonos inteligentes, pero no sabemos cómo tratar eso como “redes”. Los ISP están ocupados instalando cachés y redes CDN para distribuir vídeo y audio de forma escalable. Las cachés son una parte necesaria de la solución, pero no hay ninguna parte de las redes actuales - desde la Teoría de la Información, la de Colas o la de Tráfico, hasta las especificaciones de los protocolos Internet - que nos diga cómo proyectarlas e implantarlas. Creo y espero que, en los próximos años, las redes evolucionarán para abarcar esa visión de la comunicación mucho más amplia que subyace a la Web.

¿Qué personas le han inspirado profesionalmente?

Cuando estaba en la Universidad, Richard Feynman vino de visita y nos dio una charla. Nos habló acerca de una parte de la Teoría Cuántica con la que yo había estado luchando todo el semestre, y su explicación fue tan simple y tan lúcida, que lo que había sido un galimatías incomprendible para mí, se convirtió en algo obvio e inevitable. Esa capacidad para ver y transmitir la simplicidad que subyace a nuestro complejo mundo, me parece un don raro y maravilloso.

¿Qué le recomendaría a los estudiantes que quieran orientar su carrera profesional hacia la computación y las redes?

Es un campo maravilloso: las computadoras y las redes probablemente han tenido más impacto sobre la sociedad que cualquier otro invento, desde la imprenta. Las redes se ocupan, fundamentalmente, de conectar cosas, y su estudio nos ayuda a hacer conexiones mentales: el forrajeo de las hormigas y las danzas de las abejas ilustran el diseño de protocolos mejor que los documentos RFC; los atascos de tráfico o la masa de gente que sale de un estadio abarrotado son la esencia de la congestión; y los estudiantes que tratan de encontrar vuelo de vuelta hacia la universidad tras una escapada por Navidad son la mejor representación del enrutamiento dinámico. Si sientes interés por multitud de cosas distintas y quieres dejar huella, es difícil imaginar un campo mejor que el de las redes.

La capa de red: el plano de datos

Hemos estudiado en el capítulo anterior que la capa de transporte proporciona varias formas de comunicación proceso a proceso basándose en el servicio de comunicación host a host de la capa de red. También hemos visto que la capa de transporte lleva a cabo esta tarea sin saber cómo la capa de red implementa realmente este servicio. Así que es posible que se esté preguntando, ¿cuál es el mecanismo subyacente al servicio de comunicación de host a host que lo hace funcionar?

En este capítulo y en el siguiente, vamos a ver exactamente cómo la capa de red puede proporciona su servicio de comunicación host a host. Veremos que, a diferencia de las capas de transporte y de aplicación, *existe un componente de la capa de red en todos y cada uno de los hosts y routers de la red*. Por esta razón, los protocolos de la capa de red se encuentran entre los más desafiantes (¡y, por tanto, entre los más interesantes!) de la pila de protocolos.

La capa de red probablemente sea también la capa más compleja de la pila de protocolos y, por tanto, serán muchas las cuestiones que vamos a tener que abordar. De hecho, son tantos los temas, que dedicaremos dos capítulos a la capa de red. Veremos que esta capa puede descomponerse en dos partes que interactúan mutuamente, el **plano de datos** y el **plano de control**. En el Capítulo 4, hablaremos primero de las funciones del plano de datos de la capa de red —las funciones *implementadas en cada router* de la capa de red que determinan cómo reenviar a uno de los enlaces de salida de ese router los datagramas (es decir, los paquetes de la capa de red) que llegan a través de alguno de los enlaces de entrada del router. Hablaremos tanto del reenvío IP tradicional (en el que el reenvío se basa en la dirección de destino del datagrama), como del reenvío generalizado (en el que pueden llevarse a cabo el reenvío y otra funciones, dependiendo de los valores contenidos en diversos campos de la cabecera del datagrama). Estudiaremos en detalle los protocolos y el direccionamiento IPv4 e IPv6. En el Capítulo 5 hablaremos de las funciones del plano de control de la capa de red —la lógica *global de la red* que controla el modo en que se enruta un datagrama a lo largo de una serie de routers que componen un trayecto extremo a extremo, desde el host de origen hasta el host de destino. Veremos los algoritmos de enrutamiento, así como los protocolos de enrutamiento de uso más extendido en la Internet actual, como OSPF y BGP. Tradicionalmente, estos protocolos de enrutamiento del plano de control y las funciones de reenvío del plano de datos se han implementado de forma conjunta, monolítica, dentro de un router. La técnica de redes definidas por software (SDN, *Software-Defined Networking*) separa explícitamente el plano de datos y el plano de control, implementando las

funciones del plano de control como un servicio separado, situado típicamente en un “controlador” remoto. También hablaremos de los controladores SDN en el Capítulo 5.

Esta distinción entre las funciones del plano de datos y del plano de control, dentro de la capa de red, es un concepto importante, que debemos tener presente mientras estudiemos la capa de red —nos ayudará a estructurar nuestros conceptos sobre la capa de red y refleja también la visión moderna de cuál es el papel que la capa de red juega en las redes de computadoras.

4.1 Introducción a la capa de red

La Figura 4.1 muestra una red simple formada por dos hosts, H1 y H2, y varios routers en la ruta que va de H1 a H2. Supongamos que H1 está enviando información a H2 y veamos el papel de la capa de red en estos hosts y en los routers intervenientes. La capa de red en H1 toma segmentos de la capa de transporte en H1, encapsula cada segmento en un datagrama y, a continuación, envía los datagramas al router más próximo, R1. En el host de recepción, H2, la capa de red recibe los datagramas de su router más próximo R2, extrae los segmentos de la capa de transporte y los entrega a la capa de transporte de H2. La función principal del plano de datos de cada router consiste en reenviar los datagramas desde sus enlaces de entrada a sus enlaces de salida; la función principal del plano de control de la red consiste en coordinar estas acciones de reenvío locales de cada router individual, de modo que los datagramas terminen transfiriéndose de extremo a extremo, a lo largo de series de routers comprendidos entre los hosts de origen y de destino. Observe que los routers de la Figura 4.1 se ilustran con una pila de protocolos truncada, es decir, sin capas por encima de la capa de red, porque los routers no ejecutan protocolos de la capa de transporte ni de la capa de aplicación como los que hemos examinado en los Capítulos 2 y 3.

4.1.1 Reenvío y enrutamiento: los planos de datos y de control

La función principal de la capa de red es engañosamente simple: transporta paquetes desde un host emisor a un host receptor. En la realización de esta tarea podemos identificar dos importantes funciones de la capa de red:

- *Reenvío (forwarding).* Cuando un paquete llega al enlace de entrada de un router, este tiene que pasar el paquete al enlace de salida apropiado. Por ejemplo, un paquete que llega procedente de H1 al router R1 de la Figura 4.1, debe ser reenviado al siguiente router de alguna ruta que conduzca a H2. Como tendremos oportunidad de ver, el reenvío es solo una de las funciones (¡aunque la más importante y común!) implementadas en el plano de datos. En el caso más general, que veremos en la Sección 4.4, también puede prohibirse a un paquete que salga de un router (por ejemplo, si el paquete tiene su origen en un host emisor del que se sabe que es malicioso, o si el paquete está dirigido a un host de destino prohibido), o bien el paquete puede duplicarse y enviarse a través de múltiples enlaces de salida.
- *Enrutamiento (routing).* La capa de red tiene que determinar la ruta o camino que deben seguir los paquetes a medida que fluyen de un emisor a un receptor. Los algoritmos que calculan estas rutas se conocen como **algoritmos de enrutamiento**. Un algoritmo de enrutamiento determinaría, por ejemplo, la ruta por la que fluirán los paquetes para ir de H1 a H2 en la Figura 4.1. El enrutamiento se implementa en el plano de control de la capa de red.

A menudo, los autores que hablan acerca de la capa de red emplean de forma indistinta los términos *reenvío* y *enrutamiento*. Sin embargo, en este libro utilizaremos estos términos de manera mucho más precisa. El **reenvío** hace referencia a la acción local que realiza un router al transferir un paquete desde una interfaz de un enlace de entrada a una interfaz del enlace de salida apropiado.

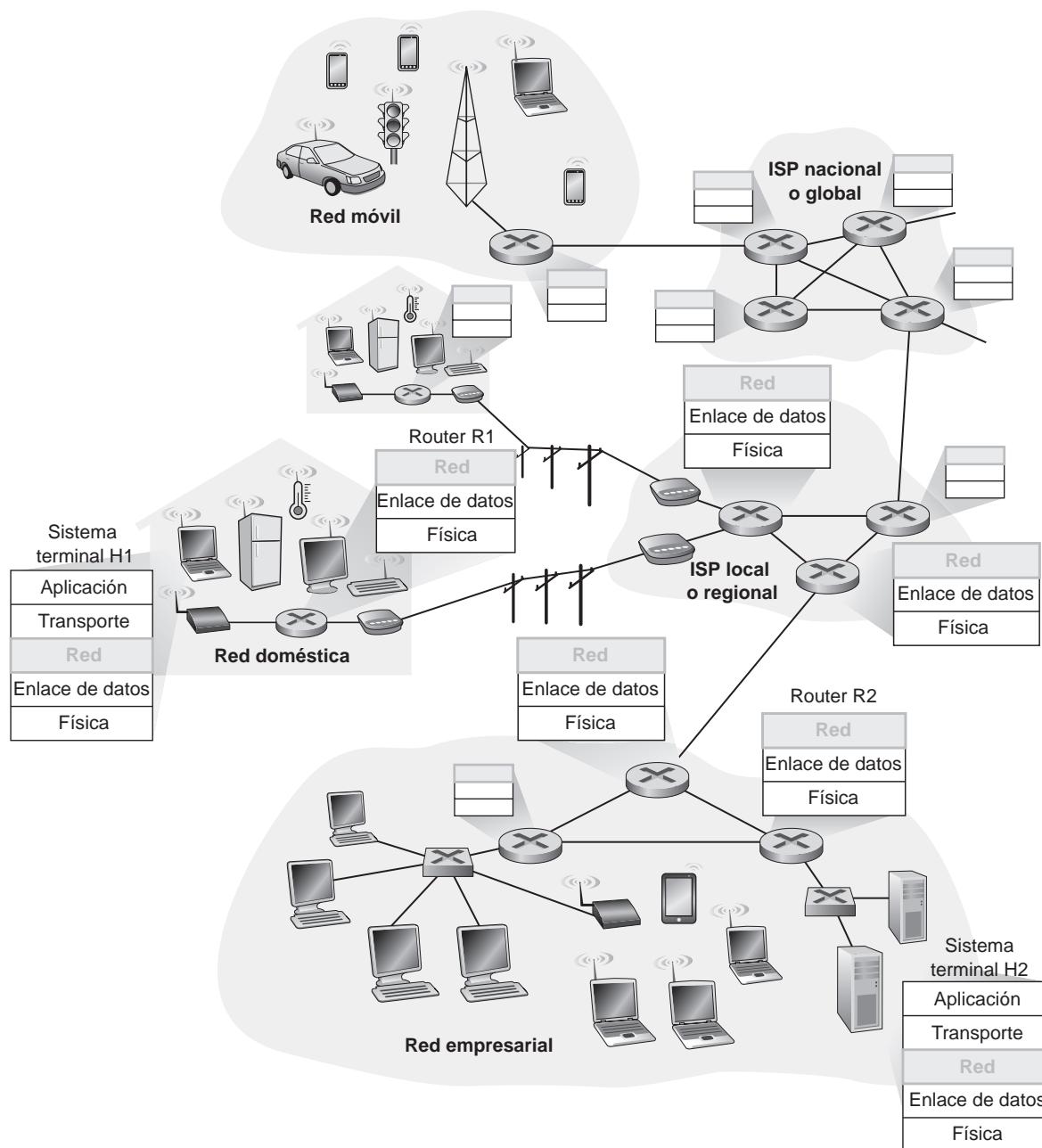


Figura 4.1 ♦ La capa de red.

El reenvío tiene lugar en escalas de tiempo muy cortas (típicamente de unos pocos nanosegundos) y se implementa normalmente en hardware. El **enrutamiento** hace referencia al proceso que realiza la red en conjunto para determinar las rutas extremo a extremo que los paquetes siguen desde el origen al destino. El enrutamiento tiene lugar con escalas de tiempo mucho más largas (normalmente de segundos) y, como veremos, suele implementarse en software. Utilizando nuestra analogía sobre conducir un vehículo, recuerde el trayecto desde Pensilvania a Florida que realizó nuestro viajero de la Sección 1.3.1. Durante ese viaje, nuestro conductor atraviesa muchas intersecciones en su camino a Florida. Podemos pensar en el reenvío como en el proceso de atravesar una intersección: un coche entra en una intersección viniendo por una calle y determina qué otra calle tomar para salir de la intersección. Podemos pensar en el enrutamiento como en el proceso de planificación del viaje desde Pensilvania hasta Florida: antes de iniciar el viaje, el conductor consulta un

mapa y elige uno de los muchos posibles caminos, estando cada uno de ellos definido por una serie de tramos de carretera que se conectan en las intersecciones.

Un elemento crucial en todo router de una red es su **tabla de reenvío**. Un router reenvía un paquete examinando el valor de uno o más campos de la cabecera del paquete entrante y utilizando después esos valores de la cabecera para realizar una indexación dentro de su tabla de reenvío. El valor almacenado en la entrada de la tabla de reenvío correspondiente a esos valores indica cuál es la interfaz del enlace de salida del router a la que hay que reenviar el paquete. Por ejemplo, en la Figura 4.2 un paquete con un valor 0110 en el campo de cabecera llega a un router. El router busca en su tabla de reenvío y determina que la interfaz del enlace de salida para este paquete es la interfaz 2. Entonces, el router reenvía internamente el paquete a la interfaz 2. En la Sección 4.2 examinaremos el interior de un router y analizaremos la función de reenvío con mucho más detalle. El reenvío es la funcionalidad clave del plano de datos de la capa de red.

Plano de control: el enfoque tradicional

El lector se estará ahora preguntando cómo se configuran, para empezar, las tablas de reenvío de un router. Esta cuestión es crucial, ya que expone la importante relación existente entre el reenvío (en el plano de datos) y el enrutamiento (en el plano de control). Como se muestra en la Figura 4.2, el algoritmo de enrutamiento determina el contenido de las tablas de reenvío de los routers. En este ejemplo, se ejecuta un algoritmo de enrutamiento en todos y cada uno de los routers, y cada router contiene funciones tanto de reenvío como de enrutamiento. Como veremos en las secciones 5.3 y 5.4, la función encargada del algoritmo de enrutamiento en un router se comunica con la función correspondiente en otros routers para calcular los valores con los que llenar su tabla de reenvío. ¿Cómo se lleva a cabo esta comunicación? ¡Intercambiando mensajes de enrutamiento, que contienen información de enrutamiento, de acuerdo con lo dispuesto por un protocolo de enrutamiento! Hablaremos de los algoritmos y protocolos de enrutamiento en las secciones 5.2 a 5.4.

La diferencia en los propósitos de las funciones de reenvío y de enrutamiento puede ilustrarse aún más claramente considerando el caso hipotético (y nada realista, pero técnicamente factible) de una red en la que todas las tablas de reenvío fueran configuradas directamente por operadores de red humanos que estuvieran físicamente presentes en los routers. ¡En este caso, *no* se necesitarían

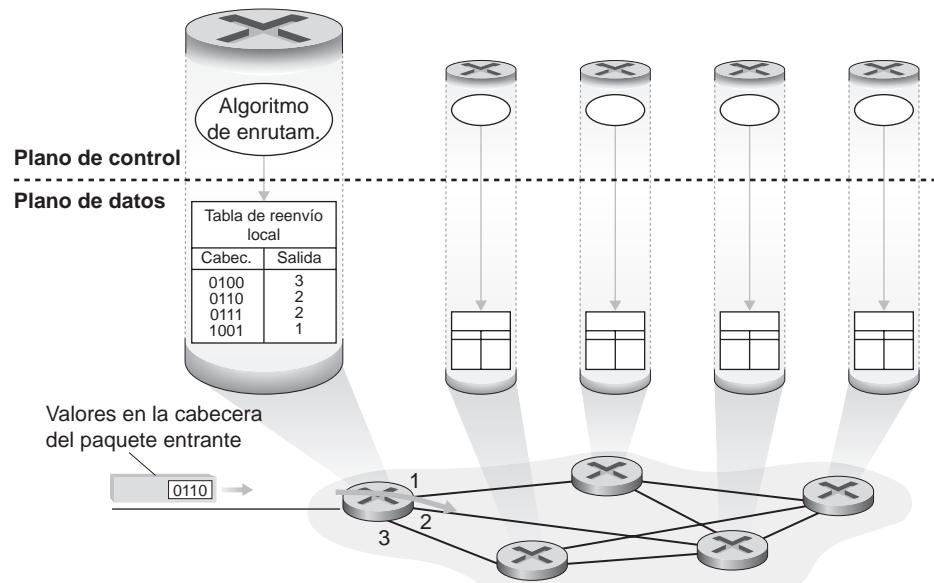


Figura 4.2 ♦ Los algoritmos de enrutamiento determinan los valores almacenados en las tablas de reenvío.

protocolos de enrutamiento! Por supuesto, los operadores humanos tendrían que interactuar entre sí para garantizar que las tablas de reenvío estuvieran configuradas de tal forma que los paquetes llegaran a sus destinos. Probablemente también, si esta configuración la hicieran personas sería más propensa a errores y mucho más lenta en responder a los cambios en la topología de la red que un protocolo de enrutamiento. Por tanto, ¡tenemos suerte de que todas las redes dispongan *tanto* de la función de reenvío *como* de la de enrutamiento!

Plano de control: la técnica SDN

La técnica de implementación de la funcionalidad de enrutamiento mostrada en la Figura 4.2 (en la que cada router tiene un componente de enrutamiento que se comunica con los correspondientes componentes de enrutamiento de otros routers) ha sido la solución tradicional adoptada por los fabricantes de productos de enrutamiento, al menos hasta hace poco. La observación que hacíamos, de que los seres humanos podrían configurar manualmente las tablas de enrutamiento, sugiere, sin embargo, que puede haber otras formas de que las funciones del plano de control determinen el contenido de las tablas de reenvío del plano de datos.

La Figura 4.3 muestra un enfoque alternativo, en el que un controlador remoto, físicamente separado (de los routers), calcula y distribuye las tablas de reenvío que hay que usar en cada router. Observe que los componentes del plano de datos de las Figuras 4.2 y 4.3 son idénticos. En la Figura 4.3, sin embargo, la funcionalidad de enrutamiento del plano de control está separada del router físico —el dispositivo de enrutamiento solo se encarga del reenvío, mientras que el controlador remoto calcula y distribuye las tablas de reenvío—. El controlador remoto puede implementarse en un centro de datos remoto de alta fiabilidad y redundancia, y puede ser gestionado por el ISP o por algún otro proveedor. ¿Cómo pueden comunicarse los routers y el

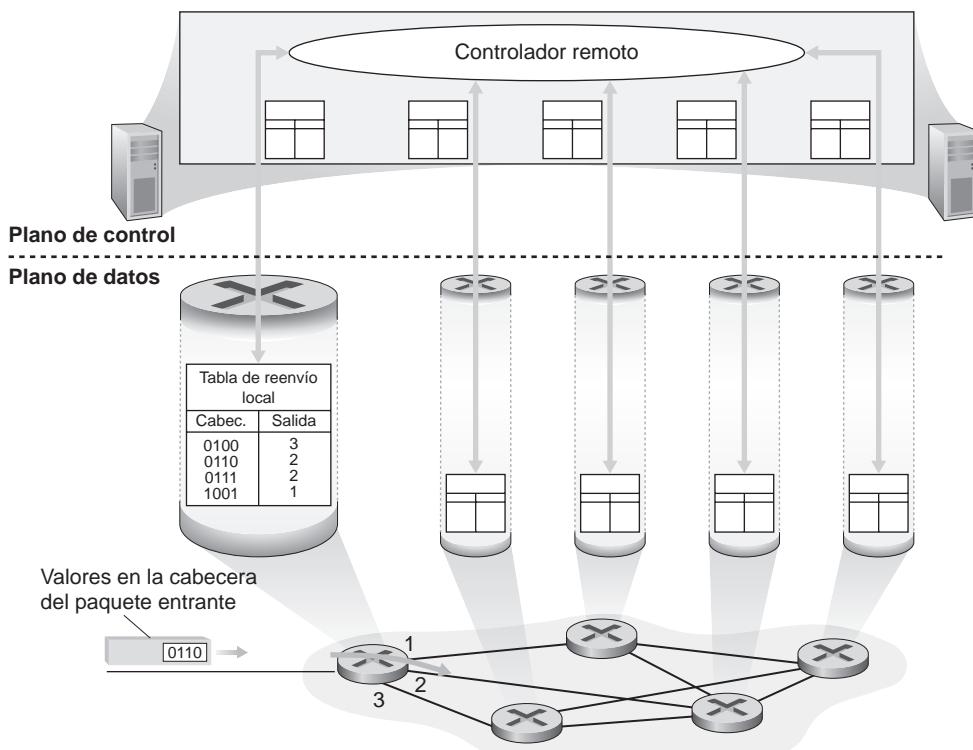


Figura 4.3 ♦ Un controlador remoto determina y distribuye los valores que se almacenan en las tablas de reenvío.

controlador remoto? Intercambiando mensajes que contienen tablas de reenvío y otros tipos de información de enrutamiento. La técnica de implementación del plano de control mostrada en la Figura 4.3 constituye la base de las **redes definidas por software (SDN, Software-Defined Networking)**, donde la red está “definida por software” porque el controlador que calcula las tablas de reenvío y que interacciona con los routers está implementado en software. Cada vez más, estas implementaciones software son también abiertas, es decir, de forma similar al código del sistema operativo Linux, el código de esas implementaciones software está públicamente disponible, permitiendo a los ISP (¡y a los investigadores y estudiantes del campo de las redes!) innovar y proponer cambios del software que controla la funcionalidad de la capa de red. Hablaremos del plano de control SDN en la Sección 5.5.

4.1.2 Modelo de servicio de red

Antes de profundizar en la capa de red, terminemos nuestra introducción considerando desde un punto de vista más amplio los distintos tipos de servicio que puede ofrecer esta capa. Cuando la capa de transporte de un host emisor transmite un paquete a la red (es decir, lo pasa a la capa de red del host emisor), ¿puede la capa de transporte confiar en que la capa de red entregue el paquete al destino? Cuando se envían varios paquetes, ¿se entregan a la capa de transporte del host receptor en el orden en que fueron enviados? ¿El intervalo de tiempo entre el envío de transmisiones de dos paquetes secuenciales será el mismo que el intervalo entre sus respectivas recepciones? ¿Realimentará la red información acerca de la congestión de la misma? Las respuestas a estas y otras preguntas están determinadas por el modelo de servicio proporcionado por la capa de red. El **modelo de servicio de red** define las características del transporte de paquetes extremo a extremo entre los hosts emisor y receptor.

Consideremos ahora algunos de los posibles servicios que podría proporcionar la capa de red. Estos servicios podrían incluir:

- *Entrega garantizada.* Este servicio garantiza que un paquete enviado por un host de origen terminará por llegar al host de destino.
- *Entrega garantizada con retardo limitado.* Este servicio no solo garantiza la entrega del paquete, sino que dicha entrega tendrá un límite de retardo especificado de host a host (por ejemplo, de 100 milisegundos).
- *Entrega de los paquetes en orden.* Este servicio garantiza que los paquetes llegan al destino en el orden en que fueron enviados.
- *Ancho de banda mínimo garantizado.* Este servicio de la capa de red emula el comportamiento de un enlace de transmisión con una velocidad de bit específica (por ejemplo, de 1 Mbps) entre los hosts emisor y receptor. Mientras que el host emisor transmita los bits (como parte de los paquetes) a una velocidad inferior a la velocidad de bit especificada, todos los paquetes terminarán por entregarse al host de destino.
- *Seguridad.* La capa de red podría cifrar todos los datagramas en el origen y descifrarlos en el destino, proporcionando así confidencialidad a todos los segmentos de la capa de transporte.

Esta solo es una lista parcial de los servicios que una capa de red podría proporcionar —existen incontables posibles variaciones—.

La capa de red de Internet proporciona un único servicio, conocido con el nombre de **servicio de mejor esfuerzo (best-effort service)**. Con un servicio de mejor esfuerzo, no está garantizado que los paquetes se reciban en el orden que fueron emitidos y ni siquiera se garantiza la entrega de los paquetes transmitidos. No hay ninguna garantía sobre el retardo extremo a extremo, ni tampoco se garantiza un ancho de banda mínimo. Podría parecer que lo de *servicio de mejor esfuerzo* es un eufemismo para no decir que no proporciona ningún servicio *en absoluto* —¡una red que no entregara *ningún* paquete a su destino satisfaría la definición de servicio con entrega de mejor esfuerzo!—. Otras arquitecturas de red han definido e implementado modelos de servicio que van

más allá del servicio de mejor esfuerzo de Internet. Por ejemplo, la arquitectura de red ATM [MFA Forum 2016, Black 1995] proporciona un retardo secuencial garantizado, un retardo máximo y un ancho de banda mínimo garantizado. También se han propuesto extensiones al modelo de servicio de la arquitectura Internet; por ejemplo, la arquitectura Intserv [RFC 1633] trata de proporcionar garantías de retardo extremo a extremo y una comunicación libre de congestión. Resulta interesante observar que, a pesar de estas alternativas bien desarrolladas, el modelo básico de servicio de mejor esfuerzo de Internet, combinado con una provisión adecuada de ancho de banda, ha resultado ser más que “suficientemente bueno” como para hacer posible un asombroso rango de aplicaciones, incluyendo servicios de flujos de vídeo como Netflix y aplicaciones de voz y vídeo sobre IP para conferencias en tiempo real, como Skype y Facetime.

Una panorámica del Capítulo 4

Habiendo hecho una somera presentación de la capa de red, en las siguientes secciones de este capítulo hablaremos del plano de datos de dicha capa. En la Sección 4.2 profundizaremos en el funcionamiento del hardware interno de un router, incluyendo el procesamiento de los paquetes entrantes y salientes, el mecanismo interno de conmutación de un router y el tema de las colas y la planificación de paquetes. En la Sección 4.3 examinaremos el reenvío IP tradicional, en el que los paquetes se reenvían a los puertos de salida basándose en sus direcciones IP de destino. En esa sección hablaremos del direccionamiento IP, los famosos protocolos IPv4 e IPv6 y muchas cosas más. En la Sección 4.4 veremos mecanismos de reenvío más generalizados, en los que los paquetes pueden ser reenviados a los puertos de salida basándose en un gran número de valores de la cabecera (es decir, no solo basándose en la dirección IP de destino). Los paquetes pueden ser bloqueados o duplicados en el router, o puede que ciertos campos de su cabecera sean reescritos (todo ello bajo control software). Esta forma más generalizada de reenvío de los paquetes es un componente clave del plano de datos de una red moderna, incluyendo el plano de datos de las redes definidas por software (SDN).

Mencionaremos aquí de pasada que los términos *reenvío* y *comutación* se utilizan a menudo de forma intercambiable por parte de los investigadores y profesionales de las redes de computadoras; nosotros también usaremos ambos términos de forma intercambiable en este libro. Y continuando con las cuestiones terminológicas, también merece la pena comentar que hay otros dos términos que se emplean a menudo indistintamente, pero que nosotros emplearemos con más cuidado. Reservaremos el término *comutador de paquetes* para referirnos a un dispositivo de conmutación de paquetes general, que transfiere un paquete desde la interfaz del enlace de entrada a la interfaz del enlace de salida, de acuerdo con los valores almacenados en los campos de la cabecera del paquete. Algunos comutadores de paquetes, denominados **comutadores de la capa de enlace** o switches (que se examinan en el Capítulo 6), basan su decisión de reenvío en valores almacenados en los campos de la trama de la capa de enlace; por eso se dice que los switches son dispositivos de la capa de enlace (capa 2). Otros dispositivos de conmutación de paquetes, conocidos como **routers**, basan su decisión de reenvío en los valores almacenados en los campos de la cabecera del datagrama de la capa de red. Los routers son, por tanto, dispositivos de la capa de red (capa 3). (Con el fin de apreciar esta importante distinción, le invitamos a que repase la Sección 1.5.2, en la que se abordaron los datagramas de la capa de red y las tramas de la capa de enlace, junto con sus relaciones). Puesto que este capítulo está dedicado a la capa de red, utilizaremos fundamentalmente el término *router* en lugar de *comutador de paquetes*.

4.2 El interior de un router

Ahora que ya hemos visto una introducción a los planos de datos y de control de la capa de red, ahora que hemos señalado la importante diferencia entre reenvío y enrutamiento y ahora que hemos presentado los servicios y funciones de la capa de red, vamos a volver nuestra atención a su

función de reenvío: la transferencia real de paquetes desde los enlaces de entrada de un router a los apropiados enlaces de salida.

En la Figura 4.4 se muestra una visión de alto nivel de la arquitectura de un router genérico. Podemos identificar los cuatro componentes siguientes de un router:

- *Puertos de entrada.* Un **puerto de entrada** realiza varias funciones clave. Lleva a cabo la función de la capa física consistente en terminar un enlace físico entrante en un router; esto se muestra mediante el recuadro situado más a la izquierda del puerto de entrada y el recuadro más a la derecha del puerto de salida en la Figura 4.4. Un puerto de entrada también realiza las funciones de la capa de enlace necesarias para interoperar con la capa de enlace en el lado remoto del enlace de entrada; esto se representa mediante los recuadros centrales de los puertos de entrada y de salida. Quizá lo más importante es que también se lleva a cabo una función de búsqueda en el puerto de entrada; esto sucede en el recuadro situado más a la derecha del puerto de entrada. Es aquí donde se consulta la tabla de reenvío, para determinar el puerto de salida del router al que se reenviará un paquete entrante a través del entramado de conmutación. Los paquetes de control (por ejemplo, paquetes que transportan la información del protocolo de enrutamiento) son reenviados desde un puerto de entrada al procesador de enrutamiento. Observe que aquí el término “puerto” —que hace referencia a las interfaces físicas de entrada y salida del router— es claramente distinto de los puertos software asociados con las aplicaciones de red y los sockets, de los que hemos hablado en los Capítulos 2 y 3. En la práctica, el número de puertos soportados por un router puede ir desde un número relativamente pequeño en los routers empresariales, hasta centenares de puertos a 10 Gbps para un router situado en la frontera de un ISP, que es donde el número de líneas entrantes tiende a ser mayor. El router de frontera Juniper MX2020, por ejemplo, soporta hasta 960 puertos Ethernet a 10 Gbps, teniendo el router una capacidad global del sistema de 80 Tbps [Juniper MX 2020 2016].
- *Entramado de conmutación.* El entramado de conmutación conecta los puertos de entrada del router a sus puertos de salida. Este entramado de conmutación está completamente contenido dentro del router: ¡una red dentro de un router de red!
- *Puertos de salida.* Un **puerto de salida** almacena los paquetes recibidos desde el entramado de conmutación y los transmite al enlace de salida, llevando a cabo las funciones necesarias de la capa de enlace y de la capa física. Cuando un enlace es bidireccional (es decir, transporta tráfico en ambas direcciones), el puerto de salida del enlace normalmente estará emparejado con el puerto de entrada de dicho enlace, en la misma tarjeta de línea.
- *Procesador de enrutamiento.* El procesador de enrutamiento lleva a cabo las funciones del plano de control. En los routers tradicionales, ejecuta los protocolos de enrutamiento (que veremos en las secciones 5.3 y 5.4), mantiene las tablas de enrutamiento y la información asociada de estado de los enlaces y calcula la tabla de reenvío para el router. En los routers SDN, el procesador de enrutamiento se encarga de comunicarse con el controlador remoto para (entre otras actividades) recibir entradas de la tabla de reenvío calculadas por el controlador remoto e instalar dichas entradas en los puertos de entrada del router. El procesador de enrutamiento también realiza las funciones de gestión de red que estudiaremos en la Sección 5.7.

Los puertos de entrada, los puertos de salida y el entramado de conmutación de un router se implementan casi siempre en hardware, como se muestra en la Figura 4.4. Para entender por qué hace falta una implementación hardware, piense en que con un enlace de entrada a 10 Gbps y un datagrama IP de 64 bytes, el puerto de entrada solo dispone de 51,2 ns para procesar el datagrama, antes de la posible llegada de otro. Si se combinan N puertos en una tarjeta de línea (como se suele hacer en la práctica), la *pipeline* de procesamiento de datagramas debe funcionar N veces más rápido —demasiada velocidad para una implementación software—. El hardware de reenvío puede implementarse usando los propios diseños hardware del fabricante del router o construirse a partir de chips comerciales de silicio que se adquieran (por ejemplo, como los ofrecidos por empresas como Intel y Broadcom).

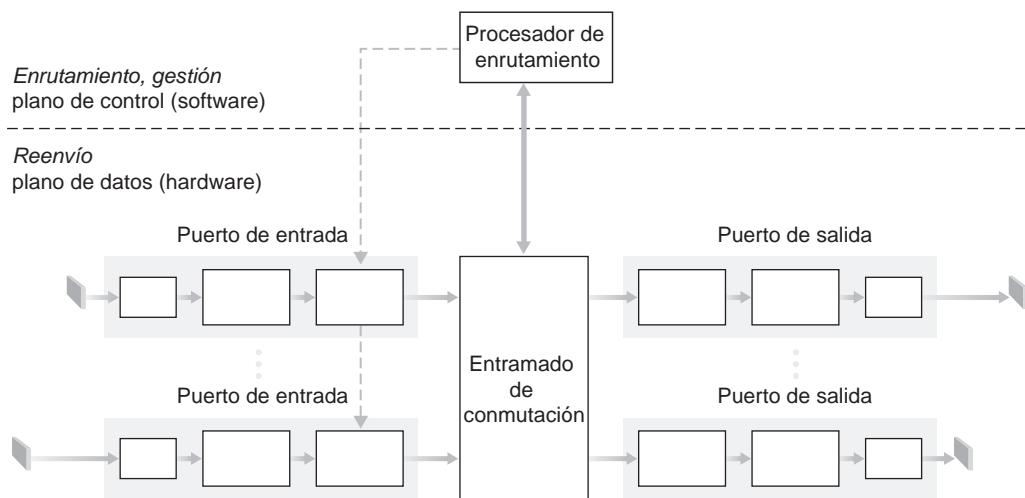


Figura 4.4 ♦ Arquitectura de un router.

Mientras que el plano de datos opera con una escala de tiempo de nanosegundos, las funciones de control de un router (ejecutar los protocolos de enrutamiento, responder a la activación o desactivación de los enlace conectados, comunicarse con el controlador remoto en el caso de SDN y realizar funciones de gestión) operan con una escala de tiempos de milisegundos o incluso de segundos. Por ello, estas funciones del **plano de control** suelen implementarse en software y se ejecutan en el procesador de enrutamiento (normalmente una CPU tradicional).

Antes de entrar en los detalles internos de un router, volvamos a nuestra analogía del principio del capítulo, en la que comparábamos el reenvío de paquetes con los vehículos que entran y salen de una intersección. Supongamos que la intersección es una rotonda y que hace falta un poco de procesamiento cuando un vehículo entra en la rotonda. Veamos qué información hace falta para ese procesamiento:

- *Reenvío basado en el destino (destination-based forwarding)*. Suponga que el vehículo se detiene en una garita a la entrada de la intersección y que indica cuál es su destino final (no en la rotonda, sino el destino último de su viaje). Un operario situado en la garita busca el destino final, determina la salida de la rotonda que conduce a ese destino final y le dice al conductor del vehículo qué salida de la rotonda debe tomar.
- *Reenvío generalizado (generalized forwarding)*. El operario podría determinar también la salida apropiada para el vehículo basándose en otros muchos factores, además del destino. Por ejemplo, la salida podría depender del origen del coche o de la provincia a la que pertenezca su placa de matrícula. Podría ordenarse a los vehículos de una serie de provincias que usaran una determinada salida (que conduce al destino por una carretera más lenta), mientras que a los de otras provincias se les podría indicar que usaran una salida diferente (que conduce al destino a través de una autopista). La misma decisión podría adoptarse basándose en el fabricante, modelo y año de fabricación del vehículo. O podría prohibirse el paso a través de la rotonda a un vehículo que no se considerara apto para circular. En el caso del reenvío generalizado, son muchos los factores que pueden contribuir a la selección de la salida realizada por el operario para un vehículo concreto.

Una vez que el vehículo entra en la rotonda (que puede estar repleta de otros vehículos incorporándose desde otras carreteras de entrada y dirigiéndose hacia otras salidas), terminará por salir a través de la salida prescrita, donde puede encontrarse con otros vehículos que también estén intentando usar esa salida para abandonar la rotonda.

Podemos reconocer fácilmente en esta analogía los cuatro componentes principales de un router mostrados en la Figura 4.4: la carretera de entrada y la garita de entrada se corresponden con el puerto de entrada (con una función de búsqueda para determinar el puerto local de salida); la rotonda se corresponde con el entrampado de conmutación y la carretera de salida de la rotonda se corresponde con el puerto de salida. Con esta analogía, resulta instructivo pensar en dónde podrían producirse cuellos de botella. ¿Qué sucede si los vehículos llegan a una velocidad impresionante (¡por ejemplo, porque la rotonda está en Alemania o Italia!), pero el operario de la garita es lento? ¿A qué velocidad debe trabajar el operario para garantizar que no se forme una cola en una carretera de entrada? Incluso con un operario asombrosamente veloz, ¿qué sucede si los vehículos atraviesan la rotonda lentamente? ¿Podrían producirse atascos en ese caso? ¿Y qué sucede si la mayoría de los vehículos que acceden a la rotonda a través de todas las carreteras de entrada quieren tomar la misma salida de la rotonda? ¿Podrían producirse atascos en la salida, o en algún otro lugar? ¿Cómo debería funcionar la rotonda si quisieramos asignar prioridades a los diferentes vehículos, o si quisieramos prohibir a ciertos vehículos el acceso a la rotonda? Todas estas preguntas son análogas a ciertas cuestiones críticas a las que se enfrentan los diseñadores de routers y switches.

En las siguientes subsecciones, vamos a examinar con mayor detalle las funciones de un router. [Iyer 2008; Chao 2001; Chuand 2005; Turner 1998; McKeown 1997a; Partridge 1998; Serpanos 2011] proporcionan una explicación de arquitecturas de router específicas. En aras de la concreción y de la simplicidad, asumiremos inicialmente en esta sección que las decisiones de reenvío solo están basadas en la dirección de destino del paquete, en vez de en un conjunto generalizado de campos de la cabecera del paquete. Cubriremos el caso del reenvío más generalizado de paquetes en la Sección 4.4.

4.2.1 Procesamiento en el puerto de entrada y reenvío basado en el destino

En la Figura 4.5 se muestra una visión más detallada del procesamiento de entrada. Como acabamos de ver, la función de terminación de línea y el procesamiento de la capa de enlace en el puerto de entrada implementan las capas física y de enlace de datos para ese enlace de entrada concreto. La búsqueda realizada en el puerto de entrada resulta crucial para el funcionamiento del router —es aquí donde el router usa la tabla de reenvío para determinar el puerto de salida al que será reenviado un paquete entrante a través del entrampado de conmutación. La tabla de reenvío es calculada y actualizada por el procesador de enrutamiento (usando un protocolo de enrutamiento para interaccionar con los procesadores de enrutamiento de otros routers de la red) o se recibe desde un controlador SDN remoto. La tabla de reenvío es copiada en las tarjetas de línea desde el procesador de enrutamiento a través de un bus independiente (por ejemplo, un bus PCI), indicado por la línea punteada que va del procesador de enrutamiento a las tarjetas de las líneas de entrada en la Figura 4.4. Estando disponible esa copia de la tabla en cada tarjeta de línea de entrada, las decisiones de reenvío pueden tomarse localmente, en cada puerto de entrada, sin invocar al procesador centralizado de enrutamiento para cada paquete y evitando así la aparición de un cuello de botella en el procesamiento centralizado.

Consideremos ahora el caso “más simple” de que el puerto de salida hacia el que haya que conmutar un paquete entrante esté basado en la dirección de destino del paquete. En el caso de

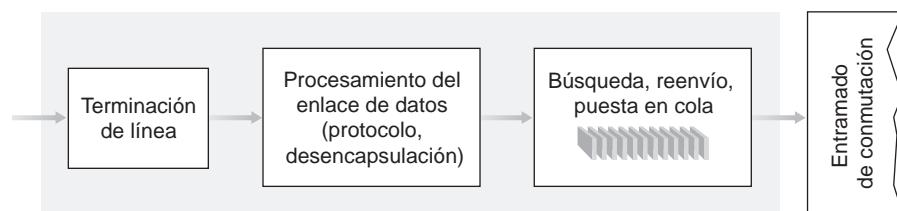


Figura 4.5 ♦ Procesamiento en el puerto de entrada.

direcciones IP de 32 bits, una implementación por fuerza bruta de la tabla de reenvío tendría una entrada para cada una de las posibles direcciones de destino. Dado que existen más de 4.000 millones de posibles direcciones, esta opción está totalmente fuera de lugar.

Como ejemplo de cómo se puede manejar esta cuestión de escala, supongamos que nuestro router tiene cuatro enlaces, numerados de 0 a 3, y que los paquetes deben ser reenviados a las interfaces de enlace como sigue:

Rango de direcciones de destino	Interfaz de enlace
11001000 00010111 00010000 00000000 hasta 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 hasta 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 hasta 11001000 00010111 00011111 11111111	2
En otro caso	3

Evidentemente, para este ejemplo no es necesario disponer de 4.000 millones de entradas en la tabla de reenvío del router. Podríamos, por ejemplo, tener la siguiente tabla de reenvío con solo cuatro entradas:

Prefijo	Interfaz de enlace
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
En otro caso	3

Con este tipo de tabla de reenvío, el router busca la coincidencia de un **prefijo** de la dirección de destino del paquete con las entradas de la tabla; si existe una coincidencia, el router reenvía el paquete a un enlace asociado con esa coincidencia. Por ejemplo, suponga que la dirección de destino del paquete es 11001000 00010111 00010110 10100001; puesto que el prefijo de 21 bits de esta dirección coincide con la primera entrada de la tabla, el router reenvía el paquete a la interfaz de enlace 0. Si un prefijo no se corresponde con ninguna de las tres primeras entradas, entonces el router reenvía el paquete a la interfaz predeterminada, la número 3. Aunque este método puede parecer bastante simple, esconde una importante sutileza. Es posible que se haya dado cuenta de que la dirección de destino puede corresponderse con más de una entrada. Por ejemplo, los primeros 24 bits de la dirección 11001000 00010111 00011000 10101010 coinciden con la segunda entrada de la tabla y los primeros 21 bits con la tercera entrada de la tabla. Cuando existen varias coincidencias, el router aplica la **regla de coincidencia con el prefijo más largo**; es decir, busca la entrada más larga de la tabla con la que exista una coincidencia y reenvía el paquete a la interfaz de enlace asociada con el prefijo más largo. Cuando estudiemos con más detalle el direccionamiento de Internet, en la Sección 4.3, veremos *por qué* exactamente se utiliza esta regla de coincidencia con el prefijo más largo.

Dada la existencia de una tabla de reenvío, la búsqueda resulta conceptualmente simple: la lógica hardware simplemente recorre la tabla de reenvío, buscando la coincidencia con el prefijo más largo. Pero para velocidades de transmisión del orden de los Gigabits, esta búsqueda debe realizarse en nanosegundos (recuerde nuestro ejemplo anterior de un enlace a 10 Gbps y un datagrama IP de 64 bytes). Por tanto, no solo debe realizarse la búsqueda mediante hardware, sino que hacen falta técnicas que vayan más allá de una simple búsqueda lineal en una tabla de gran

tamaño; puede encontrar una revisión de algoritmos rápidos de búsqueda en [Gupta 2001, Ruiz-Sánchez 2001]. También debe prestarse especial atención a los tiempos de acceso a memoria, lo que da como resultado diseños con DRAM integrada en el chip y memorias SRAM más rápidas (usadas como caché DRAM). En la práctica, también se utilizan a menudo para las búsquedas las memorias TCAM (*Ternary Content Addressable Memory*, memoria ternaria direccionable por contenido) [Yu 2004]. Con una TCAM, se presenta una dirección IP de 32 bits a la memoria, que devuelve el contenido de la entrada de la tabla de reenvío correspondiente a esa dirección en un tiempo esencialmente constante. Los routers y switches de las series Cisco Catalyst 6500 y 7600 pueden almacenar más de un millón de entradas TCAM de tabla de reenvío [Cisco TCAM 2014].

Una vez determinado, mediante la búsqueda, el puerto de salida de un paquete, dicho paquete puede ser enviado al entramado de conmutación. En algunos diseños, se puede bloquear temporalmente la entrada del paquete en el entramado de conmutación, si hay paquetes procedentes de otros puertos de entrada que están usando actualmente el entramado. Los paquetes bloqueados serán puestos en cola en el puerto de entrada, planificándose su paso por el entramado para algún instante de tiempo posterior. En breve examinaremos más detalladamente el bloqueo, las colas y la planificación de paquetes (tanto en los puertos de entrada como en los de salida). Aunque la “búsqueda” es, probablemente, la acción más importante dentro del procesamiento en el puerto de entrada, existen muchas otras acciones que hay que llevar a cabo: (1) debe realizarse el procesamiento físico y de la capa de enlace, como hemos comentado anteriormente; (2) hay que comprobar los campos de número de versión, de suma de comprobación y de tiempo de vida del paquete (de todos los cuales hablaremos en la Sección 4.3) y es necesario reescribir estos dos últimos campos y (3) es necesario actualizar los contadores utilizados para la gestión de red (como el del número de datagramas IP recibidos).

Vamos a cerrar nuestra exposición sobre el procesamiento en el puerto de entrada resaltando que los pasos llevados a cabo por el puerto de entrada, en lo referente a buscar una dirección IP de destino (“correspondencia”) y luego enviar el paquete a través del entramado de conmutación al puerto de salida especificado (“acción”), constituyen un caso específico de una abstracción “correspondencia más acción” (*match plus action*) más general que se lleva a cabo en muchos dispositivos de red, no solo en los routers. En los switches de la capa de enlace (de los que hablaremos en el Capítulo 6), se buscan las direcciones de destino de la capa de enlace y pueden llevarse a cabo varias acciones, además de enviar la trama hacia el puerto de salida a través del entramado de conmutación. En los cortafuegos (de los que hablaremos en el Capítulo 8 y que son dispositivos que eliminan diversos paquetes entrantes seleccionados), un paquete entrante cuya cabecera se corresponda con un determinado conjunto de criterios (por ejemplo, una combinación de números de puerto de la capa de transporte y de direcciones IP de origen/destino) puede ser eliminado (acción). En un traductor de direcciones de red (NAT, *Network Address Translator*, de los que hablaremos en la Sección 4.3), un paquete entrante cuyo número de puerto de la capa de transporte se corresponda con un cierto valor, verá su número de puerto reescrito antes del reenvío (acción). De hecho, la abstracción “correspondencia más acción” es potente y prevalente en los dispositivos de red actuales, y resulta clave para la noción de reenvío generalizado de la que hablaremos en la Sección 4.4.

4.2.2 Comutación

El entramado de conmutación se encuentra en el corazón de todo router, ya que es a través de este entramado donde los paquetes son realmente conmutados (es decir, reenviados) desde un puerto de entrada a un puerto de salida. La conmutación puede llevarse a cabo de varias formas, como se muestra en la Figura 4.6:

- *Comutación vía memoria*. Los primeros routers, más simples, eran computadoras tradicionales, en las que la conmutación entre los puertos de entrada y de salida se realizaba bajo el control directo de la CPU (procesador de enrutamiento). Los puertos de entrada y de salida funcionaban como dispositivos de E/S tradicionales en un sistema operativo tradicional. El puerto de entrada al que llegaba un paquete enviaba una señal en primer lugar al procesador de enruta-

miento mediante una interrupción. A continuación, el paquete se copiaba desde el puerto entrada en la memoria del procesador. Después, el procesador de enrutamiento extraía la dirección de destino de la cabecera, buscaba en la tabla de reenvío el puerto de salida apropiado y copiaba el paquete en los buffers del puerto de salida. En este escenario, si el ancho de banda de la memoria es tal que pueden escribirse en, o leerse de, la memoria un máximo de B paquetes por segundo, entonces la tasa de transferencia global de reenvío (la velocidad total a la que los paquetes se transfieren desde los puertos de entrada a los de salida) tiene que ser menor que $B/2$. Observe también que no pueden reenviarse dos paquetes al mismo tiempo, incluso aunque tengan diferentes puertos de destino, ya que solo puede realizarse una lectura/escritura de memoria cada vez a través del bus compartido del sistema.

Algunos routers modernos también realizan la commutación vía memoria. Sin embargo, una diferencia importante con los primeros routers es que los procesadores de las tarjetas de línea de entrada se encargan de realizar la búsqueda de la dirección de destino y de almacenar el paquete en la posición de memoria adecuada. De alguna forma, los routers que commutan vía memoria se parecen mucho a los multiprocesadores de memoria compartida, encargándose los procesadores de las tarjetas de línea de commutar (escribir) los paquetes en la memoria del puerto de salida apropiado. Los commutadores de la serie Catalyst 8500 de Cisco [Cisco 8500 2016] commutan internamente los paquetes mediante una memoria compartida.

- *Commutación vía bus*. Con esta técnica, el puerto de entrada transfiere directamente un paquete al puerto de salida a través de un bus compartido, sin intervención del procesador de enrutamiento. Esto se suele realizar haciendo que el puerto de entrada añada al paquete como prefijo, antes de transmitir el paquete hacia el bus, una etiqueta interna al commutador (cabecera), que indica el puerto local de salida al que se está transfiriendo dicho paquete. Todos los puertos de salida reciben el paquete, pero solo el puerto que se corresponda con la etiqueta lo conservará. Después, la etiqueta es eliminada en el puerto de salida, ya que dicha etiqueta solo se usa dentro del commutador para atravesar el bus. Si llegan múltiples paquetes al router simultáneamente, cada uno a través de un

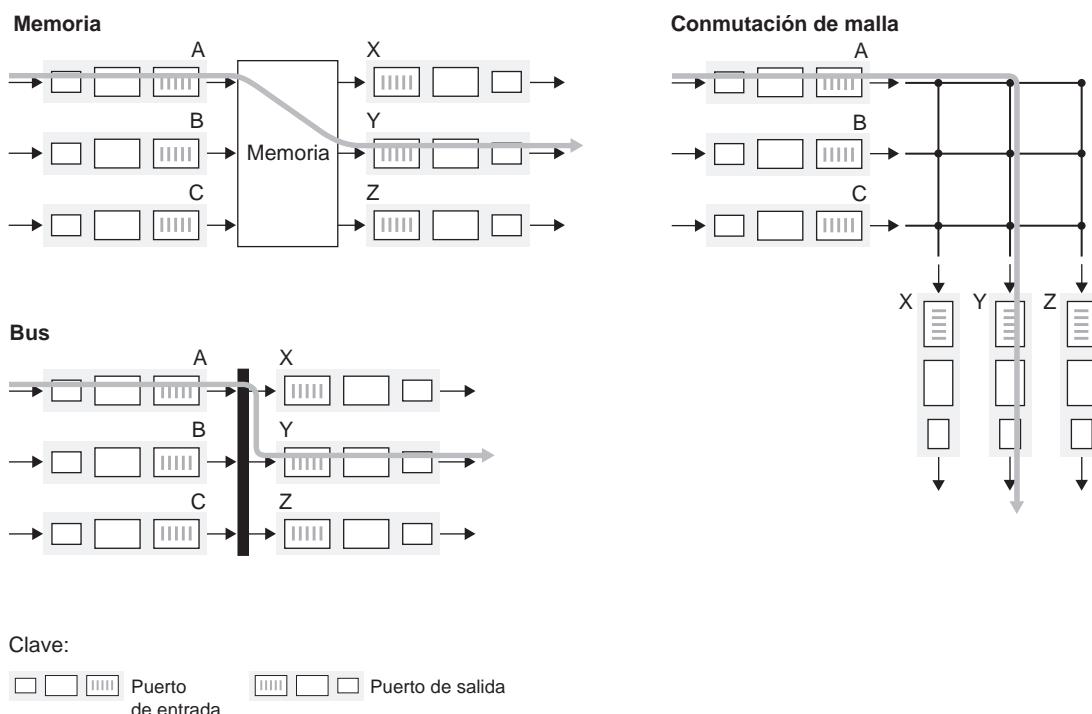


Figura 4.6 ♦ Tres técnicas de commutación.

puerto de entrada distinto, todos los paquetes menos uno deberán esperar, ya que los paquetes deben atravesar el bus de uno en uno. Puesto que todos los paquetes tienen que atravesar el único bus, la velocidad de conmutación del router está limitada por la velocidad del bus; en nuestra analogía de la rotonda, es como si la rotonda solo pudiera contener un único vehículo cada vez. De todos modos, la conmutación vía bus suele ser suficiente para los routers que operan en redes de área local o redes empresariales de pequeño tamaño. El router Cisco 6500 [Cisco 6500 2016] conmuta internamente los paquetes mediante un bus tipo *backplane* a 32 Gbps.

- *Comutación vía una red de interconexión.* Una forma de soslayar la limitación del ancho de banda de un único bus compartido, consiste en emplear una red de interconexión más sofisticada, como las que se han empleado en el pasado para interconectar procesadores en una arquitectura de computadora multiprocesador. Un comutador de malla (*crossbar switch*) es una red de interconexión que consta de $2N$ buses que conectan N puertos de entrada a N puertos de salida, como se muestra en la Figura 4.6. Cada bus vertical intersecta con cada bus horizontal en un punto de cruce, que puede ser abierto o cerrado en cualquier momento por el controlador del entramado de conmutación (cuya lógica forma parte del propio entramado de conmutación). Cuando un paquete llega a través del puerto A y necesita ser reenviado al puerto Y, el controlador del comutador cierra el punto de cruce situado en la intersección de los buses A e Y, y el puerto A envía a continuación a través de su bus el paquete, que será transferido (solo) al bus Y. Observe que puede reenviarse al mismo tiempo un paquete del puerto B hacia el puerto X, ya que los paquetes de A a Y y de B a X utilizan diferentes buses de entrada y de salida. Por tanto, a diferencia de las dos técnicas de conmutación anteriores, los comutadores de malla son capaces de reenviar múltiples paquetes en paralelo. Un comutador de malla es **no bloqueante**: un paquete que esté siendo reenviado hacia un puerto de salida no se verá bloqueado para alcanzar ese puerto, siempre y cuando no haya ningún otro paquete que esté siendo reenviado actualmente hacia ese puerto de salida. Sin embargo, si dos paquetes de dos puertos de entrada distintos están destinados a un mismo puerto de salida, entonces uno de ellos deberá esperar a la entrada, ya que solo se puede enviar un único paquete en cada momento a través de un bus determinado. Los comutadores de la familia 12000 de Cisco [Cisco 12000 2016] utilizan una red de conmutación de malla; la familia Cisco 7600 puede configurarse para utilizar un bus o un comutador de malla [Cisco 7600 2016].

Otras redes de interconexión más sofisticadas utilizan múltiples etapas de elementos de conmutación para permitir que paquetes procedentes de diferentes puertos de entrada viajen simultáneamente hacia un mismo puerto de salida, a través del entramado de conmutación multi-etapa. En [Tobagi 1990] podrá encontrar una panorámica de las arquitecturas de conmutación. El sistema Cisco CRS emplea una estrategia de conmutación no bloqueante de tres etapas. La capacidad de conmutación de un router también puede ampliarse utilizando múltiples entramados de conmutación en paralelo. Con esta solución, los puertos de entrada y de salida se conectan a N entramados de conmutación que funcionan en paralelo. El puerto de entrada descompone el paquete en K segmentos de menor tamaño y envía (“distribuye”) los segmentos a través de K de esos N entramados de conmutación hacia el puerto de salida seleccionado, que vuelve a ensamblar los K segmentos para obtener el paquete original.

4.2.3 Procesamiento en el puerto de salida

El procesamiento en los puertos de salida, como se muestra en la Figura 4.7, toma los paquetes que hayan sido almacenados en la memoria del puerto de salida y los transmite a través del enlace de salida. Esto incluye seleccionar los paquetes y extraerlos de la cola para su transmisión y realizar las funciones de transmisión necesarias de las capas de enlace y física.

4.2.4 ¿Dónde se crean colas?

Si nos fijamos en la funcionalidad de los puertos de entrada y de salida, y en las configuraciones mostradas en la Figura 4.6, es evidente que pueden formarse colas de paquetes en los puertos de

entrada y en los puertos de salida, de la misma forma que habíamos identificado casos en los que los vehículos podían tener que esperar en las entradas y en las salidas de la intersección de tráfico, en nuestra analogía de la rotonda. La ubicación y la longitud de las colas (en los puertos de entrada o en los de salida) dependerá de la carga de tráfico, de la velocidad relativa del entrampado de conmutación y de la velocidad de la línea. Consideremos ahora estas colas con un poco más de detalle, ya que a medida que estas colas crecen, la memoria del router podría terminar agotándose, con lo que se producirá una **pérdida de paquetes** cuando no quede memoria disponible para almacenar los paquetes que lleguen. Recuerde que en nuestras explicaciones anteriores hemos dicho que los paquetes se “perdían dentro de la red” o eran “descartados por un router”. *Es aquí, en estas colas de los routers, donde los paquetes realmente se descartan y se pierden.*

Suponga que las velocidades de línea de entrada y de salida (velocidades de transmisión) son todas ellas idénticas e iguales a R_{line} paquetes por segundo, y que existen N puertos de entrada y N puertos de salida. Para simplificar aún más las explicaciones, supongamos que todos los paquetes tienen la misma longitud fija y que los paquetes llegan a los puertos de entrada sincronizadamente. Es decir, suponemos que el tiempo necesario para enviar un paquete a través de cualquier enlace es igual al tiempo necesario para recibir un paquete a través de cualquier enlace, y que durante ese intervalo de tiempo pueden llegar cero o un paquete a través de cada enlace de entrada. Definimos la velocidad de transferencia del entrampado de conmutación R_{switch} como la velocidad a la que dicho entrampado puede mover los paquetes desde los puertos de entrada hasta los puertos de salida. Si R_{switch} es N veces mayor que R_{line} , entonces las colas que se produzcan en los puertos de entrada serán despreciables. Esto es así porque, incluso en el caso peor, en el que las N líneas de entrada estén recibiendo paquetes y todos esos paquetes deban reenviarse al mismo puerto de salida, el conmutador podrá transferir cada lote de N paquetes (uno por cada puerto de entrada) antes de que llegue el siguiente lote.

Colas de entrada

¿Pero qué sucede si el entrampado de conmutación no es lo suficiente rápido (respecto a las velocidades de línea de entrada) como para transferir *todos* los paquetes que le llegan sin producir retardos? En este caso, también pueden crearse colas de paquetes en los puertos de entrada, ya que los paquetes se irán añadiendo a las colas de los puertos de entrada con el fin de esperar su turno para ser transferidos hacia el puerto de salida a través del entrampado de conmutación. Para ilustrar una importante consecuencia de estas colas, considere un entrampado de conmutación de malla y suponga que (1) todas las velocidades de línea son idénticas, (2) que un paquete puede ser transferido desde cualquier puerto de entrada a un puerto de salida concreto en el mismo intervalo de tiempo que tarda un paquete en ser recibido a través de un enlace de entrada y (3) que los paquetes pasan de una cola de entrada concreta a la cola de salida deseada siguiendo una planificación FCFS. Múltiples paquetes pueden ser transferidos en paralelo, siempre y cuando sus puertos de salida sean diferentes. Sin embargo, si dos paquetes situados en primer lugar de dos colas de entrada están ambos destinados a la misma cola de salida, entonces uno de los paquetes quedará bloqueado y tendrá que esperar en la cola de entrada (el entrampado de conmutación solo puede transferir un paquete a un determinado puerto de salida cada vez).

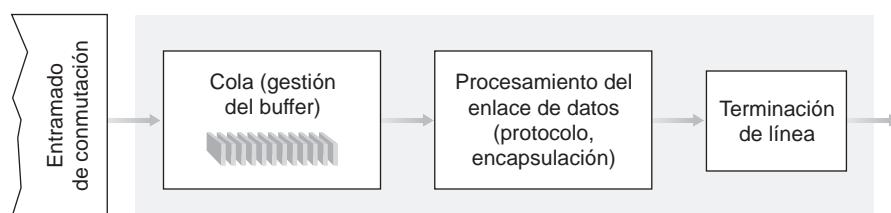


Figura 4.7 ♦ Procesamiento en el puerto de salida.

La Figura 4.8 muestra un ejemplo en el que dos paquetes (los más oscuros) se encuentran al frente de sus respectivas colas de entrada y ambos están destinados al mismo puerto de salida (el de más arriba). Suponga que el entramado de conmutación elige transferir el paquete de la cabecera de la cola superior izquierda. En este caso, el paquete más oscuro de la cola inferior izquierda tendrá que esperar. Pero no solo este paquete más oscuro tiene que esperar, sino también el paquete sombreado más claro que está en la cola detrás del paquete inicial de la cola inferior izquierda, incluso aunque no exista contención para el puerto de salida intermedio (que es el destino del paquete más claro). Este fenómeno se conoce como **bloqueo de cabeza** o **bloqueo HOL** (*Head-of-the-line*, cabeza de línea) en un conmutador con colas de entrada (un paquete que se encuentra en una cola de entrada tiene que esperar a ser transferido a través del entramado de conmutación, aunque su puerto de salida esté libre, porque está bloqueado por otro paquete que se encuentra en la cabeza de la línea). [Karol 1987] demuestra que, a causa del bloqueo HOL, la cola de entrada crecerá de manera ilimitada (informalmente, esto es equivalente a decir que se producirá una pérdida de paquetes significativa) bajo ciertas suposiciones, en cuanto la tasa de llegada de paquetes a través de los enlaces de entrada alcance el 58 por ciento de su capacidad. En [McKeown 1997] se exponen una serie de soluciones para el bloqueo HOL.

Colas de salida

Consideremos a continuación si pueden producirse colas en los puertos de salida de un conmutador. Supongamos de nuevo que R_{switch} es N veces mayor que R_{line} y que los paquetes que llegan a cada uno de los N puertos de entrada están destinados al mismo puerto de salida. En este caso, en el tiempo que se tarda en enviar un único paquete a través del enlace saliente, llegarán N nuevos paquetes a este puerto de salida (uno desde cada uno de los N puertos de entrada). Puesto que el puerto de

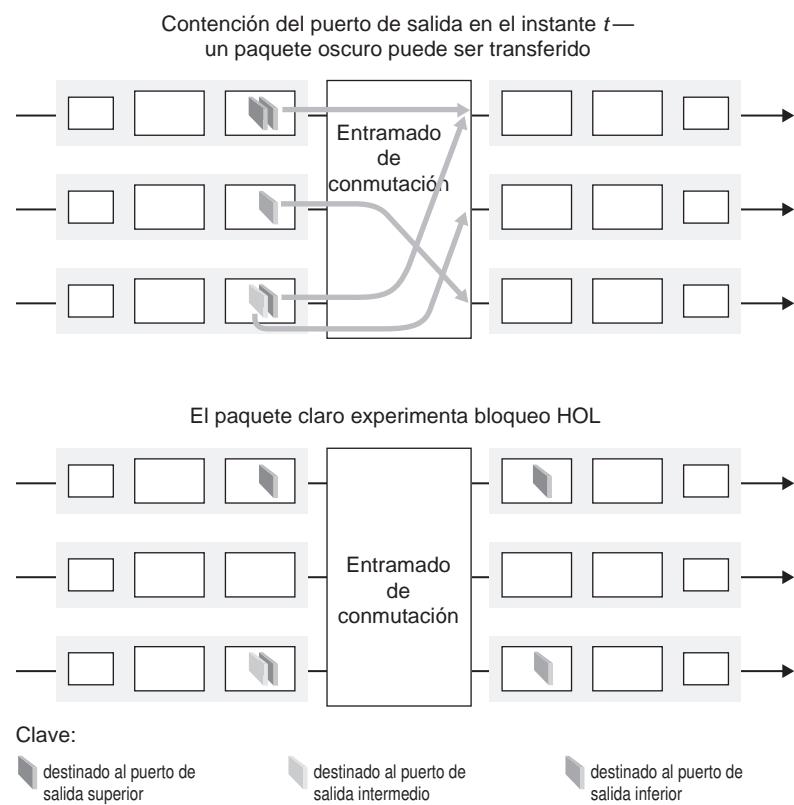


Figura 4.8 ♦ Bloqueo HOL en una cola de entrada de un conmutador.

salida solo puede transmitir un único paquete en cada unidad de tiempo (el tiempo de transmisión del paquete), los N paquetes entrantes tendrán que ponerse a la cola (esperar) para poder ser transmitidos a través del enlace saliente. Entonces, N paquetes adicionales podrían llegar en el tiempo que se tarda en transmitir solo uno de los N paquetes que acababan de ser introducidos en la cola. Y así sucesivamente. Por tanto, pueden formarse colas de paquetes en los puertos de salida incluso aunque el entramado de conmutación sea N veces más rápido que la velocidad de línea de los puertos. Podría así llegar a darse el caso de que el número de paquetes en cola creciera hasta ser lo suficientemente grande como para agotar la memoria disponible en el puerto de salida.

Cuando no hay suficiente memoria como para introducir en el buffer un paquete entrante, hay que tomar la decisión de eliminar ese paquete entrante (una política conocida con el nombre de **eliminación del último**, o en inglés *drop-tail*) o, por el contrario, eliminar uno o más de los paquetes que ya se encuentran en la cola, para hacer sitio para el paquete recién llegado. En algunos casos, puede resultar conveniente eliminar (o marcar la cabecera de) un paquete antes de que el buffer se llene, para así proporcionar al emisor una indicación de congestión. Se han propuesto y analizado diversas políticas proactivas de eliminación y marcado de paquetes (a las que colectivamente se les ha dado en denominar algoritmos **AQM** —*Active Queue Management*, gestión activa de colas—) [Labrador 1999, Hollot 2002]. Uno de los algoritmos AQM más ampliamente estudiados e implementados es el algoritmo **RED** (*Random Early Detection*, detección aleatoria temprana) [Christiansen 2001; Floyd 2016].

El fenómeno de las colas en el puerto de salida se ilustra en la Figura 4.9. En el instante t , ha llegado un paquete a cada uno de los puertos de entrada, y todos ellos están destinados al puerto de salida situado más arriba. Suponiendo velocidades de línea idénticas y un commutador operando a tres veces la velocidad de línea, una unidad de tiempo después (es decir, en el tiempo que se necesita para recibir o enviar un paquete) los tres paquetes originales habrán sido transferidos al puerto de salida y estarán en cola esperando a ser transmitidos. En la siguiente unidad de tiempo, uno de estos tres paquetes habrá sido transmitido a través del enlace de salida. En nuestro ejemplo, llegan *dos nuevos* paquetes a la entrada del dispositivo de conmutación y uno de estos paquetes también tiene como destino el puerto de salida de más arriba. Una consecuencia de la aparición de esa cola es que un

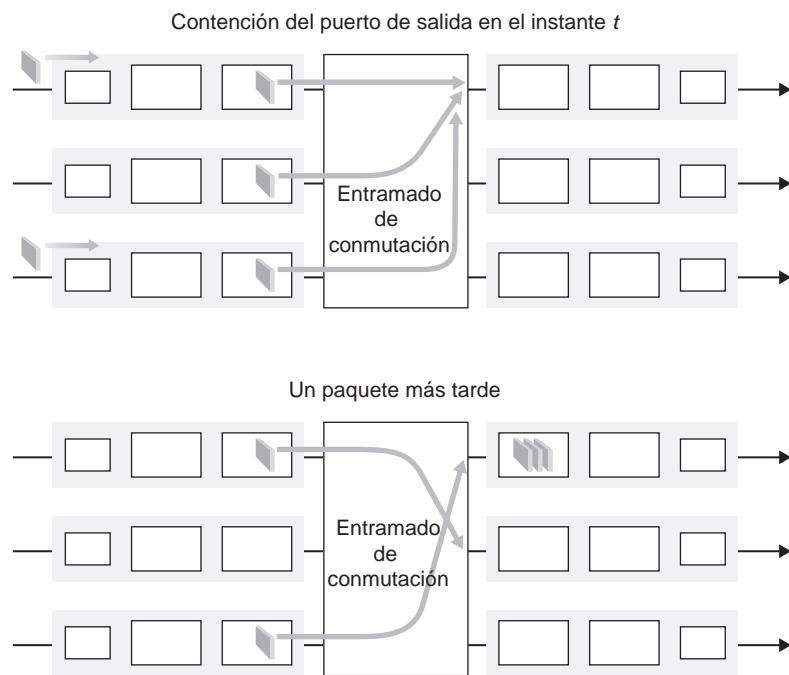


Figura 4.9 ♦ Puesta en cola en el puerto de salida.

planificador de paquetes en el puerto de salida deberá elegir un paquete, de entre todos los de la cola, para su transmisión - un tema del que hablaremos en la próxima sección.

Dado que hacen falta buffers en los routers para absorber las fluctuaciones de la carga de tráfico, es natural plantearse *cuánto* espacio de buffer será necesario. Durante muchos años, la regla heurística [RFC 3439] que se empleaba para determinar el tamaño del buffer era que la cantidad de espacio de buffer (B) debía ser igual al valor promedio del tiempo de ida y vuelta (RTT, digamos 250 milisegundos) multiplicado por la capacidad del enlace (C). Este resultado está basado en el análisis de la dinámica de colas de un número relativamente pequeño de flujos TCP [Villamizar 1994]. Por tanto, un enlace a 10 Gbps con un RTT de 250 milisegundos necesitaría un espacio de buffer igual a $B = RTT \cdot C = 2,5$ Gbits. Sin embargo, recientes esfuerzos teóricos y experimentales [Appenzeller 2004] sugieren que cuando existe un número grande de flujos TCP (N) atravesando un enlace, la cantidad de espacio en buffer necesaria es $B = RTT \cdot C / \sqrt{N}$. Con una gran cantidad de flujos atravesando normalmente los enlaces de router troncales (véase por ejemplo [Fraleigh 2003]), el valor de N puede ser grande, con lo que la reducción del tamaño de buffer necesario se hace bastante significativa. [Appenzeller 2004; Wischik 2005; Beheshti 2008] proporcionan una serie de estudios comprensibles acerca del problema del tamaño del buffer desde los puntos de vista teórico, de implementación y operacional.

4.2.5 Planificación de paquetes

Volvamos a la cuestión de determinar el orden en el que se transmiten a través de un enlace saliente los paquetes existentes en la cola. Dado que el lector habrá tenido, sin lugar a dudas, que esperar en una larga cola en numerosas ocasiones y habrá observado cómo se presta servicio a los clientes que están esperando, estará familiarizado con muchas de las disciplinas de colas utilizadas en los routers. Existe, por ejemplo, la política de “primero que llega, primero al que se sirve”, conocida por las siglas FCFS (*First-Come-First-Served*) o FIFO (*First-In-First-Out*, primero en entrar, primero en salir). Los británicos son famosos por hacer paciente y ordenadamente colas FCFS en las paradas de autobús y en el mercado (“¿Ah, está usted en la cola?”). Otros países utilizan las colas con prioridad, en las que se da prioridad de servicio a una clase de clientes con respecto a otros clientes que están esperando. Existen también las colas de tipo *round-robin* o por turnos, en las que de nuevo se divide a los clientes en clases (como en las colas con prioridad), pero se presta servicio a cada clase de cliente por turno.

FIFO

La Figura 4.10 muestra la abstracción del modelo de cola para la disciplina FIFO de planificación de un enlace. Los paquetes que llegan a la cola de salida del enlace esperan para su transmisión en caso de que el enlace esté actualmente ocupado transmitiendo otro paquete. Si no hay suficiente espacio de buffer para almacenar el paquete entrante, la política de descarte de paquetes de la cola determinará si el paquete es eliminado (perdido) o si se eliminarán otros paquetes de la cola para hacer sitio al paquete entrante, como hemos explicado anteriormente. En las explicaciones que siguen, vamos a ignorar el descarte de paquetes. Cuando se termina de transmitir completamente un

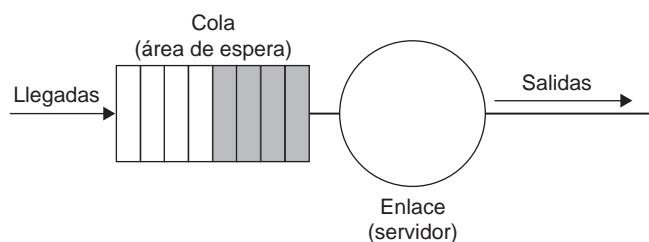


Figura 4.10 ♦ Abstracción de cola FIFO.

paquete a través del enlace saliente (es decir, en cuanto el paquete termina de recibir servicio), se lo elimina de la cola.

La disciplina de planificación FIFO (también denominada FCFS) selecciona los paquetes para su transmisión a través del enlace en el mismo orden en el que llegaron a la cola de salida del enlace. Todos estamos familiarizados con las colas FIFO gracias a nuestra experiencia en cualquier tipo de centros de servicio, donde los clientes que llegan se ponen al final de la única cola existente, permanecen en orden y reciben servicio cuando alcanzan la posición inicial de la línea. La Figura 4.11 ilustra el funcionamiento de la cola FIFO. Las llegadas de paquetes se indican mediante flechas numeradas situadas encima de la línea temporal superior, con el número indicando el orden de llegada del paquete. Las salidas de paquetes individuales se muestran debajo de la línea temporal inferior. El tiempo que un paquete invierte en ser servido (en ser transmitido) se indica mediante el correspondiente rectángulo sombreado situado entre las dos líneas temporales. En estos ejemplos, vamos a asumir que cada paquete tarda tres unidades de tiempo en transmitirse. Bajo la disciplina FIFO, los paquetes salen en el mismo orden en el que entraron. Observe que después de la salida del paquete 4, el enlace permanece inactivo (ya que los paquetes 1 a 4 han sido transmitidos y eliminados de la cola) hasta la llegada del paquete 5.

Colas con prioridad

En el caso de las colas con prioridad, los paquetes que llegan al enlace de salida se clasifican en clases de prioridad al llegar a la cola, como se muestra en la Figura 4.12. En la práctica, un operador de red puede configurar una cola de modo que los paquetes que transporten información de gestión de la red (lo cual se puede saber, por ejemplo, por el número de puerto TCP/UDP de origen o de destino) tengan prioridad sobre el tráfico de usuario; asimismo, los paquetes de voz sobre IP en tiempo real podrían tener prioridad sobre el tráfico que no sea de tiempo real, como los paquetes de correo electrónico SMTP o IMAP. Cada clase de prioridad suele tener su propia cola. Al elegir un paquete para transmitirlo, la disciplina de cola con prioridad transmitirá un paquete de la clase de prioridad más alta que tenga una cola no vacía (es decir, para la que haya paquetes esperando a ser transmitidos). La elección de los paquetes dentro de la misma clase de prioridad suele hacerse por el sistema FIFO.

La Figura 4.13 ilustra el funcionamiento de una cola con prioridad que consta de dos clases de prioridad distintas. Los paquetes 1, 3 y 4 pertenecen a la clase de alta prioridad, mientras que los paquetes 2 y 5 pertenecen a la clase de baja prioridad. El paquete 1 llega y, al encontrar el enlace inactivo, comienza a ser transmitido. Durante la transmisión del paquete 1 llegan los paquetes 2 y 3, que se introducen en las colas de baja prioridad y de alta prioridad, respectivamente. Después de la transmisión del paquete 1, se selecciona para la transmisión el paquete 3 (un paquete de alta prioridad) con preferencia al paquete 2 (el cual, aunque llegó antes, es un paquete de baja prioridad). Al finalizar la transmisión del paquete 3, se comienza a transmitir el paquete 2. El paquete 4

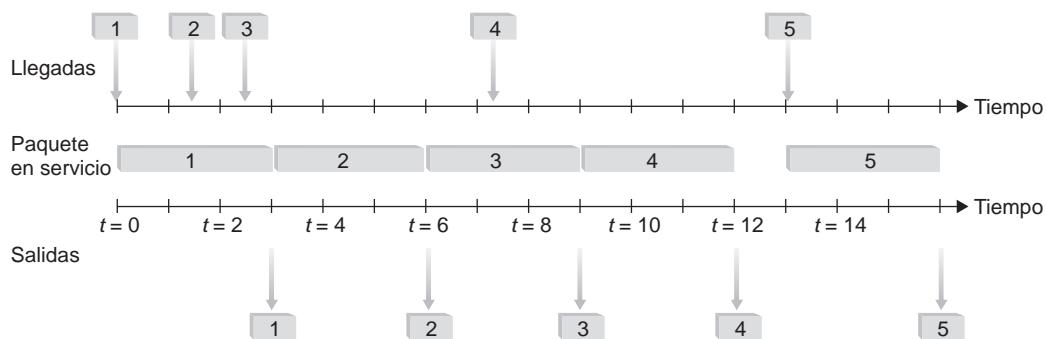


Figura 4.11 ♦ Funcionamiento de la cola FIFO.

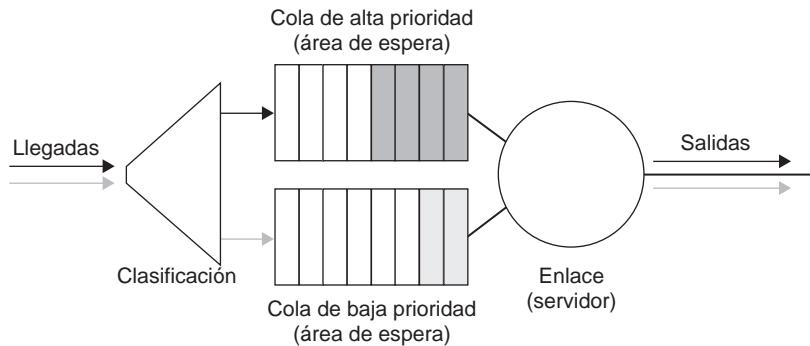


Figura 4.12 ♦ Modelo de cola con prioridad.

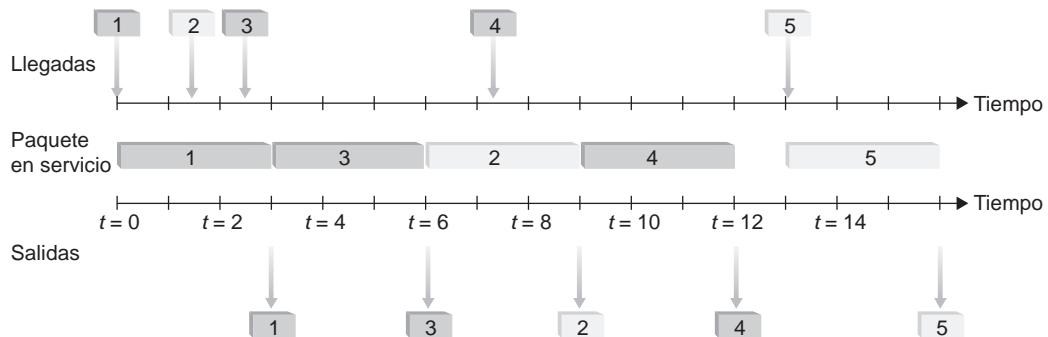


Figura 4.13 ♦ Funcionamiento de la cola con prioridad.

(un paquete de alta prioridad) llega durante la transmisión del paquete 2 (un paquete de baja prioridad). Con la disciplina de **cola con prioridad sin desalojo**, la transmisión de un paquete no se interrumpe una vez que se ha iniciado. En este caso, el paquete 4 se pone en cola de transmisión y comienza a ser transmitido en cuanto se completa la transmisión del paquete 2.

Round robin y colas equitativas ponderadas (WFQ)

Con la disciplina de cola *round robin* (por turnos), los paquetes se distribuyen en clases, igual que en las colas con prioridad. Sin embargo, en lugar de haber una estricta prioridad de servicio entre clases, un planificador *round robin* va alternando el servicio entre las distintas clases. En la forma más simple de planificación *round robin*, se transmite un paquete de clase 1, seguido por un paquete de clase 2, seguido por un paquete de clase 1, seguido de un paquete de clase 2 y así sucesivamente. La denominada disciplina de **cola con conservación del trabajo** no permitirá nunca que el enlace esté inactivo mientras haya paquetes (de cualquier clase) esperando en cola para su transmisión. Una disciplina *round robin* con conservación del trabajo que busque un paquete de una cierta clase y encuentre que no hay ninguno, comprobará inmediatamente la siguiente clase de la secuencia de turnos.

La Figura 4.14 ilustra el funcionamiento de una cola *round robin* con dos clases. En este ejemplo, los paquetes 1, 2 y 4 pertenecen a la clase 1, mientras que los paquetes 3 y 5 pertenecen a la segunda clase. El paquete 1 comienza a ser transmitido inmediatamente después de su llegada a la cola de salida. Los paquetes 2 y 3 llegan durante la transmisión del paquete 1 y, por tanto, se introducen en sus respectivas colas de transmisión. Después de la transmisión del paquete 1, el planificador del enlace busca un paquete de la clase 2 y transmite, por tanto, el paquete 3. Despues de la transmisión del paquete 3, el planificador busca un paquete de la clase 1 y transmite, por tanto, el paquete 2.

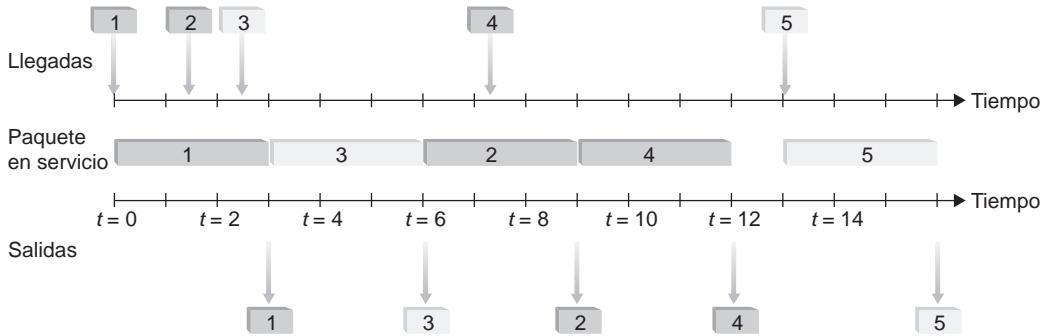


Figura 4.14 ♦ Funcionamiento de una cola round robin con dos clases.

Después de la transmisión del paquete 2, el único paquete que hay en la cola es el paquete 4; por ello, se lo transmite inmediatamente después del paquete 2.

Una forma generalizada de cola *round robin* que se ha implementado ampliamente en routers es la denominada disciplina de **cola equitativa ponderada (WFQ, Weighted Fair Queueing)** [Demers 1990; Parekh 1993; Cisco QoS 2016]. La disciplina WFQ se ilustra en la Figura 4.15. Aquí, los paquetes entrantes se clasifican y se ponen en cola en el área de espera específica de la clase apropiada. Como en la planificación *round robin*, un planificador WFQ prestará servicio a las clases de forma circular: sirviendo primero a la clase 1, luego a la clase 2, luego a la clase 3 y luego (suponiendo que solo existan tres clases) repitiendo la secuencia de servicio. WFQ es también una disciplina de cola con conservación del trabajo y, por tanto, se moverá inmediatamente a la siguiente clase de la secuencia de servicio cuando se encuentre con una cola de clase vacía.

WFQ difiere de la disciplina *round robin* en que cada clase puede recibir una cantidad diferente de servicio en cualquier intervalo de tiempo determinado. Específicamente, a cada clase i se le asigna un peso w_i . Con WFQ, durante cualquier intervalo de tiempo en el que haya paquetes de la clase i para enviar, a la clase i se le garantiza que recibirá una fracción del servicio igual a $w_i/(\sum w_j)$, donde la suma del denominador se realiza para todas las clases que también dispongan de paquetes en cola para su transmisión. En el caso peor, que es cuando todas las clases dispongan de paquetes en cola, a la clase i se le seguirá garantizando que recibirá una fracción $w_i/(\sum w_j)$ del ancho de banda, donde la suma del denominador en el caso peor se realiza para todas las clases. Por tanto, para un enlace con una velocidad de transmisión R , la clase i siempre conseguirá una tasa de transferencia de al menos $R \cdot w_i/(\sum w_j)$. Nuestra descripción de WFQ está idealizada, ya que no hemos tenido en cuenta el hecho de que los paquetes son discretos y de que la transmisión de un paquete no se interrumpirá para comenzar la transmisión de otro paquete; [Demers 1990; Parekh 1993] explican esta cuestión del tamaño discreto de los paquetes.

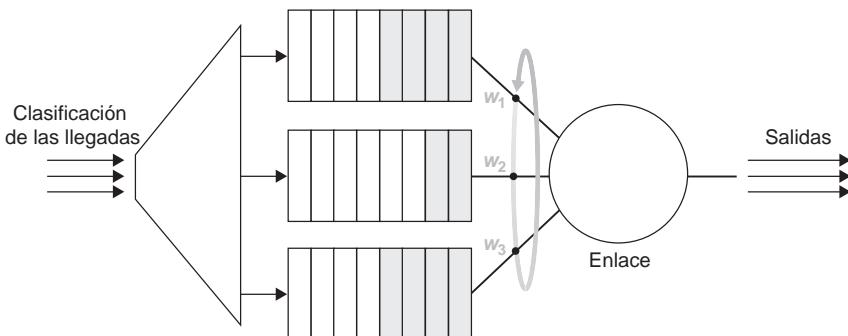


Figura 4.15 ♦ Cola equitativa con prioridad.

4.3 Protocolo de Internet (IP): IPv4, direccionamiento, IPv6 y más

En la exposición que hemos realizado hasta el momento de la capa de red en el Capítulo 4 (la noción de que la capa de red tiene dos componentes, que son el plano de datos y el plano de control; la distinción entre reenvío y enrutamiento; la identificación de diversos modelos de servicio de la red y el examen que hemos hecho del interior de un router), no hemos hecho apenas referencia a ninguna arquitectura de red de computadoras ni protocolo específicos. En esta sección, vamos a fijar nuestra atención en aspectos clave de la capa de red de la actual Internet y en el famoso Protocolo de Internet (IP).

Actualmente hay dos versiones en uso de IP. En primer lugar, examinaremos en la Sección 4.3.1 el muy implantado protocolo IP versión 4, que habitualmente se denomina simplemente IPv4 [RFC 791]. En la Sección 4.3.5, abordaremos la versión 6 de IP [RFC 2460; RFC 4291], protocolo propuesto para sustituir a IPv4. Entre medias, hablaremos principalmente del direccionamiento Internet, un tema que puede parecer árido y demasiado detallado pero que, como veremos, resulta crucial para comprender cómo funciona la capa de red de Internet. ¡Dominar el direccionamiento IP es dominar la propia capa de red de Internet!

4.3.1 Formato de los datagramas IPv4

Recuerde que los paquetes de la capa de red de Internet se denominan *datagramas*. Iniciamos nuestro estudio del protocolo IP con una introducción a la sintaxis y la semántica del datagrama IPv4. Es posible que esté pensando que no puede haber nada más árido que la sintaxis y la semántica de los bits de un paquete. Sin embargo, los datagramas desempeñan un papel central en Internet: todos los estudiantes y profesionales de las redes necesitan comprenderlos y dominarlos. (Y simplemente para ver que puede resultar divertido estudiar las cabeceras de los protocolos, consulte [Pomeranz 2010].) El formato de los datagramas de IPv4 se muestra en la Figura 4.16. Los campos clave de los datagramas de IPv4 son los siguientes:

- *Número de versión.* Estos 4 bits especifican la versión del protocolo IP del datagrama. A partir del número de versión, el router puede determinar cómo interpretar el resto del datagrama IP. Las distintas versiones de IP utilizan distintos formatos de datagrama. El formato de datagrama

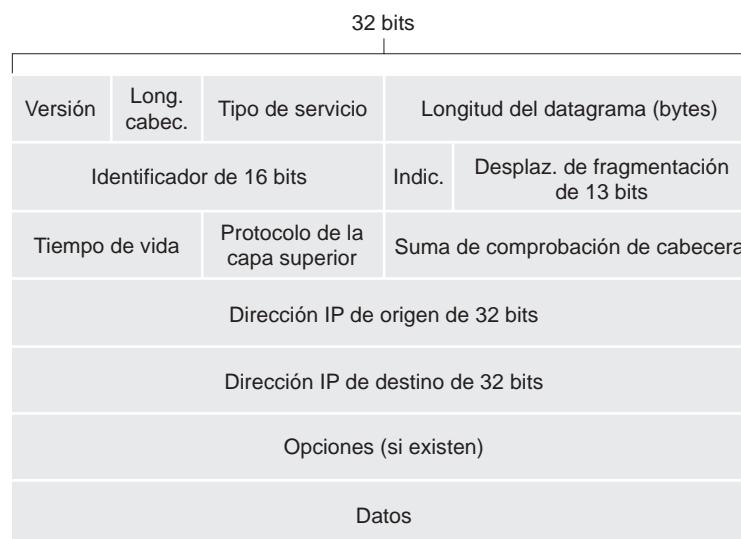


Figura 4.16 ♦ Formato del datagrama IPv4.

para IPv4 es el mostrado en la Figura 4.16. En la Sección 4.3.5 veremos el formato de datagrama correspondiente a la nueva versión de IP (IPv6).

- *Longitud de la cabecera.* Puesto que un datagrama IPv4 puede contener un número variable de opciones (las cuales se incluyen en la cabecera del datagrama IPv4), estos 4 bits son necesarios para determinar dónde comienzan realmente la carga útil (por ejemplo, el segmento de la capa de transporte encapsulado en este datagrama) del datagrama IP. La mayoría de los datagramas IP no contienen opciones, por lo que el datagrama IP típico tiene una cabecera de 20 bytes.
- *Tipo de servicio.* Los bits del tipo de servicio (TOS, *Type Of Service*) se incluyeron en la cabecera de IPv4 con el fin de poder diferenciar entre los distintos tipos de datagramas IP. Por ejemplo, puede resultar útil diferenciar datagramas en tiempo real (como los utilizados en aplicaciones de telefonía IP) del tráfico que no es en tiempo real (como por ejemplo el tráfico FTP). El nivel específico de servicio que se proporcione es una política que determinará y configurará el administrador de red de cada router concreto. También vimos en la Sección 3.7.2 que dos de los bits TOS se utilizan para la notificación explícita de congestión (ECN, *Explicit Congestion Notification*).
- *Longitud del datagrama.* Es la longitud total del datagrama IP (la cabecera más los datos) en bytes. Puesto que este campo tiene una longitud de 16 bits, el tamaño máximo teórico del datagrama IP es de 65.535 bytes. Sin embargo, los datagramas rara vez tienen una longitud mayor de 1.500 bytes, lo que permite que los datagramas IP quepan en el campo de carga útil de una trama Ethernet de tamaño máximo.
- *Identificador, indicadores, desplazamiento de fragmentación.* Estos tres campos tienen que ver con lo que se denomina fragmentación IP, un tema del que hablaremos enseguida. Conviene resaltar que la nueva versión de IP, IPv6, no permite la fragmentación.
- *Tiempo de vida.* El campo Tiempo de vida (TTL, *Time-To-Live*) se incluye con el fin de garantizar que los datagramas no estarán eternamente en circulación a través de la red (debido, por ejemplo, a un bucle de enrutamiento de larga duración). Este campo se decrementa en una unidad cada vez que un router procesa un datagrama. Si el campo TTL alcanza el valor 0, el datagrama tiene que ser descartado por el router.
- *Protocolo.* Este campo solo se suele emplear cuando un datagrama IP alcanza su destino final. El valor de este campo indica el protocolo específico de la capa de transporte al que se pasarán los datos contenidos en ese datagrama IP. Por ejemplo, un valor de 6 indica que los datos se pasan a TCP, mientras que un valor igual a 17 indica que los datos se pasan a UDP. Puede obtener una lista de todos los valores posibles en [IANA Protocol Numbers 2016]. Observe que el número de protocolo especificado en el datagrama IP desempeña un papel análogo al del campo que almacena el número de puerto en un segmento de la capa de transporte. El número de protocolo es el elemento que enlaza las capas de red y de transporte, mientras que el número de puerto es el componente que enlaza las capas de transporte y de aplicación. En el Capítulo 6 veremos que la trama de la capa de enlace también contiene un campo especial que enlaza la capa de enlace con la capa de red.
- *Suma de comprobación de cabecera.* La suma de comprobación de cabecera ayuda a los routers a detectar errores de bit en un datagrama IP recibido. Esta suma de comprobación se calcula tratando cada pareja de 2 bytes de la cabecera como un número y sumando dichos números utilizando aritmética de complemento a 1. Como se ha visto en la Sección 3.3, el complemento a 1 de esta suma, conocido como suma de comprobación Internet, se almacena en el campo Suma de comprobación. Un router calcula la suma de comprobación de cabecera para cada datagrama IP recibido y detecta una condición de error si la suma de comprobación incluida en la cabecera del datagrama no coincide con la suma de comprobación calculada. Normalmente, los routers descartan los datagramas en los que se ha detectado que existe un error. Observe que la suma de comprobación tiene que volver a calcularse y almacenarse en cada router, ya que el campo TTL, y posiblemente también el campo de opciones, pueden cambiar. Una interesante

exposición acerca de algoritmos rápidos para el cálculo de la suma de comprobación Internet puede verse en [RFC 1071]. Una cuestión que suele plantearse en este punto es ¿por qué TCP/IP lleva a cabo una comprobación de errores tanto en la capa de transporte como en la capa de red? Existen varias razones para esta redundancia. En primer lugar, fíjese en que en la capa IP solo se calcula la suma de comprobación para la cabecera IP, mientras que la suma de comprobación TCP/UDP se calcula sobre el segmento TCP/UDP completo. En segundo lugar, TCP/UDP e IP no necesariamente tienen que pertenecer a la misma pila de protocolos. En principio, TCP puede ejecutarse sobre un protocolo diferente de la capa de red (por ejemplo, ATM [Black 1995]) e IP puede transportar datos que no se pasarán a TCP/UDP.

- *Direcciones IP de origen y de destino.* Cuando un origen crea un datagrama, inserta su dirección IP en el campo de dirección IP de origen e inserta la dirección del destino final en el campo de dirección IP de destino. A menudo, el host de origen determina la dirección de destino mediante una búsqueda DNS, como se ha explicado en el Capítulo 2. En la Sección 4.3.3 hablaremos en detalle del direccionamiento IP.
- *Opciones.* El campo de opciones permite ampliar una cabecera IP. La idea original era que las opciones de cabecera rara vez se emplearan: de ahí la decisión de ahorrar recursos no incluyendo la información de los campos opcionales en la cabecera de todos los datagramas. Sin embargo, la mera existencia de opciones complica las cosas, ya que las cabeceras de datagrama pueden tener una longitud variable, por lo que no puede determinarse a priori dónde comenzará el campo de datos. Además, dado que algunos datagramas pueden requerir el procesamiento de opciones y otros no, la cantidad de tiempo necesario para procesar un datagrama IP en un router puede variar enormemente. Estas consideraciones cobran una particular importancia en el procesamiento IP realizado en los hosts y routers de altas prestaciones. Por estas razones y otras, las opciones IP fueron eliminadas en la cabecera de IPv6, como veremos en la Sección 4.3.5.
- *Datos (carga útil).* Finalmente, llegamos al último campo y el más importante: *¡la razón de ser del datagrama!* En la mayoría de las circunstancias, el campo de datos del datagrama IP contiene el segmento de la capa de transporte (TCP o UDP) que va a entregarse al destino. Sin embargo, el campo de datos puede transportar otros tipos de datos, como por ejemplo mensajes ICMP (que veremos en la Sección 5.6).

Observe que un datagrama IP tiene un total de 20 bytes de cabecera (suponiendo que no contiene opciones). Si el datagrama transporta un segmento TCP, entonces cada datagrama (no fragmentado) transporta un total de 40 bytes de cabecera (20 bytes de la cabecera IP más 20 bytes de la cabecera TCP) junto con el mensaje de la capa de aplicación.

4.3.2 Fragmentación del datagrama IPv4

En el Capítulo 6 veremos que no todos los protocolos de la capa de enlace pueden transportar paquetes de la capa de red del mismo tamaño. Algunos protocolos pueden transportar datagramas grandes, mientras que otros solo transportan datagramas pequeños. Por ejemplo, las tramas Ethernet pueden transportar hasta 1.500 bytes de datos, mientras que las tramas para algunos enlaces de área extensa no pueden transportar más de 576 bytes. La cantidad máxima de datos que una trama de la capa de enlace puede transportar se conoce como **unidad máxima de transmisión (MTU, Maximum Transmission Unit)**. Puesto que cada datagrama IP se encapsula dentro de una trama de la capa de enlace para ir de un router al siguiente, la MTU del protocolo de la capa de enlace impone un límite estricto a la longitud de un datagrama IP. Esta limitación del tamaño de un datagrama IP no supone un problema importante. Lo que realmente es un problema es que cada uno de los enlaces existentes a lo largo de la ruta entre el emisor y el destino pueden utilizar diferentes protocolos de la capa de enlace y cada uno de estos protocolos puede emplear una MTU diferente.

Con el fin de comprender mejor la cuestión del reenvío, imagine que *usted* es un router que interconecta varios enlaces, que ejecutan distintos protocolos de la capa de enlace con MTU diferentes. Suponga que recibe un datagrama IP procedente de un enlace. Comprueba su tabla de reenvío para

determinar el enlace de salida y ese enlace de salida tiene una MTU que es menor que la longitud del datagrama IP. Lo que nos lleva a plantearnos la pregunta de cómo meter ese datagrama IP sobredimensionado en el campo de carga útil de la trama de la capa de enlace. La solución consiste en fragmentar la carga útil del datagrama IP en dos o más datagramas IP más pequeños, encapsular cada uno de los datagramas IP más pequeños en una trama de la capa de enlace distinta y enviar dichas tramas a través del enlace de salida. Cada uno de estos datagramas más pequeños se conocen como **fragmentos**.

Los fragmentos tienen que ser reensamblados antes de llegar a la capa de transporte del destino. De hecho, tanto TCP como UDP están esperando recibir de la capa de red segmentos completos, no fragmentos. Los diseñadores de IPv4 pensaron que reensamblar los datagramas en los routers añadiría una complejidad significativa al protocolo y reduciría el rendimiento de los routers. (Si usted fuera un router, ¿querría reensamblar fragmentos, además de todo lo que ya tiene que hacer?) Siguiendo el principio de mantener el núcleo de la red simple, los diseñadores de IPv4 decidieron asignar el trabajo de reensamblar los datagramas a los sistemas terminales, en lugar de a los routers de red.

Cuando un host de destino recibe una serie de datagramas procedentes del mismo origen, tiene que determinar si algunos de esos datagramas son fragmentos de algún otro datagrama original más grande. Si algunos datagramas son fragmentos, tiene que determinar además cuándo ha recibido el último fragmento y cómo debe ensamblar los fragmentos que ha recibido para formar el datagrama original. Para que el host de destino pueda llevar a cabo estas tareas de reensamblado, los diseñadores de IP (versión 4) incluyeron los campos *identificación, indicadores y desplazamiento de fragmentación* en la cabecera del datagrama IP. Cuando se crea un datagrama, el host emisor marca el datagrama con un número de identificación, así como con las direcciones de origen y de destino. Normalmente, el host emisor incrementa el número de identificación para cada datagrama que envía. Cuando un router necesita fragmentar un datagrama, cada datagrama resultante (es decir, cada fragmento) se marca con la dirección de origen, la dirección de destino y el número de identificación del datagrama original. Cuando el destino recibe una serie de datagramas procedentes del mismo host emisor, puede examinar los números de identificación de los datagramas para determinar cuáles de ellos son fragmentos de un mismo datagrama más largo. Puesto que IP es un servicio no fiable, es posible que uno o más de los fragmentos nunca lleguen a su destino. Por esta razón, con el fin de que el host de destino esté absolutamente seguro de que ha recibido el último fragmento del datagrama original, ese último fragmento tiene un bit indicador puesto a 0, mientras que los demás fragmentos tienen el bit indicador puesto a 1. Además, para que el host de destino determine si falta un fragmento (y también para que pueda reensamblar los fragmentos en el orden apropiado), se utiliza el campo desplazamiento para especificar en qué posición dentro del datagrama IP original encaja el fragmento.

La Figura 4.17 proporciona un ejemplo como ilustración. Un datagrama de 4.000 bytes (20 bytes de cabecera IP, más 3.980 bytes de carga útil IP) llega a un router y tiene que ser reenviado a un enlace con una MTU de 1.500 bytes. Esto implica que los 3.980 bytes de datos del datagrama original tienen que ser alojados en tres fragmentos distintos (cada uno de los cuales es también un datagrama IP).

El material en línea del libro y los problemas del final de este capítulo le permitirán explorar con más detalle la cuestión de la fragmentación. Asimismo, en el sitio web del libro proporcionamos un applet de Java que genera fragmentos. No tiene más que proporcionar el tamaño del datagrama de entrada, la MTU y la identificación del datagrama de entrada, y el applet generará automáticamente los fragmentos. Visite el sitio <http://www.pearsonhighered.com/cs-resources/>.

4.3.3 Direcciónamiento IPv4

Ahora vamos a ocuparnos del direcciónamiento IPv4. Aunque puede que piense que el direcciónamiento es un tema sencillo, al terminar esta sección probablemente se haya convencido de que el direcciónamiento en Internet no solo es un tema interesante, profundo y útil, sino que también tiene

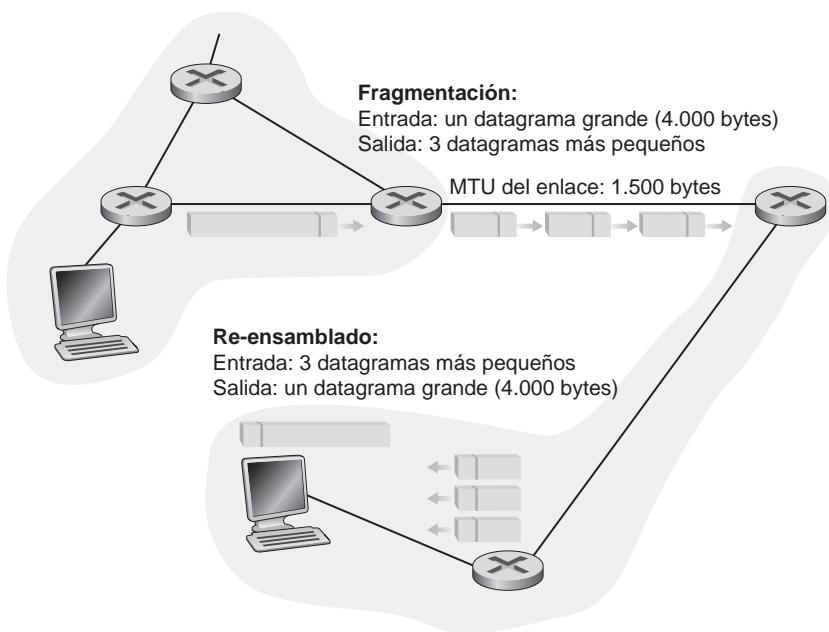


Figura 4.17 ♦ Fragmentación y reensamblado IP.

una importancia crucial para Internet. El primer capítulo de [Stewart 1999] proporciona un excelente tratamiento del direccionamiento IPv4.

Sin embargo, antes de abordar el direccionamiento IP, necesitamos dedicar unas pocas palabras a cómo se conectan los hosts y los routers a la red. Normalmente, un host dispone de un único enlace hacia la red; cuando el protocolo IP del host desea enviar un datagrama, lo hace a través de este enlace. El límite entre el host y el enlace físico se denomina **interfaz**. Consideremos a continuación un router y sus interfaces. Puesto que la tarea de un router consiste en recibir un datagrama por un enlace y reenviarlo a algún otro enlace, un router necesariamente está conectado a dos o más enlaces. El límite entre el router y cualquiera de sus enlaces también se conoce como interfaz. Por tanto, un router tiene varias interfaces, una para cada uno de los enlaces. Puesto que todos los hosts y todos los routers son capaces de enviar y recibir datagramas IP, el protocolo IP requiere que cada interfaz de host y de router tenga su propia dirección IP. *Por tanto, técnicamente, una dirección IP está asociada con una interfaz, en lugar de con el host o con el router que contiene dicha interfaz.*

Las direcciones IP tienen una longitud de 32 bits (lo que equivale a 4 bytes), por lo que existen un total de 2^{32} (unos 4.000 millones) direcciones IP posibles. Estas direcciones normalmente se expresan utilizando la denominada **notación decimal con puntos**, en la que cada byte de la dirección se escribe en formato decimal y se separa mediante un punto del resto de los bytes de la dirección. Por ejemplo, considere la dirección IP 193.32.216.9. El 193 es el número decimal equivalente a los 8 primeros bits de la dirección; el 32 es el equivalente decimal de los segundos 8 bits de la dirección, y así sucesivamente. Por tanto, la dirección 193.32.216.9 en notación binaria se expresa como sigue:

11000001 00100000 11011000 00001001

Cada una de las interfaces de un host o de un router de Internet tiene que tener asociada una dirección IP que sea globalmente única (excepto en el caso de las interfaces utilizadas para NAT, que veremos en la Sección 4.3.4). No obstante, estas direcciones no se pueden elegir a tontas y a locas. Una parte de la dirección IP de una interfaz estará determinada por la subred a la que la interfaz esté conectada.

La Figura 4.18 proporciona un ejemplo de las interfaces y del direccionamiento IP. En esta figura, se utiliza un router (con tres interfaces) para interconectar siete hosts. Echemos un vistazo a

las direcciones IP asignadas a las interfaces de los hosts y del router; hay varios puntos que merece la pena destacar. Los tres hosts de la parte superior izquierda de la Figura 4.18 y la interfaz del router a la que están conectados, tienen todos ellos una dirección IP con el formato 223.1.1.xxx. Es decir, los 24 bits más a la izquierda de la dirección IP de todos ellos son iguales. Además, las cuatro interfaces están interconectadas mediante una red *que no contiene routers*. Esta red podría estar interconectada mediante una LAN Ethernet, en cuyo caso las interfaces se interconectarían mediante un switch Ethernet (como veremos en el Capítulo 6) o mediante un punto de acceso inalámbrico (como veremos en el Capítulo 7). Por el momento, representaremos mediante una nube esa red sin routers que interconecta a esos hosts, y en los Capítulos 6 y 7 profundizaremos en las interioridades de ese tipo de redes.

En términos de IP, esta red que interconecta tres interfaces de host y una interfaz de router forma una **subred** [RFC 950]. (Una subred también se conoce como *red IP* o simplemente *red* en la literatura dedicada a Internet.) El direccionamiento IP asigna una dirección a esta subred: 223.1.1.0/24, donde la notación /24, que en ocasiones se denomina **máscara de subred**, indica que los 24 bits más a la izquierda de ese número de 32 bits definen la dirección de subred. Por tanto, la subred 223.1.1.0/24 consta de tres interfaces de host (223.1.1.1, 223.1.1.2 y 223.1.1.3) y de una interfaz del router (223.1.1.4). Cualquier host adicional conectado a la subred 223.1.1.0/24 requeriría una dirección de la forma 223.1.1.xxx. En la Figura 4.18 se muestran otras dos subredes adicionales: la subred 223.1.2.0/24 y la subred 223.1.3.0/24. La Figura 4.19 ilustra las tres subredes IP presentes en la Figura 4.18.

La definición IP de una subred no está restringida a los segmentos Ethernet que conectan varios hosts a una interfaz de un router. Para profundizar un poco más en esta cuestión, considere la Figura 4.20, que muestra tres routers interconectados entre sí mediante enlaces punto a punto. Cada router tiene tres interfaces, una para cada enlace punto a punto y una para el enlace de difusión que conecta directamente el router a una pareja de hosts. ¿Qué subredes hay presentes aquí? Tres subredes, 223.1.1.0/24, 223.1.2.0/24 y 223.1.3.0/24, son similares a las subredes de la Figura 4.18. Pero fíjese en que, en este ejemplo, también existen tres subredes adicionales: una subred, 223.1.9.0/24, para las interfaces que conectan los routers R1 y R2; otra subred, 223.1.8.0/24, para las interfaces que conectan los routers R2 y R3; y una tercera subred, 223.1.7.0/24, para las interfaces que conectan los routers R3 y R1. En un sistema interconectado general de routers y hosts, podemos utilizar la siguiente receta para definir las subredes existentes en el sistema:

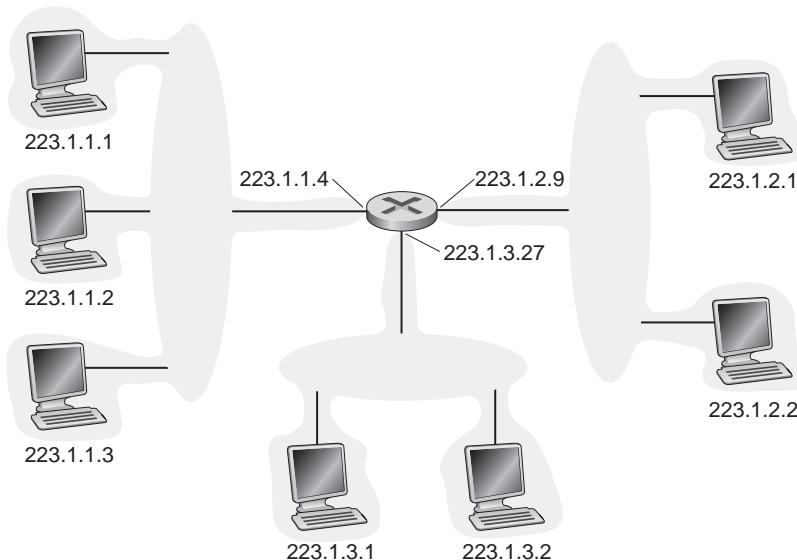
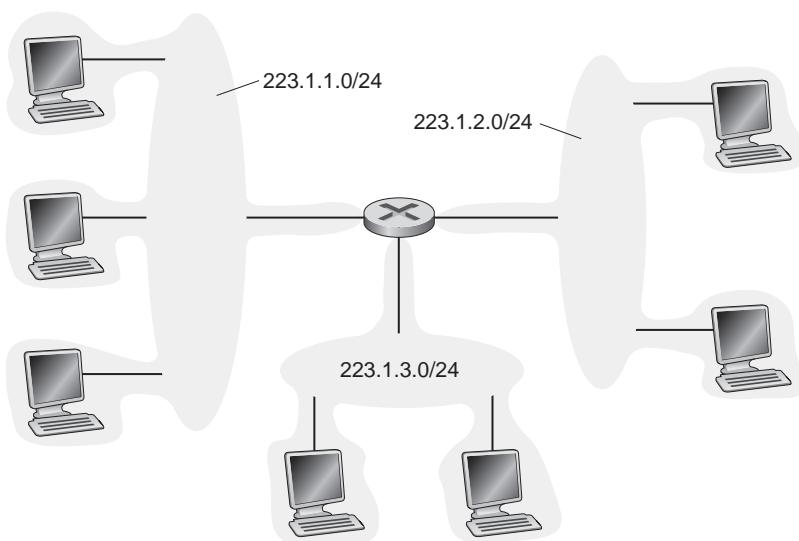
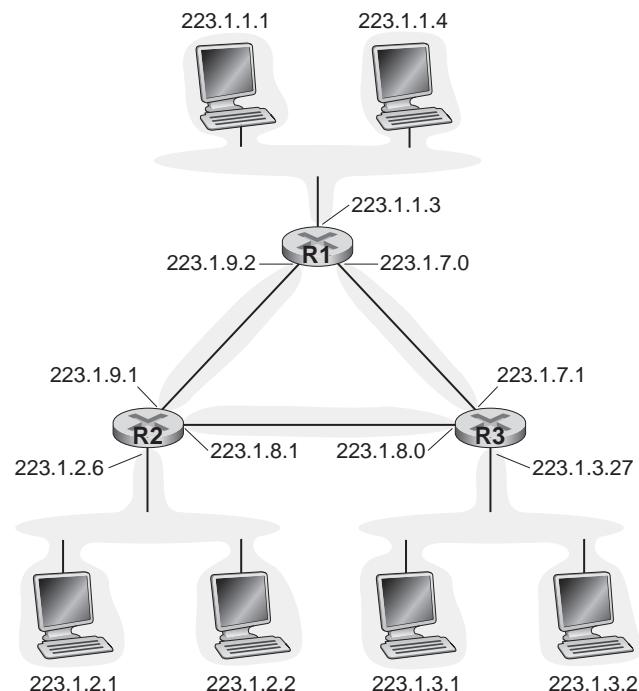


Figura 4.18 ♦ Subredes y direcciones de las interfaces.

**Figura 4.19** ♦ Direcciones de subred.

Para determinar las subredes, desconecte cada interfaz de su host o router, creando islas de redes aisladas, en las que las interfaces actúan como puntos terminales de las redes aisladas. Cada una de estas redes aisladas se dice que es una subred.

Si aplicamos este procedimiento al sistema interconectado de la Figura 4.20, obtenemos seis islas o subredes.

**Figura 4.20** ♦ Tres routers que interconectan seis subredes.

A partir de la exposición anterior, está claro que una organización (como por ejemplo una empresa o una institución académica) con múltiples segmentos Ethernet y enlaces punto a punto tendrá varias subredes, teniendo todos los dispositivos de una subred dada la misma dirección de subred. En principio, las distintas subredes podrían tener direcciones de subred bastante diferentes. Sin embargo, en la práctica, sus direcciones de subred a menudo tienen mucho en común. Para entender por qué, veamos cómo se gestiona el direccionamiento en la Internet global.

La estrategia de asignación de direcciones en Internet se conoce como **enrutamiento entre dominios sin clase (CIDR, Classless Interdomain Routing)** [RFC 4632]. CIDR generaliza la noción de direccionamiento de subred. Al igual que sucede con el direccionamiento de subredes, la dirección IP de 32 bits se divide en dos partes y de nuevo se expresa en notación decimal con puntos como $a.b.c.d/x$, donde x indica el número de bits de la primera parte de la dirección.

Los x bits más significativos de una dirección en el formato $a.b.c.d/x$ constituyen la parte de red de la dirección IP y a menudo se los denomina **prefijo** (o *prefix de red*) de la dirección. Normalmente, una organización tiene asignado un bloque de direcciones contiguas; es decir, un rango de direcciones con un prefijo común (véase el recuadro En la práctica adjunto). En este caso, las direcciones IP de todos los dispositivos de la organización compartirán el mismo prefijo. Cuando estudiemos el protocolo de enrutamiento BGP de Internet en la Sección 5.4, veremos que los routers externos a la red de la organización solo tienen en cuenta estos x primeros bits de prefijo. Es decir, cuando un router externo a la organización reenvía un datagrama cuya dirección de destino está dentro de la organización, únicamente necesita tener en cuenta los primeros x bits de la dirección. Esto reduce considerablemente el tamaño de la tabla de reenvío de los routers, ya que una única entrada con el formato $a.b.c.d/x$ bastará para reenviar paquetes a cualquier destino dentro de la organización.

Los $32-x$ bits restantes de una dirección pueden emplearse para diferenciar los dispositivos *dentro* de la organización, todos los cuales tienen el mismo prefijo de red. Estos son los bits que habrá que considerar para reenviar paquetes en los routers dentro de la organización. Estos bits de menor peso pueden tener (o no) una estructura en subred adicional, como la que hemos visto anteriormente. Por ejemplo, suponga que los 21 primeros bits de la dirección CIDR $a.b.c.d/21$ especifican el prefijo de red de la organización y son comunes a las direcciones IP de todos los dispositivos de dicha organización. Los restantes 11 bits identifican entonces a los hosts específicos de la organización. La estructura interna de la organización puede ser tal que estos 11 bits de más a la derecha se empleen para dividir en subredes la organización, como hemos visto anteriormente. Por ejemplo, $a.b.c.d/24$ podría hacer referencia a una subred específica dentro de la organización.

Antes de que se adoptara el enrutamiento CIDR, la parte de red de una dirección IP estaba restringida a longitudes de 8, 16 o 24 bits, un esquema de direccionamiento conocido como **direccionamiento con clases**, ya que las subredes con direcciones de 8, 16 y 24 bits se conocían, respectivamente, como redes de clase A, B y C. El requisito de que la parte de subred de una dirección IP tuviera exactamente una longitud de 1, 2 o 3 bytes se volvió problemático a la hora de dar soporte al rápido crecimiento del número de organizaciones con subredes de tamaño medio y pequeño. Una subred de clase C (/24) solo puede acomodar hasta $2^8 - 2 = 254$ hosts (dos de las $2^8 = 256$ direcciones están reservadas para usos especiales), que son muy pocos hosts para muchas organizaciones. Sin embargo, una subred de clase B (/16), que puede dar soporte a 65.634 hosts, era demasiado grande. Con el direccionamiento con clases, a una organización con, por ejemplo, 2.000 hosts, se le asignaba normalmente una dirección de subred de clase B (/16). Esto llevó a un rápido agotamiento del espacio de direcciones de clase B y a una poco eficiente utilización del espacio de direcciones asignado. Por ejemplo, la organización que empleaba una dirección de clase B para sus 2.000 hosts tenía asignado espacio suficiente para hasta 65.534 interfaces, dejando bloqueadas más de 63.000 direcciones, que no podían ser utilizadas por otras organizaciones.

Seríamos negligentes si no mencionáramos que existe otro tipo de dirección IP, la dirección IP de difusión 255.255.255.255. Cuando un host envía un datagrama cuya dirección de destino es 255.255.255.255, el mensaje se entrega a todos los hosts existentes en la misma subred. Opcionalmente, los routers también reenvían el mensaje a las subredes vecinas (aunque habitualmente no lo hacen).

EN LA PRÁCTICA

Este ejemplo de un ISP que conecta a ocho organizaciones a Internet ilustra de forma conveniente cómo la asignación cuidadosa de direcciones CIDR facilita el enruteamiento. Suponga, como se muestra en la Figura 4.21, que el ISP (al que llamaremos ISP A) anuncia al mundo exterior que se le debe enviar cualquier datagrama cuyos primeros 20 bits de dirección se correspondan con 200.23.16.0/20. El resto del mundo no necesita saber que dentro del bloque de direcciones 200.23.16.0/20 existen en realidad otras ocho organizaciones, cada una con sus propias subredes. Esta capacidad de emplear un mismo prefijo para anunciar múltiples redes suele denominarse **agregación de direcciones** (o **agregación de rutas**, o también **resumen de rutas**).

La técnica de agregación de direcciones funciona extraordinariamente bien cuando las direcciones se asignan en bloques a los ISP y éstos las asignan en bloques a las organizaciones cliente. Pero, ¿qué ocurre si las direcciones no están asignadas de esa forma jerárquica? ¿Qué ocurriría, por ejemplo, si el ISP A adquiere el ISP B y luego hace que la Organización 1 se conecte a Internet a través del ISP B subsidiario? Como se muestra en la Figura 4.21, el ISP B subsidiario posee el bloque de direcciones 199.31.0.0/16, pero las direcciones IP de la Organización 1 lamentablemente no pertenecen a este bloque de direcciones. ¿Qué habría que hacer en este caso? Por supuesto, la Organización 1 podría renombrar todos sus routers y hosts para disponer de direcciones contenidas en el bloque de direcciones del ISP B. Pero ésta es una solución costosa y la Organización 1 podría ser reasignada en el futuro a otro ISP subsidiario. La solución que normalmente se adoptará es que la Organización 1 mantenga sus direcciones IP en 200.23.18.0/23. En este caso, como se muestra en la Figura 4.22, el ISP A continúa anunciando el bloque de direcciones 200.23.16.0/20 y el ISP B continúa anunciando el bloque 199.31.0.0/16. Sin embargo, el ISP B también anuncia ahora el bloque de direcciones de la Organización 1, 200.23.18.0/23. Cuando otros routers de Internet vean los bloques de direcciones 200.23.16.0/20 (del ISP A) y 200.23.18.0/23 (del ISP B) y deseen enrutar hacia una dirección contenida en el bloque 200.23.18.0/23, utilizarán la regla de *coincidencia con el prefijo más largo* (véase la Sección 4.2.1) y llevarán a cabo el enruteamiento hacia el ISP B, ya que anuncia el prefijo de dirección más largo (es decir, más específico) que se corresponde con la dirección de destino.

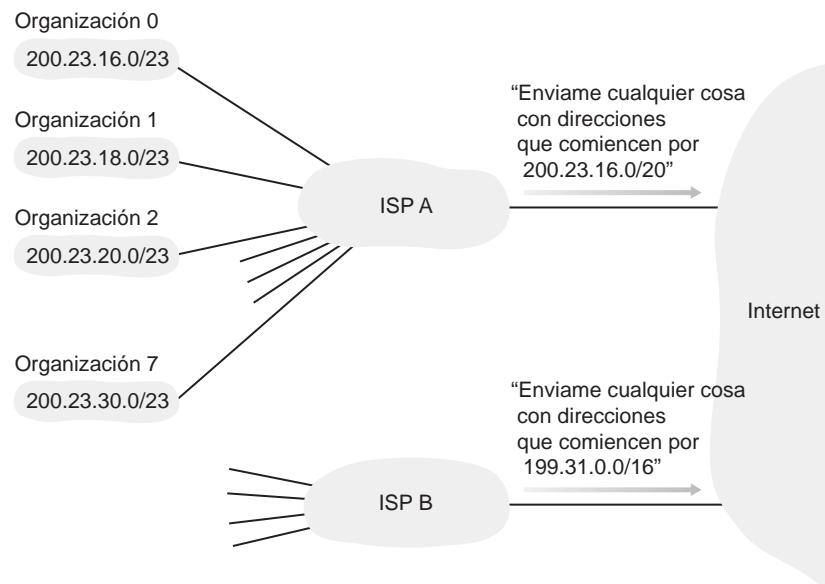


Figura 4.21 ♦ Agregación de rutas y direccionamiento jerárquicos.

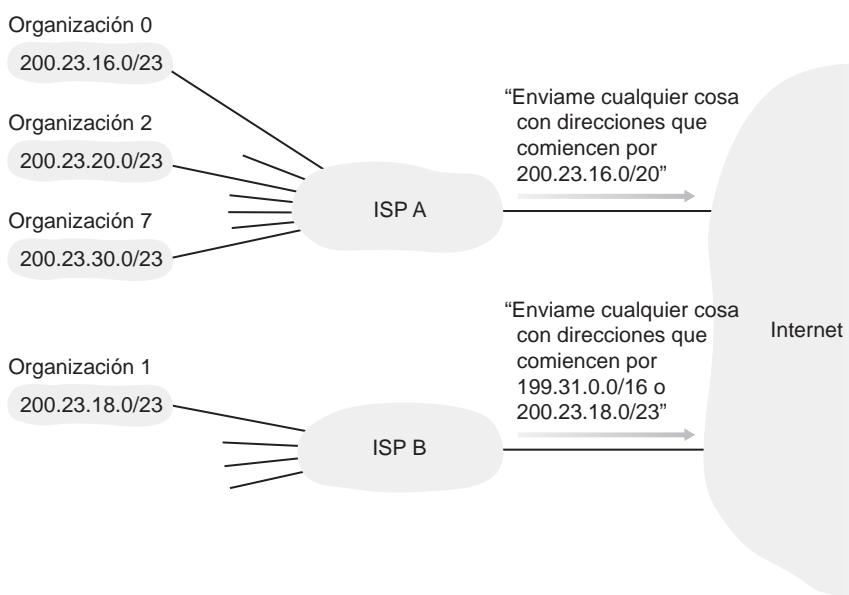


Figura 4.22 ♦ El ISP B tiene una ruta más específica hacia la Organización 1.

Ahora que hemos estudiado en detalle el direccionamiento IP, necesitamos saber cómo los hosts y las subredes obtienen sus direcciones en primer lugar. Comenzaremos viendo cómo una organización obtiene un bloque de direcciones para sus dispositivos, y luego veremos cómo se asigna una dirección del bloque de direcciones de la organización a un dispositivo (por ejemplo, a un host).

Cómo obtener un bloque de direcciones

Para obtener un bloque de direcciones IP que pueda ser utilizado dentro de la subred de una organización, el administrador de red tiene que contactar en primer lugar con su ISP, el cual le proporcionará direcciones extraídas de un bloque de direcciones mayor que ya habrá sido asignado al ISP. Por ejemplo, al ISP pueden haberle asignado el bloque de direcciones 200.23.16.0/20. A su vez, el ISP podría dividir su bloque de direcciones en ocho bloques de direcciones contiguas del mismo tamaño y asignar cada uno de estos bloques de direcciones a hasta ocho organizaciones a las que puede prestar servicio, como se muestra a continuación (hemos subrayado la parte de subred de estas direcciones, para facilitar su identificación).

Bloque del ISP:	200.23.16.0/20	<u>11001000 00010111 00010000 00000000</u>
Organización 0	200.23.16.0/23	<u>11001000 00010111 00010000 00000000</u>
Organización 1	200.23.18.0/23	<u>11001000 00010111 00010010 00000000</u>
Organización 2	200.23.20.0/23	<u>11001000 00010111 00010100 00000000</u>
...
Organización 7	200.23.30.0/23	<u>11001000 00010111 00011110 00000000</u>

Obtener un conjunto de direcciones de un ISP es una forma de conseguir un bloque de direcciones, pero no es la única forma. Evidentemente, también debe existir una forma para el propio ISP de obtener un bloque de direcciones. ¿Existe una autoridad global cuya responsabilidad última sea gestionar el espacio de direcciones IP y asignar bloques de direcciones a los ISP y otras organizaciones? ¡Por supuesto que existe! Las direcciones IP son gestionadas por la ICANN

(*Internet Corporation for Assigned Names and Numbers*, Corporación de Internet para los números y nombres asignados) [ICANN 2016], basándose en las directrices establecidas en [RFC 7020]. El papel de la organización sin ánimo de lucro ICANN [NTIA 1998] no es solo el de asignar direcciones IP, sino también gestionar los servidores raíz DNS. También tiene el polémico trabajo de asignar nombres de dominio y de resolver las disputas por dichos nombres. La organización ICANN asigna direcciones a los registros regionales de Internet (por ejemplo, ARIN, RIPE, APNIC y LACNIC, que forman la Organización de Soporte de Direcciones de ICANN [ASO-ICANN 2016]), los cuales gestionan la asignación/administración de direcciones dentro de sus regiones.

Cómo obtener una dirección de host: Protocolo de configuración dinámica de host

Una vez que una organización ha obtenido un bloque de direcciones, puede asignar direcciones IP individuales a las interfaces de sus hosts y routers. Normalmente, un administrador de sistemas configura manualmente las direcciones IP de un router (a menudo de forma remota, mediante una herramienta de gestión de red). Las direcciones de host también se pueden configurar manualmente, pero frecuentemente esta tarea se lleva cabo utilizando el **Protocolo de configuración dinámica de host (DHCP, Dynamic Host Configuration Protocol)** [RFC 2131]. DHCP permite a un host obtener (permite que se le asigne) automáticamente una dirección IP. Un administrador de red puede configurar DHCP de modo que un host dado reciba la misma dirección IP cada vez que se conecte a la red, o bien a un host puede asignársele una **dirección IP temporal** que será diferente cada vez que el host se conecte a la red. Además de la asignación de direcciones IP de host, DHCP también permite que un host obtenga información adicional, como por ejemplo su máscara de subred, la dirección de su router del primer salto [a menudo denominado pasarela (*gateway*) predeterminada] y la dirección de su servidor DNS local.

Gracias a la capacidad de DHCP de automatizar el proceso de conexión de un host a una red, a menudo se dice que es un protocolo **plug-and-play** o de **configuración cero (zeroconf)** ¡Esta capacidad le hace muy atractivo para el administrador de la red, que en otro caso tendría que realizar estas tareas manualmente! DHCP también disfruta de un amplio uso en las redes residenciales de acceso a Internet, las redes empresariales y las redes LAN inalámbricas, en las que los hosts se unen a la red y salen de ella frecuentemente. Considere, por ejemplo, un estudiante que traslada una computadora portátil desde su casa a la biblioteca y luego a clase. Probablemente, en cada localización el estudiante se conectará a una subred y, por tanto, necesitará una nueva dirección IP en cada lugar. DHCP está idealmente adaptado para estas situaciones, ya que existen muchos usuarios que van y vienen, y que necesitan direcciones solo durante un periodo de tiempo limitado. La ventaja de la capacidad plug-and-play de DHCP está claro, ya que resulta inimaginable que un administrador de sistemas pueda reconfigurar las computadoras portátiles en cada posible ubicación y pocos estudiantes (¡excepto los que estén siguiendo un curso de redes de computadoras!) tienen los conocimientos necesarios para configurar sus portátiles manualmente.

DHCP es un protocolo cliente-servidor. Normalmente, el cliente es un host recién llegado que desea obtener información de configuración de la red, incluyendo una dirección IP para sí mismo. En el caso más simple, cada subred (en el sentido de direccionamiento mostrado en la Figura 4.20) tendrá un servidor DHCP. Si en la subred no hay ningún servidor, es necesario un agente de retransmisión DHCP (normalmente un router) que conozca la dirección de un servidor DHCP para dicha red. La Figura 4.23 muestra un servidor DHCP conectado a la subred 223.1.2/24, con el router actuando como agente de retransmisión para los clientes recién llegados que se conectan a las subredes 223.1.1/24 y 223.1.3/24. En la siguiente exposición, supondremos que hay disponible un servidor DHCP en la subred.

Para un host recién llegado a una red, el protocolo DHCP es un proceso de cuatro pasos, como se muestra en la Figura 4.24 para la configuración de red mostrada en la Figura 4.23. En esta figura, *sudirI* (“su dirección Internet”) indica la dirección que se asigna al cliente que acaba de llegar. Los cuatro pasos son los siguientes:

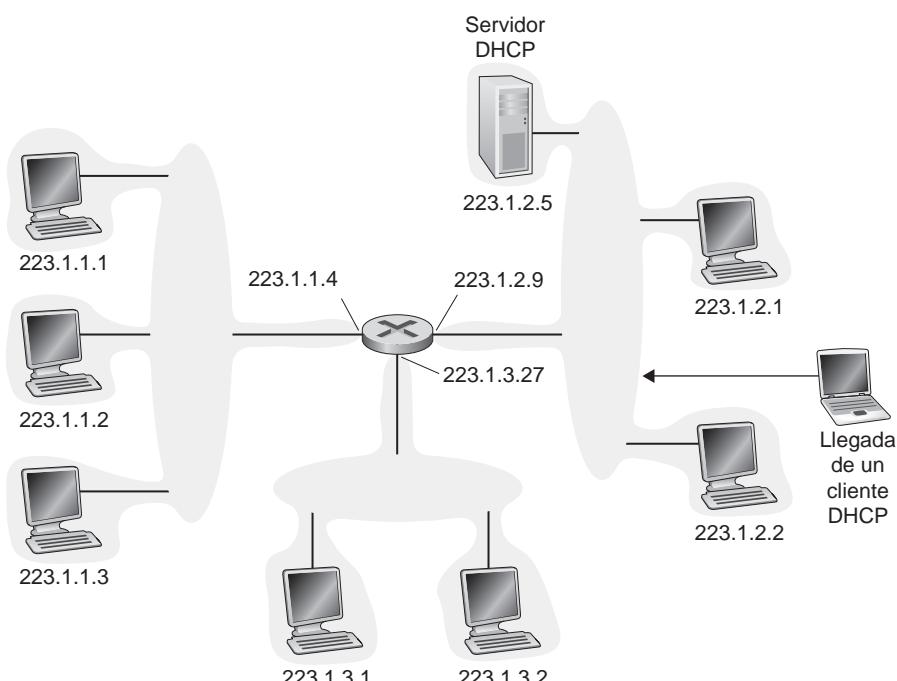


Figura 4.23 ♦ Servidor y clientes DHCP.

- **Descubrimiento del servidor DHCP.** La primera tarea de un host recién llegado es encontrar un servidor DHCP con el que interactuar. Esto se hace mediante un **mensaje de descubrimiento DHCP**, que envía un cliente dentro de un paquete UDP al puerto 67. El paquete UDP se encapsula en un datagrama IP. Pero, ¿a quién debería enviarse este datagrama? El host ni siquiera conoce la dirección IP de la red a la que se está conectando, y mucho menos la dirección de un servidor DHCP de esa red. En esta situación, el cliente DHCP crea un datagrama IP que contiene su mensaje de descubrimiento DHCP junto con la dirección IP de difusión 255.255.255.255 y una dirección IP de origen de “este host” igual a 0.0.0.0. El cliente DHCP pasa el datagrama IP a la capa de enlace, la cual difunde esta trama a todos los nodos conectados a la subred (en la Sección 6.4 estudiaremos en detalle el proceso de difusión de la capa de enlace).
- **Oferta(s) del servidor DHCP.** Un servidor DHCP que recibe un mensaje de descubrimiento DHCP responde al cliente con un **mensaje de oferta DHCP**, que se difunde a todos los nodos de la subred utilizando de nuevo la dirección IP de difusión 255.255.255.255 (trate de pensar en por qué la respuesta de este servidor también debe difundirse a todos los nodos). Puesto que en la subred pueden existir varios servidores DHCP, el cliente puede encontrarse en la enviable situación de poder elegir entre varias ofertas. Cada mensaje de oferta de un servidor contiene el ID de transacción del mensaje de descubrimiento recibido, la dirección IP propuesta para el cliente, la máscara de red y el **tiempo de arrendamiento de la dirección IP** (el tiempo durante el que la dirección IP será válida). Es habitual que el servidor defina un tiempo de arrendamiento de varias horas o días [Droms 2002].
- **Solicitud DHCP.** El cliente recién llegado seleccionará de entre las ofertas de servidor y responderá a la oferta seleccionada con un **mensaje de solicitud DHCP**, devolviendo los parámetros de configuración.
- **ACK DHCP.** El servidor contesta al mensaje de solicitud DHCP con un **mensaje ACK DHCP**, que confirma los parámetros solicitados.

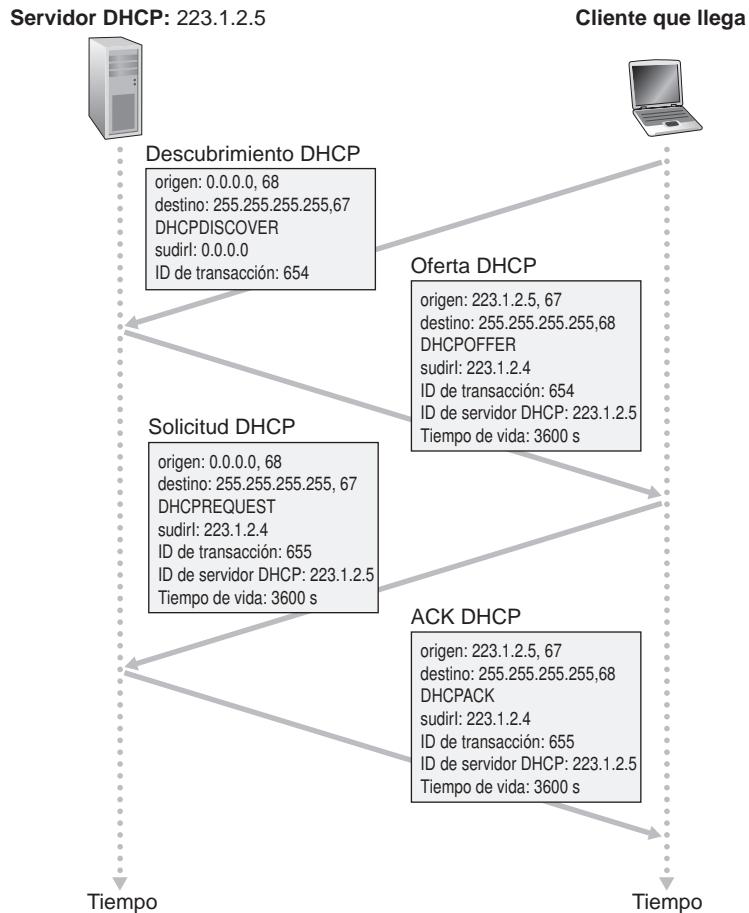


Figure 4.24 ♦ Interacción cliente-servidor DHCP.

Una vez que el cliente recibe el mensaje de confirmación (ACK) DHCP, la interacción se habrá completado y el cliente puede utilizar la dirección IP asignada por DHCP durante todo el tiempo de arrendamiento. Dado que un cliente puede desear utilizar su dirección durante más tiempo del arrendado, DHCP también proporciona un mecanismo que permite a un cliente renovar su arrendamiento de una dirección IP.

En lo que respecta a la movilidad, DHCP presenta una deficiencia importante. Puesto que se obtiene una nueva dirección IP mediante DHCP cada vez que un nodo se conecta a una nueva subred, una conexión TCP con una aplicación remota no podría mantenerse a medida que un nodo móvil se desplaza de una subred a otra. En el Capítulo 6 veremos la infraestructura de IP móvil, una extensión de la infraestructura IP que permite a un nodo móvil utilizar una única dirección permanente según se va desplazando entre subredes. Puede encontrar información adicional sobre DHCP en [Droms 2002] y [dhc 2016]. En Internet Systems Consortium [ISC 2016] hay disponible una implementación de referencia de código fuente abierto para DHCP.

4.3.4 Traducción de direcciones de red (NAT)

Después de haber estudiado las direcciones de Internet y el formato de los datagramas IPv4, somos completamente conscientes de que todo dispositivo IP necesita una dirección IP. Con la proliferación de las subredes de oficina doméstica y pequeña oficina (SOHO, *Small Office, Home Office*), podría parecer que esto implica que, cuando una red SOHO desea instalar una LAN para conectar

varias máquinas, el ISP debería asignar un rango de direcciones para cubrir todos los dispositivos IP de la red SOHO (incluyendo teléfonos, tabletas, dispositivos de juegos, televisiones IP, impresoras y otros). Si la subred creciera, habría que asignar un bloque de direcciones mayor. Pero ¿qué ocurre si el ISP ya ha asignado las porciones adyacentes al rango de direcciones actual de la red SOHO? ¿Y qué persona normal querría (o necesitaría) saber cómo gestionar las direcciones IP de la red de su casa? Afortunadamente, existe una forma más simple de asignar direcciones que ha encontrado un uso cada vez más amplio en escenarios de este tipo: la **traducción de direcciones de red (NAT, Network Address Translation)** [RFC 2663; RFC 3022; Huston 2004; Zhang 2007; Cisco NAT 2016].

La Figura 4.25 muestra el funcionamiento de un router con funcionalidad NAT. Este router, que se encuentra en una vivienda, tiene una interfaz que forma parte de la red doméstica situada en la parte derecha de la Figura 4.25. El direccionamiento dentro de la red doméstica es exactamente como hemos visto anteriormente (las cuatro interfaces de la red tienen la misma dirección de subred 10.0.0.24). El espacio de direcciones IP que está reservado en [RFC 1918] para una **red privada** o para un **ámbito con direcciones privadas**, como la red doméstica de la Figura 4.25. Un ámbito con direcciones privadas hace referencia a una red cuyas direcciones solo tienen significado para los dispositivos internos de dicha red. Veamos por qué esto es importante. Considere el hecho de que existen cientos de miles de redes domésticas y que muchas utilizan el mismo espacio de direcciones, 10.0.0.0/24. Los dispositivos de una red doméstica dada pueden enviar paquetes entre sí utilizando el direccionamiento 10.0.0.0/24. Sin embargo, los paquetes reenviados *más allá* de la red doméstica, hacia la Internet global, evidentemente no pueden utilizar estas direcciones (ni como dirección de origen ni como dirección de destino), porque existen cientos de miles de redes que emplean ese mismo bloque de direcciones. Es decir, las direcciones 10.0.0.0/24 solo tienen significado dentro de una red doméstica dada. Pero si las direcciones privadas solo tienen significado dentro de la red, ¿cómo se dirigen los paquetes cuando se envían a Internet o se reciben de Internet, donde necesariamente las direcciones tienen que ser únicas? Para entender esto hay que comprender cómo funciona NAT.

El router NAT no *parece* un router a ojos del mundo exterior. En su lugar, el router NAT se comporta de cara al exterior como un *único* dispositivo con una dirección IP única. En la Figura 4.25, todo el tráfico que sale del router doméstico hacia Internet tiene una dirección IP de origen igual a 138.76.29.7, y todo el tráfico que entra en él tiene que tener la dirección de destino 138.76.29.7.

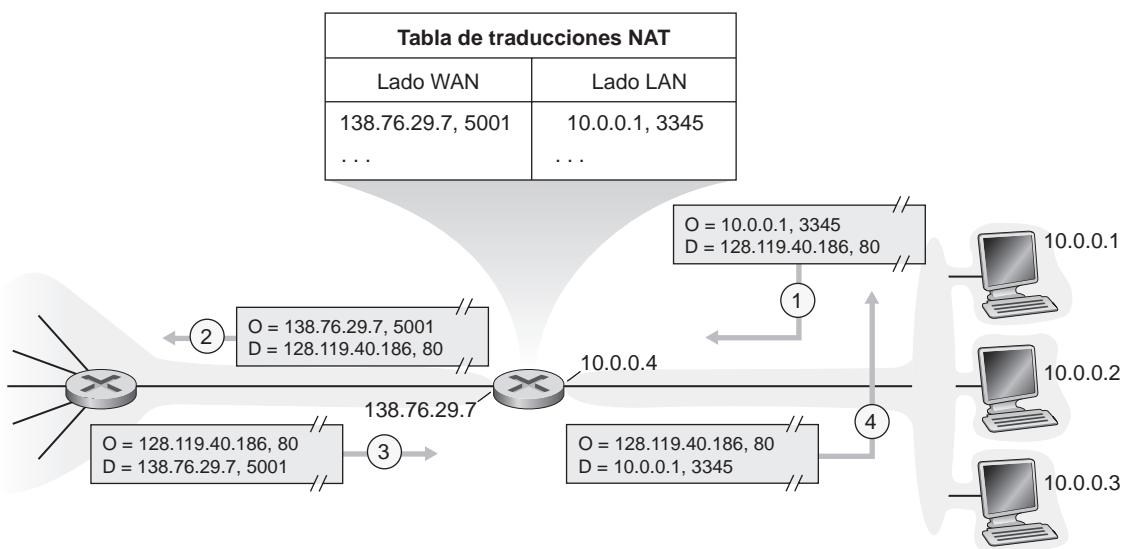


Figura 4.25 ♦ Traducción de direcciones de red.

En resumen, el router NAT oculta los detalles de la red doméstica al mundo exterior. (Como nota al margen, posiblemente se esté preguntando dónde obtienen las computadoras de la red doméstica sus direcciones y dónde obtiene el router su dirección IP única. A menudo, la respuesta a ambas preguntas es la misma: ¡DHCP! El router obtiene su dirección del servidor DHCP del ISP y el router ejecuta un servidor DHCP para proporcionar direcciones a las computadoras, dentro del espacio de direcciones de la red doméstica controlado por el router NAT-DHCP.)

Si todos los datagramas que llegan al router NAT procedentes de la WAN tienen la misma dirección IP de destino (específicamente, la de la interfaz WAN del router NAT), entonces ¿cómo sabe el router a qué host interno debería reenviar un datagrama dado? El truco consiste en utilizar una **tabla de traducciones NAT** almacenada en el router NAT, e incluir números de puerto junto con las direcciones IP en las entradas de la tabla.

Consideré el ejemplo de la Figura 4.25. Suponga que un usuario de una red doméstica que utiliza el host con la dirección 10.0.0.1 solicita una página web almacenada en un servidor web (puerto 80) con la dirección IP 128.119.40.186. El host 10.0.0.1 asigna el número de puerto de origen (arbitrario) 3345 y envía el datagrama a la LAN. El router NAT recibe el datagrama, genera un nuevo número de puerto de origen, 5001, para el datagrama, sustituye la dirección IP de origen por su propia dirección IP de la red WAN 138.76.29.7, y sustituye el número de puerto de origen original 3345 por el nuevo número de puerto de origen 5001. Al generar un nuevo número de puerto de origen, el router NAT puede seleccionar cualquier número de puerto de origen que no se encuentre actualmente en la tabla de traducciones NAT. (¡Observe que, puesto que la longitud del campo de número de puerto es de 16 bits, el protocolo NAT puede dar soporte a más de 60.000 conexiones simultáneas utilizando una única dirección IP para el lado WAN del router!) En el router, NAT también añade una entrada a su tabla de traducciones. El servidor web, que afortunadamente no es consciente de que el datagrama entrante que contiene la solicitud HTTP ha sido manipulado por el router NAT, responde con un datagrama cuya dirección de destino es la dirección IP del router NAT y cuyo número de puerto de destino es 5001. Cuando este datagrama llega al router NAT, éste indexa la tabla de traducciones NAT utilizando la dirección IP de destino y el número de puerto de destino para obtener la dirección IP (10.0.0.1) y el número de puerto de destino (3345) apropiados para el navegador de la red doméstica. A continuación, el router reescribe la dirección de destino y el número de puerto de destino del datagrama y lo reenvía a la red doméstica.

NAT ha disfrutado de una gran difusión en los últimos años, aunque no carece de detractores. En primer lugar, se podría argumentar que los números de puerto están pensados para direccionar procesos, no para direccionar hosts. De hecho, esta violación puede causar problemas en los servidores que se estén ejecutando en la red doméstica, ya que, como hemos visto en el Capítulo 2, los procesos de servidor están a la espera de las solicitudes entrantes en números de puerto bien conocidos; y los homólogos en un protocolo P2P necesitan aceptar conexiones entrantes cuando actúan como servidores. Entre las soluciones técnicas a estos problemas están las herramientas de **NAT traversal** [RFC 5389], así como UPnP (*Universal Plug and Play*), un protocolo que permite a un host descubrir y configurar un NAT cercano [UPnP Forum 2016].

Los puristas de la arquitectura han planteado también argumentos más “filosóficos” contra NAT. Lo que les preocupa es que los routers están pensados como dispositivos de capa 3 (es decir, de la capa de red) y solo deberían procesar paquetes hasta la capa de red. NAT viola este principio de que los hosts deberían hablar directamente unos con otros, sin que nodos intermedios modifiquen las direcciones IP, y mucho menos los números de puerto. Pero, guste o no, NAT se ha convertido en un componente importante de Internet, al igual que otros tipos de **dispositivos intermediarios** (*middleboxes*) [Sekar 2011] que operan en la capa de red, pero tienen funciones muy diferentes de las de los routers. Los dispositivos intermediarios no llevan a cabo el reenvío tradicional de datagramas, sino que en su lugar realizan funciones como NAT, equilibrado de carga de los flujos de tráfico, cortafuegos (véase el recuadro Seguridad adjunto) y otras. El paradigma de reenvío generalizado, que en breve estudiaremos en la Sección 4.4, hace posible llevar a cabo varias de estas funciones intermediarias y el reenvío tradicional de los routers, de una forma integrada y común.

SEGURIDAD

INSPECCIÓN DE DATAGRAMAS: CORTAFUEGOS Y SISTEMAS DE DETECCIÓN DE INTRUSIONES

Suponga que le han asignado la tarea de administrar una red doméstica, departamental, universitaria o corporativa. Los atacantes, que saben cuál es el rango de direcciones IP de su red, pueden enviar fácilmente datagramas IP a las direcciones de ese rango. Estos datagramas pueden hacer toda clase de cosas retorcidas, incluyendo confeccionar mapas de la red mediante barridos de ping y escaneo de puertos; dañar los hosts vulnerables con paquetes erróneos; hacer barridos para detectar puertos TCP/UDP abiertos en los servidores de nuestra red e infectar los hosts incluyendo software malicioso en los paquetes. Como administrador de la red, ¿qué hará con todos esos malvados capaces de enviar paquetes maliciosos a su red? Hay disponibles dos mecanismos de defensa muy populares contra los ataques de paquetes maliciosos: los cortafuegos y los sistemas de detección de intrusiones (IDS, *Intrusion Detection System*).

Como administrador de la red, primero puede probar a instalar un cortafuegos entre su red e Internet. (La mayoría de los routers actuales de acceso disponen de cortafuegos.) Los cortafuegos inspeccionan los campos de cabecera de los segmentos y datagramas, denegando el acceso a la red interna a los datagramas sospechosos. Por ejemplo, un cortafuegos puede configurarse para bloquear todos los paquetes de solicitud de eco ICMP (véase la Sección 5.6), impidiendo así que un atacante lleve a cabo un tradicional barrido de puertos a lo largo de su rango de direcciones IP. Los cortafuegos también pueden bloquear paquetes basándose en las direcciones IP y números de puerto de origen y de destino. Además, los cortafuegos se pueden configurar para controlar las conexiones TCP, dejando entrar sólo a los datagramas que pertenezcan a conexiones aprobadas.

Con un sistema IDS puede proporcionarse protección adicional. Un IDS, colocado normalmente en la frontera de la red, realiza una "inspección profunda de los paquetes", examinando no sólo los campos de cabecera sino también las cargas útiles de los datagramas (incluyendo los datos de la capa de aplicación). Un IDS dispone de una base de datos de firmas de paquetes que se sabe que forman parte de ataques. Esta base de datos se actualiza automáticamente a medida que se descubren nuevos tipos de ataque. A medida que los paquetes atraviesan el sistema IDS, éste intenta encontrar una coincidencia entre los campos de cabecera o las cargas útiles y las firmas que almacena en su base de datos. Si encuentra una coincidencia, genera una alerta. Un sistema de prevención de intrusiones (IPS, *Intrusion Prevention System*) es similar a un sistema IDS, salvo porque además de generar una alerta, bloquea los paquetes. En el Capítulo 8 estudiaremos los cortafuegos y los sistemas IDS con más detalle.

¿Pueden los cortafuegos y los sistemas IDS proteger completamente a la red frente a todos los ataques? Evidentemente, la respuesta es no, ya que los atacantes encuentran continuamente nuevos ataques para los que las firmas todavía no están disponibles. No obstante, los cortafuegos y los IDS tradicionales basados en firmas son útiles para proteger a las redes de los ataques conocidos.

4.3.5 IPv6

A principios de la década de 1990, el Internet Engineering Task Force comenzó a desarrollar un sucesor para el protocolo IPv4. La principal motivación de esta iniciativa fue que se dieron cuenta de que el espacio de direcciones IP de 32 bits estaba comenzando a agotarse, a causa de la velocidad pasmosa con que estaban conectándose a Internet nuevas subredes y nodos IP (a los que se les asignaban direcciones IP unívocas). Para responder a esta necesidad de un espacio de direcciones IP más grande, se desarrolló un nuevo protocolo IP, el protocolo IPv6. Los diseñadores de IPv6 también aprovecharon la oportunidad para ajustar y expandir otros aspectos de IPv4, basándose en la experiencia acumulada sobre el funcionamiento de dicho protocolo.

El momento en que las direcciones IPv4 se agotarían (y por tanto ninguna nueva subred podría conectarse a Internet) fue objeto de un importante debate. Las estimaciones de los dos

líderes del grupo de trabajo *Address Lifetime Expectations* (Expectativas del tiempo de vida de las direcciones) del IETF fueron que las direcciones se agotarían en 2008 y 2018, respectivamente [Solensky 1996]. En febrero de 2011, la IANA asignó a un registro regional el último grupo de direcciones IPv4 que quedaba sin asignar. Aunque los registros regionales aun disponen de direcciones IPv4 sin asignar dentro de sus respectivos bloques, una vez que se agoten esas direcciones ya no habrá más bloques de direcciones disponibles que puedan asignarse desde ninguna autoridad central [Huston 2011a]. En [Richter 2015] puede encontrar un repaso reciente del tema del agotamiento del espacio de direcciones IPv4, así como de los pasos tomados para prolongar la vida del espacio de direcciones.

Aunque las estimaciones sobre el agotamiento de las direcciones IPv4 realizadas a mediados de la década de 1990 sugerían que quedaba bastante tiempo para que el espacio de direcciones de IPv4 se agotara, se hizo evidente que se necesitaría un tiempo considerable para implantar una nueva tecnología a tan gran escala, y por eso se comenzó a trabajar [RFC 1752] en el desarrollo de la versión 6 de IP (IPv6) [RFC 2460]. (Una pregunta que se plantea a menudo es qué ocurrió con IPv5. Inicialmente se pensó que el protocolo ST-2 se convertiría en IPv5, pero ST-2 fue descartado más tarde.) Una excelente fuente de información sobre IPv6 es [Huitema 1998].

Formato del datagrama IPv6

El formato del datagrama IPv6 se muestra en la Figura 4.26. Los cambios más importantes introducidos en IPv6 son evidentes en el formato de su datagrama:

- *Capacidades ampliadas de direccionamiento.* IPv6 aumenta el tamaño de la dirección IP, de 32 a 128 bits. De esta manera, se asegura que el mundo no se quedará sin direcciones IP. Ahora, cada grano de arena del planeta podría tener asignada una dirección IP. Además de las direcciones de unidifusión y de multidifusión, IPv6 ha introducido un nuevo tipo de dirección, denominado **dirección anycast**, que permite entregar un datagrama a uno cualquiera de un grupo de hosts. (Esta funcionalidad podría utilizarse, por ejemplo, para enviar un mensaje GET HTTP al más próximo de una serie de sitios espejo que contengan un determinado documento.)
- *Una cabecera de 40 bytes simplificada.* Como se explica más adelante, algunos de los campos de IPv4 se han eliminado o se han hecho opcionales. La cabecera resultante, con una longitud fija de 40 bytes, permite un procesamiento más rápido del datagrama IP en los routers. Una nueva codificación de las opciones permite un procesamiento más flexible de las mismas.
- *Etiquetado del flujo.* IPv6 utiliza una definición bastante amplia de **flujo**. El documento RFC 2460 establece que esto permite “etiquetar los paquetes que pertenecen a determinados flujos para los que el emisor solicita un tratamiento especial, como una calidad de servicio no prede-

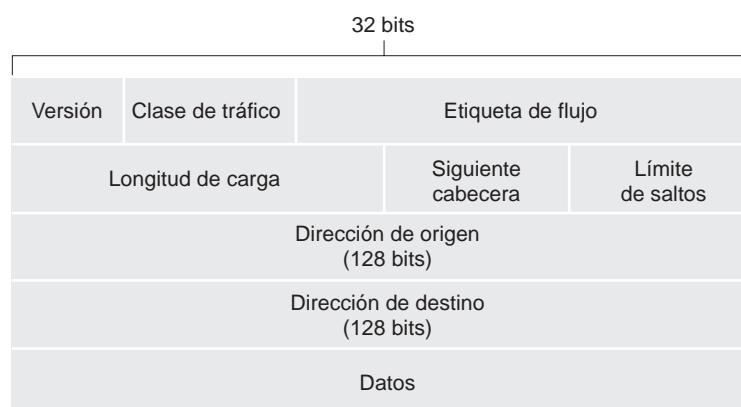


Figura 4.26 ♦ Formato del datagrama IPv6.

terminada o un servicio en tiempo real". Por ejemplo, la transmisión de audio y de vídeo puede posiblemente tratarse como un flujo. Por el contrario, aplicaciones más tradicionales, como la transferencia de archivos y el correo electrónico, podrían no tratarse como flujos. Es posible que el tráfico transportado por un usuario de alta prioridad (por ejemplo, alguien que paga por obtener un mejor servicio para su tráfico) pueda ser tratado también como un flujo. Sin embargo, lo que está claro es que los diseñadores de IPv6 previeron la eventual necesidad de poder diferenciar entre los flujos, incluso aunque el significado exacto de flujo no hubiera sido determinado aún.

Como se ha mencionado anteriormente, una comparación de la Figura 4.26 con la Figura 4.16 revela la estructura más simple y estilizada del datagrama de IPv6. En IPv6 se definen los siguientes campos:

- *Versión*. Este campo de 4 bits identifica el número de versión IP. Nada sorprendentemente, IPv6 transporta un valor de 6 en este campo. Observe que incluir en este campo el valor 4 no implica que se cree un datagrama IPv4 válido. (Si lo hiciera, la vida sería mucho más simple —véase más adelante la exposición sobre la transición de IPv4 a IPv6—).
- *Clase de tráfico*. Al igual que el campo TOS de IPv4, el campo de clase de tráfico, de 8 bits, puede utilizarse para dar prioridad a ciertos datagramas dentro de un flujo, o para dar prioridad a los datagramas de determinadas aplicaciones (por ejemplo, Voz sobre IP) frente a los datagramas de otras aplicaciones (por ejemplo, correo electrónico SMTP).
- *Etiqueta de flujo*. Como hemos mencionado anteriormente, este campo de 20 bits se utiliza para identificar un flujo de datagramas.
- *Longitud de la carga útil*. Este valor de 16 bits se trata como un entero sin signo que proporciona el número de bytes del datagrama IPv6 incluidos a continuación de la cabecera del datagrama, que es de 40 bytes y tiene longitud fija.
- *Siguiente cabecera*. Este campo identifica el protocolo (por ejemplo, TCP o UDP) al que se entregará el contenido (el campo de datos) de este datagrama. El campo utiliza los mismos valores que el campo de protocolo de la cabecera IPv4.
- *Límite de saltos*. Cada router que reenvía un datagrama decrementa el contenido de este campo en una unidad. Si el límite de saltos alcanza el valor cero, el datagrama se descarta.
- *Direcciones de origen y de destino*. Los distintos formatos de la dirección de 128 bits de IPv6 se describen en el documento RFC 4291.
- *Datos*. Esta es la parte de la carga útil del datagrama IPv6. Cuando el datagrama llegue a su destino, la carga útil se extraerá del datagrama IP y se pasará al protocolo especificado en el campo Siguiente cabecera.

Hasta aquí hemos identificado el propósito de los campos incluidos en los datagramas IPv6. Comparando el formato del datagrama IPv6 de la Figura 4.26 con el formato del datagrama IPv4 que hemos visto en la Figura 4.16, podemos observar que varios campos del datagrama IPv4 ya no aparecen en IPv6:

- *Fragmentación/Reensamblado*. IPv6 no permite ni la fragmentación ni el reensamblado en routers intermedios; estas operaciones solo pueden ser realizadas por el origen y el destino. Si un router recibe un datagrama IPv6 y es demasiado largo para ser reenviado por el enlace de salida, el router simplemente lo descarta y envía de vuelta al emisor un mensaje de error ICMP "Paquete demasiado grande" (véase la Sección 5.6). El emisor puede entonces reenviar los datos utilizando un tamaño de datagrama IP más pequeño. La fragmentación y el reensamblado son operaciones que consumen mucho tiempo, por lo que eliminando esta funcionalidad de los routers e incluyéndola directamente en los sistemas terminales se acelera considerablemente el reenvío IP dentro de la red.

- *Suma de comprobación de cabecera.* Puesto que los protocolos de la capa de transporte (por ejemplo, TCP y UDP) y de la capa de enlace de datos (por ejemplo, Ethernet) de Internet utilizan sumas de comprobación, los diseñadores de IP probablemente pensaron que esta funcionalidad ya era suficientemente redundante en la capa de red y podía eliminarse. Una vez más, el procesamiento rápido de los paquetes IP era la preocupación principal. Recuerde de la Sección 4.3.1 que, como la cabecera de IPv4 contiene un campo TTL (similar al campo de límite de saltos de IPv6), la suma de comprobación de la cabecera IPv4 necesitaba ser recalculada en cada router. Al igual que la fragmentación y el reensamblado, esta también era una operación muy costosa en IPv4.
- *Opciones.* La cabecera IP estándar ya no incluye un campo de opciones. Sin embargo, las opciones no han desaparecido. En su lugar, el campo de opciones es una de las posibles siguientes cabeceras a las que se apunta desde dentro de la cabecera IPv6. Es decir, al igual que las cabeceras de los protocolos TCP o UDP pueden ser la siguiente cabecera dentro de un paquete IP, también puede serlo un campo de opciones. La eliminación del campo de opciones dio como resultado una cabecera IP de 40 bytes de longitud fija.

Transición de IPv4 a IPv6

Ahora que hemos visto los detalles técnicos de IPv6, vamos a considerar una cuestión práctica: ¿cómo va a hacer Internet, que está basada en IPv4, la transición a IPv6? El problema está en que, mientras que los nuevos sistemas para IPv6 pueden hacerse compatibles en sentido descendente, es decir, pueden enviar, enrutar y recibir datagramas IPv4, los sistemas para IPv4 ya implantados no son capaces de manejar datagramas IPv6. Existen varias posibles soluciones [Huston 201b, RFC 4213].

Una opción sería declarar un día D: una fecha y hora en la que todas las máquinas conectadas a Internet se apagaran y actualizaran de IPv4 a IPv6. La última transición importante de una tecnología a otra (de NCP a TCP como servicio de transporte fiable) tuvo lugar hace casi 35 años. Incluso entonces [RFC 801], cuando Internet era pequeña y todavía era administrada por un número pequeño de “gurús”, se dieron cuenta de que establecer un día D no era posible. Con mayor razón, un día D que implicara a miles de millones de dispositivos sería aún más impensable hoy en día.

La solución más ampliamente adoptada en la práctica para la transición de IPv4 a IPv6 es la de **tunelización** [RFC 4213]. La idea básica en la que descansa el método de tunelización (un concepto clave de las aplicaciones en muchos otros escenarios, más allá de la transición de IPv4 a IPv6, incluyendo su uso generalizado en las redes móviles IP de las que hablaremos en el Capítulo 7) es la siguiente: suponga que dos nodos IPv6 (por ejemplo los nodos B y E de la Figura 4.27) desean interoperar utilizando datagramas IPv6, pero están conectados entre sí a través de routers IPv4. Al conjunto de routers intermedios IPv4 existentes entre dos routers IPv6 lo denominaremos **túnel**, como se ilustra en la Figura 4.27. Mediante la tunelización, el nodo IPv6 del lado emisor del túnel (en este ejemplo, B) toma el datagrama IPv6 *completo* y lo incluye en el campo de datos (carga útil) de un datagrama IPv4. Este datagrama IPv4 se direcciona entonces hacia el nodo IPv6 del lado receptor del túnel (en este ejemplo, E) y se envía al primer nodo del túnel (en este ejemplo, C). Los routers intermedios IPv4 del túnel enrutan este datagrama IPv4 entre ellos mismos, como si fuera cualquier otro datagrama, siendo totalmente inconscientes de que el datagrama IPv4 contiene un datagrama IPv6 completo. El nodo IPv6 del lado de recepción del túnel termina recibiendo el datagrama IPv4 (¡es el destino del datagrama IPv4!), determina que el datagrama IPv4 contiene un datagrama IPv6 (viendo que el campo de número de protocolo del datagrama IPv4 es 41 [RFC 4213]), lo que indica que la carga útil IPv4 es un datagrama IPv6), extrae el datagrama IPv6 y, a continuación, lo enruta exactamente igual que si hubiera recibido el datagrama IPv6 desde un vecino IPv6 directamente conectado.

Para terminar esta sección, diremos que aunque la adopción de IPv6 inicialmente fue lenta [Lawton 2001; Huston 2008b], recientemente está empezando a tomar impulso. NIST [NIST

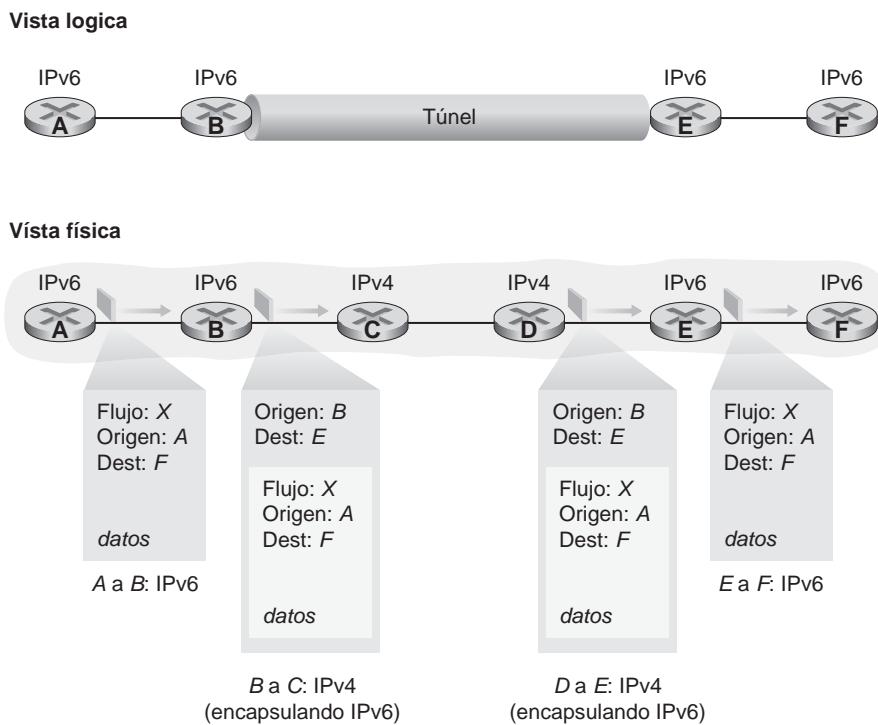


Figura 4.27 ♦ Tunelización.

IPv6 2015] informa de que más de un tercio de los dominios de segundo nivel del gobierno de los EE.UU. son compatibles con IPv6. En el lado de los clientes, Google afirma que solo un 8%, aproximadamente, de los clientes que acceden a los servicios Google lo hacen a través de IPv6 [Google IPv6 2015]. Pero otras medidas recientes [Czyz 2014] indican que la adopción de IPv6 se está acelerando. La proliferación de dispositivos como teléfonos compatibles con IP y otros dispositivos portátiles proporciona un empuje adicional para una más amplia implantación de IPv6. El programa europeo Third Generation Partnership Program [3GPP 2016] ha especificado IPv6 como esquema estándar de direccionamiento para multimedia móvil.

Una lección importante que podemos aprender de la experiencia de IPv6 es que resulta enormemente complicado cambiar los protocolos de la capa de red. Desde principios de la década de 1990, se han anunciado numerosos nuevos protocolos de la capa de red como la siguiente revolución más importante de Internet, pero la mayoría de estos protocolos han tenido una penetración limitada hasta la fecha. Entre estos protocolos se incluyen IPv6, los protocolos de multidifusión y los protocolos de reserva de recursos; en el suplemento en línea de este libro hay una explicación de estos dos últimos protocolos. De hecho, la introducción de nuevos protocolos en la capa de red es como sustituir los cimientos de una casa: resulta difícil de hacer sin tirar abajo toda la casa o al menos reubicar temporalmente a sus habitantes. Por otro lado, Internet ha sido testigo de la rápida implantación de nuevos protocolos en la capa de aplicación. Ejemplos clásicos de esto son, por supuesto, la Web, la mensajería instantánea, los flujos multimedia, los juegos distribuidos y diversos tipos de redes sociales. Introducir nuevos protocolos de la capa de aplicación es como añadir una nueva capa de pintura a las paredes de una casa: es relativamente fácil de hacer y, si se elige un color atractivo, otras personas del vecindario nos copiarán. En resumen, en el futuro podemos esperar ver cambios en la capa de red de Internet, por supuesto que sí, pero probablemente esos cambios se producirán en una escala temporal mucho más lenta que los cambios que tendrán lugar en la capa de aplicación.

4.4 Reenvío generalizado y SDN

En la Sección 4.2.1 hemos comentado que las decisiones de reenvío de un router de Internet han estado basadas tradicionalmente tan solo en la dirección de destino de los paquetes. En la sección anterior, sin embargo, hemos visto también que se ha producido una proliferación de dispositivos intermedios que realizan muchas funciones de la capa 3. Los dispositivos NAT reescriben las direcciones IP y los números de puerto de las cabeceras de los datagramas; los cortafuegos bloquean el tráfico basándose en los valores de los campos de cabecera o redirigen los paquetes para un procesamiento adicional, como por ejemplo una inspección profunda de paquetes (DPI, *Deep Packet Inspection*). Los dispositivos de equilibrado de carga reenvían los paquetes que solicitan un determinado servicio (por ejemplo, solicitudes HTTP) a uno entre un conjunto de servidores que proporcionan dicho servicio. [RFC 3234] enumera diversas funciones intermedias comunes.

Esta proliferación de dispositivos intermedios, switches de capa 2 y routers de capa 3 [Qazi 2013] —cada uno con sus propios hardware, software e interfaces de administración especializados— ha resultado, sin lugar a dudas, en costosos dolores de cabeza para muchos operadores de red. Sin embargo, los recientes avances en redes definidas por software prometían, y están ahora cumpliendo, proporcionar un enfoque unificado para la realización de muchas de estas funciones de la capa de red, así como de ciertas funciones de la capa de enlace, de una manera moderna, elegante e integrada.

Recuerde que en la Sección 4.2.1 caracterizamos el reenvío basado en el destino como un proceso en dos pasos, consistente en buscar una dirección IP de destino (“correspondencia”) y luego enviar el paquete hacia el puerto de salida especificado, a través del entramado de commutación (“acción”). Vamos a considerar ahora un paradigma “correspondencia más acción” bastante más general, en el que la “correspondencia” puede buscarse para múltiples campos de cabecera asociados con diferentes protocolos en diferentes capas de la pila de protocolos. La “acción” puede consistir en reenviar el paquete a uno o más puertos de salida (como en el caso del reenvío basado en el destino), en enviar los paquetes a una u otra de entre múltiples interfaces salientes que conducen a un cierto servicio (como en el equilibrado de carga), en reescribir valores de la cabecera (como en NAT), en bloquear/eliminar un paquete deliberadamente (como en un cortafuegos); en enviar el paquete a un servidor especial para procesamiento y acciones adicionales (como en DPI) u otras acciones.

En el reenvío generalizado, una tabla de correspondencia más acción generaliza la noción de tabla de reenvío basada en el destino que ya hemos visto en la Sección 4.2.1. Puesto que las decisiones de reenvío pueden tomarse basándose en las direcciones de origen y de destino de la capa de red y/o de la capa de enlace, los dispositivos de reenvío mostrados en la Figura 4.28 se podrían describir con más precisión como “comutadores de paquetes”, más que como “routers” de la capa 3 o “switches” de la capa 2. Por ello, en el resto de esta sección, y en la Sección 5.5, denominaremos a estos dispositivos comutadores de paquetes, adoptando la terminología que está consiguiendo una amplia aceptación en la literatura sobre SDN.

La Figura 4.28 muestra una tabla correspondencia-acción (*match plus action*) en cada comutador de paquetes, siendo dicha tabla calculada, instalada y actualizada por un controlador remoto. Hay que resaltar que, aunque resulta posible para los componentes de control de cada comutador de paquetes individual interactuar entre sí (por ejemplo, de forma similar a la de la Figura 4.2), en la práctica la funcionalidad generalizada correspondencia-acción se implementa mediante un controlador remoto que calcula, instala y actualiza esas tablas. Tómese un momento para comparar las figuras 4.2, 4.3 y 4.28. ¿Qué similitudes y diferencias encuentra entre el reenvío basado en el destino, ilustrado en las Figuras 4.2 y 4.3, y el reenvío generalizado mostrado en la Figura 4.28?

Nuestra siguiente exposición sobre el reenvío generalizado está basada en OpenFlow [McKeown 2008, OpenFlow 2008, Casado 2014, Tourrilhes 2014], un estándar muy conocido y aceptado que ha sido pionero en lo que respecta a la abstracción de reenvío correspondencia-acción y sus controladores, así como en lo que respecta a la revolución SDN en general [Fearnster 2013]. Nos centraremos principalmente en OpenFlow 1.0, que introdujo importantes abstracciones y funcionalidad SDN

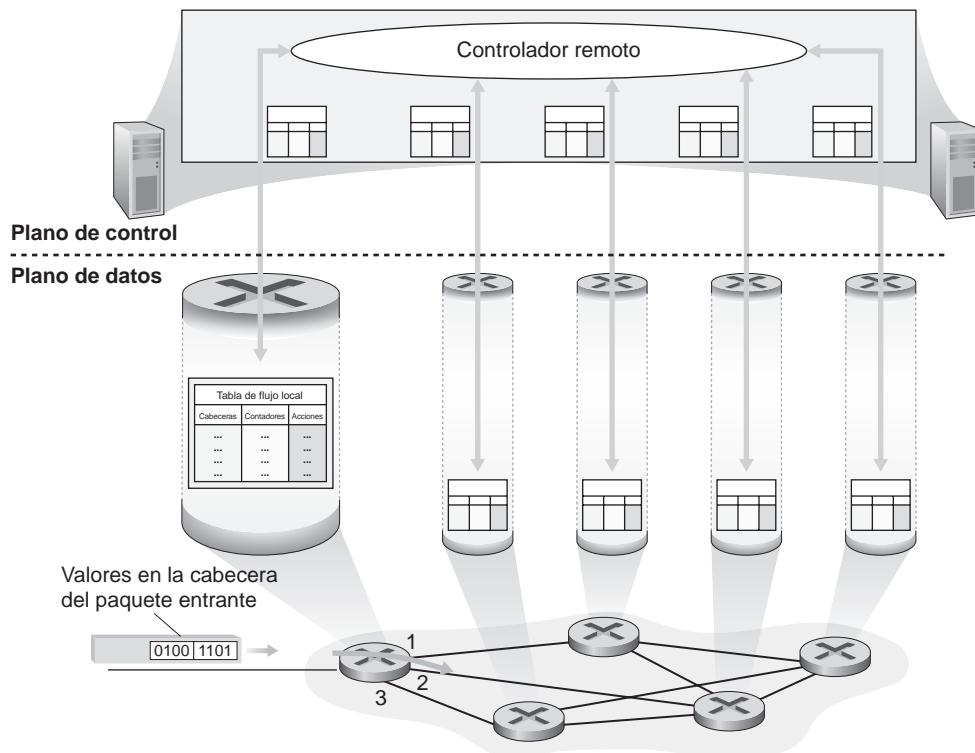


Figura 4.28 ♦ Reenvío generalizado: cada commutador de paquetes contiene una tabla de correspondencia-acción que es calculada y distribuida por un controlador remoto.

de una forma particularmente clara y concisa. Versiones posteriores de OpenFlow introdujeron capacidades adicionales como resultado de la experiencia ganada con la implementación y el uso; puede encontrar las versiones actual y anteriores del estándar OpenFlow en [ONF 2016].

Cada entrada de la tabla de reenvío correspondencia-acción, que se conoce con el nombre de **tabla de flujo** en OpenFlow, incluye:

- *Un conjunto de valores de campos de cabecera* con los que se buscará una correspondencia en el paquete entrante. Al igual que en el caso del reenvío basado en el destino, la búsqueda de correspondencias basada en hardware puede llevarse a cabo con la máxima velocidad usando memorias TCAM, siendo posible disponer de más de un millón de entradas de dirección de destino [Bosshart 2013]. Un paquete que no se corresponda con ninguna entrada de la tabla de flujo puede ser eliminado o enviado al controlador remoto para un procesamiento adicional. En la práctica, una tabla de flujo puede ser implementada mediante múltiples tablas de flujo, por razones de coste o de prestaciones [Bosshart 2013], pero aquí vamos a centrarnos en la abstracción de una única tabla de flujo.
- *Un conjunto de contadores* que se actualizan a medida que se encuentran correspondencias de paquetes con entradas de la tabla de flujo. Estos contadores podrían incluir el número de paquetes para los que se ha encontrado correspondencia con la entrada de la tabla y el tiempo transcurrido desde que la entrada de la tabla se actualizó por última vez
- *Un conjunto de acciones que hay que tomar* cuando un paquete se corresponde con una entrada de la tabla de flujo. Estas acciones podrían consistir en reenviar el paquete a un puerto de salida determinado, eliminar el paquete, hacer copias del paquete y enviarlas a múltiples puertos de salida y/o reescribir ciertos campos seleccionados de la cabecera.

Exploraremos las correspondencias y las acciones con más detalle en las secciones 4.4.1 y 4.4.2, respectivamente. Después, en la Sección 4.4.3, veremos el modo en que puede emplearse el conjunto de reglas de correspondencia de paquetes existentes en la red para implementar un amplio rango de funciones, incluyendo el enrutamiento, la conmutación de capa 2, los cortafuegos, el equilibrado de carga, las redes virtuales y otras. Para concluir, resaltaremos que la tabla de flujo es, esencialmente, una API: la abstracción mediante la cual puede programarse el comportamiento de un conmutador de paquetes individual; veremos en la Sección 4.4.3 que se pueden programar de forma similar comportamientos de la red completa, programando/configurando apropiadamente estas tablas en un conjunto de conmutadores de paquetes de la red [Casado 2014].

4.4.1 Correspondencia

La Figura 4.29 muestra los once campos de cabecera de los paquetes, más la ID del puerto entrante, con los que puede buscarse una correspondencia en una regla correspondencia-acción de OpenFlow 1.0. Recuerde de la Sección 1.5.2 que una trama de la capa de enlace (capa 2) que llegue a un conmutador de paquetes, contendrá como carga útil un datagrama de la capa de red (capa 3), que a su vez contendrá, normalmente, un segmento de la capa de transporte (capa 4). La primera observación que conviene hacer es que la abstracción correspondencia de OpenFlow permite buscar una correspondencia con campos seleccionados de tres capas de cabeceras de protocolo (violando así descaradamente el principio de separación en capas que hemos estudiado en la Sección 1.5). Puesto que todavía no hemos estudiado la capa de enlace, baste decir que las direcciones MAC de origen y de destino mostradas en la Figura 4.29 son las direcciones de la capa de enlace asociadas con las interfaces emisora y receptora de la trama; al poder reenviar basándose en las direcciones Ethernet, en lugar de en las direcciones IP, vemos que un dispositivo compatible con OpenFlow puede comportarse tanto como un router (dispositivo de capa 3) que reenvía datagramas, cuanto como un switch (dispositivo de capa 2) que reenvía tramas. El campo del tipo Ethernet se corresponde con el protocolo de la capa superior (por ejemplo, IP) hacia el que se demultiplexará la carga útil de la trama, mientras que los campos VLAN están relacionados con las denominadas redes de área local virtuales, que estudiaremos en el Capítulo 6. El conjunto de doce valores con los que se pueden buscar correspondencias en la especificación OpenFlow 1.0 ha crecido hasta 41 valores en las especificaciones OpenFlow más recientes [Bosschart 2014].

El puerto de ingreso hace referencia al puerto de entrada del conmutador de paquetes a través del cual se recibe un paquete. De los campos de dirección IP de origen, dirección IP de destino, protocolo IP y tipo de servicio (TOS) IP ya hemos hablado anteriormente, en la Sección 4.3.1. También pueden buscarse correspondencias con los campos de número de puerto de origen y de destino de la capa de transporte.

Las entradas de la tabla de flujo también pueden tener caracteres comodín. Por ejemplo, una dirección IP de 128.119.*.* en una tabla de flujo se corresponderá con todo campo de dirección relevante de cualquier datagrama que tenga como primeros 16 bits de la dirección el valor 128.119. Cada entrada de la tabla de flujo tiene también asociada una prioridad. Si un paquete se corresponde con múltiples entradas de la tabla de flujo, la correspondencia y la acción correspondiente seleccionadas serán las de la entrada de mayor prioridad con la que el paquete se corresponda.

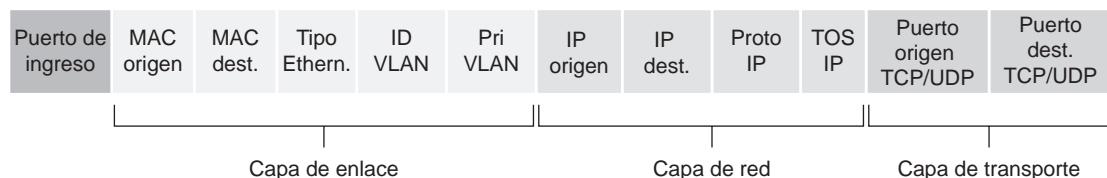


Figura 4.29 ♦ Campos de correspondencia de los paquetes en la tabla de flujo de OpenFlow 1.0.

Por último, observe que no pueden establecerse correspondencias con todos los campos de una cabecera IP. Por ejemplo, OpenFlow no permite buscar correspondencias con el campo TTL ni con el campo de longitud del datagrama. ¿Por qué se permite buscar correspondencias con algunos campos y con otros no? Indudablemente, la respuesta tiene que ver con el compromiso entre funcionalidad y complejidad. El “arte” a la hora de seleccionar una abstracción consiste en proporcionar la suficiente funcionalidad como para llevar a cabo una tarea (en este caso, implementar, configurar y administrar una amplia gama de funciones de la capa de red, que antes se implementaban mediante una variedad de dispositivos de la capa red), pero sin sobrecargar la abstracción con tantos detalles y tanta generalidad que la abstracción se expanda demasiado y resulte inutilizable. Como dice la famosa frase de Butler Lampson [Lampson 1983]:

Haz una cosa cada vez y hazla bien. Una interfaz debe capturar los detalles mínimos esenciales de una abstracción. No generalices; las generalizaciones son generalmente erróneas.

Dado el éxito de OpenFlow, podemos dar por supuesto que sus diseñadores seleccionaron bien su abstracción. Puede encontrar detalles adicionales sobre las correspondencias en OpenFlow en [OpenFlow 2009, ONF 2016].

4.4.2 Acción

Como se muestra en la Figura 4.28, cada entrada de la tabla de flujo tiene una lista de cero o más acciones, que determinan el procesamiento que hay que aplicar a un paquete que se corresponda con dicha entrada. Si hay múltiples acciones, se ejecutan en el orden especificado en la lista.

Las más importantes de las acciones posibles son:

- *Reenvío.* Un paquete entrante puede ser reenviado a un puerto físico de salida concreto; puede ser difundido a través de todos los puertos (excepto aquél por el que haya llegado) o puede ser enviado a través de un conjunto seleccionado de puertos. El paquete también puede ser encapsulado y enviado al controlador remoto correspondiente a este dispositivo. El controlador puede entonces (o no) llevar a cabo alguna acción sobre dicho paquete, incluyendo la instalación de nuevas entradas de la tabla de flujo, y puede devolver el paquete al dispositivo, para que lo reenvíe de acuerdo con el conjunto de reglas actualizado de la tabla de flujo.
- *Eliminación.* Una entrada de la tabla de flujo sin ninguna acción indica que un paquete que se corresponda con ella debe ser eliminado.
- *Modificación de campos.* Los valores de diez campos de cabecera del paquete (todos los campos de las capas 2, 3 y 4 mostrados en la Figura 4.29, con excepción del campo Protocolo IP) pueden ser reescritos antes de reenviar el paquete al puerto de salida elegido.

4.4.3 Ejemplos de correspondencia-acción en OpenFlow

Habiendo repasado los componentes de correspondencia y acción del reenvío generalizado, juntamos estos conceptos en el contexto de la red de ejemplo mostrada en la Figura 4.30. La red tiene 6 hosts (h1, h2, h3, h4, h5 y h6) y tres comutadores de paquetes (s1, s2 y s3), con cuatro interfaces locales cada uno (numeradas de 1 a 4). Vamos a considerar diversos comportamientos de la red que nos gustaría implementar, así como las entradas de tabla de flujo que s1, s2 y s3 necesitarían para implementar cada uno de esos comportamientos.

Un primer ejemplo: reenvío simple

Como ejemplo sencillo, supongamos que el comportamiento de reenvío deseado consiste en que los paquetes de h5 o h6 destinados a h3 o h4 deben reenviarse de s3 a s1 y luego de s1 a s2 (evitando así completamente el uso del enlace existente entre s3 y s2). La entrada de la tabla de flujo en s1 sería:

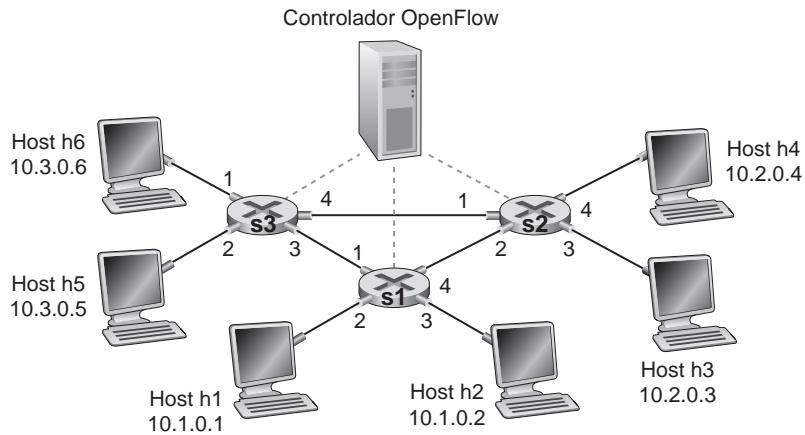


Figura 4.30 ♦ Red correspondencia-acción de OpenFlow con tres comutadores de paquetes, 6 hosts y un controlador OpenFlow.

Tabla de flujo de s1 (Ejemplo 1)	
Correspondencia	Acción
Puerto de ingreso = 1 ; Origen IP = 10.3.*.* ; Destino IP = 10.2.*.*	Reenvío(4)
...	...

Por supuesto, necesitaremos también una entrada de tabla de flujo en s3, de modo que los datagramas enviados desde h5 o h6 se reenvíen a s1 a través de la interfaz saliente 3:

Tabla de flujo de s3 (Ejemplo 1)	
Correspondencia	Acción
Origen IP = 10.3.*.* ; Destino IP = 10.2.*.*	Reenvío(3)
...	...

Por último, también necesitamos una entrada de tabla de flujo en s2 para completar este primer ejemplo, de modo que los datagramas que lleguen desde s1 se reenvíen a su destino, el host h3 o h4:

Tabla de flujo de s2 (Ejemplo 1)	
Correspondencia	Acción
Puerto de ingreso = 2 ; Destino IP = 10.2.0.3	Reenvío(3)
Puerto de ingreso = 2 ; Destino IP = 10.2.0.4	Reenvío(4)
...	...

Un segundo ejemplo: equilibrado de carga

Como segundo ejemplo, consideremos un escenario de equilibrado de carga, en el que los datagramas de h3 destinados a 10.1.*.* deben reenviarse a través del enlace directo entre s2 y s1, mientras que los datagramas de h4 destinados a 10.1.*.* deben reenviarse a través del enlace entre s2 y s3 (y luego de s3 a s1). Observe que este comportamiento no podría conseguirse mediante el reenvío basado en el destino IP. En este caso, la tabla de flujo en s2 sería:

Tabla de flujo de s2 (Ejemplo 2)	
Correspondencia	Acción
Puerto de ingreso = 3; Destino IP = 10.1.*.*	Reenvío(2)
Puerto de ingreso = 4; Destino IP = 10.1.*.*	Reenvío(1)
...	...

También hacen falta entradas de tabla de flujo en s1 para reenviar a h1 o h2 los datagramas recibidos de s2; y hacen falta entradas de tabla de flujo en s3 para reenviar hacia s1, a través de la interfaz 3, los datagramas recibidos de s2 a través de la interfaz 4. Trate de deducir cuáles deberían ser esas entradas de tabla de flujo en s1 y s3.

Un tercer ejemplo: cortafuegos

Como tercer ejemplo, consideremos un caso de cortafuegos en el que s2 solo quiere recibir (a través de cualquiera de sus interfaces) el tráfico enviado por los hosts conectados a s3.

Tabla de flujo de s2 (Ejemplo 3)	
Correspondencia	Acción
Origen IP = 10.3.*.* Destino IP = 10.2.0.3	Reenvío(3)
Origen IP = 10.3.*.* Destino IP = 10.2.0.4	Reenvío(4)
...	...

Si no hubiera ninguna otra entrada en la tabla de flujo de s2, entonces solo el tráfico procedente de 10.3.*.* sería reenviado a los hosts conectados a s2.

Aunque solo hemos considerado aquí unos pocos escenarios básicos, esperamos que hayan quedado claras la versatilidad y las ventajas del reenvío generalizado. En los problemas del capítulo, exploraremos cómo pueden usarse las tablas de flujo para crear muchos comportamientos lógicos distintos, incluyendo redes virtuales —dos o más redes separadas lógicamente (cada una con su propio comportamiento de reenvío distinto e independiente)— que usen el *mismo* conjunto físico de comutadores de paquetes y de enlaces. En la Sección 5.5, volveremos sobre las tablas de flujo cuando estudiemos los controladores SDN que las calculan y distribuyen, y el protocolo utilizado para la comunicación entre un comutador de paquetes y su controlador.

4.5 Resumen

En este capítulo hemos estudiado las funciones del **plano de datos** de la capa de red —las funciones de cada router que determinan cómo se reenvían a uno de los enlaces de salida del router los paquetes recibidos por uno de los enlaces de entrada del mismo. Hemos comenzado examinando detalladamente el funcionamiento interno del router, estudiando la funcionalidad de los puertos de entrada y de salida, el reenvío basado en el destino, el mecanismo interno de conmutación de un router y la gestión de las colas de paquetes, entre otros temas. Hemos analizado tanto del reenvío IP tradicional (en el que el reenvío se basa en la dirección de destino de un datagrama), como del reenvío generalizado (en el que pueden llevarse a cabo el reenvío y otras funciones dependiendo de los valores de varios campos distintos de la cabecera del datagrama), y hemos podido comprobar la versatilidad de esta última técnica. También hemos estudiado en detalle los protocolos IPv4 e IPv6, así como el tema del direccionamiento en Internet, que ha resultado ser mucho más profundo, sutil e interesante de lo que cabría esperar.

Armados con nuestros nuevos conocimientos acerca del plano de datos de la capa de red, estamos ya listos para sumergirnos en el plano de control de dicha capa. ¡Pero eso será en el Capítulo 5!

Problemas y cuestiones de repaso

Capítulo 4 Cuestiones de repaso

SECCIÓN 4.1

- R1. Revisemos parte de la terminología utilizada en el libro. Recuerde que el nombre que recibe un paquete de la capa de transporte es *segmento* y que el nombre de un paquete de la capa de enlace es *trama*. ¿Cuál es el nombre de un paquete de la capa de red? Recuerde que tanto los routers como los switches de la capa de enlace se denominan *comutadores de paquetes*. ¿Cuál es la diferencia fundamental entre un router y un switch de la capa de enlace?
- R2. Ya hemos indicado que la funcionalidad de la capa de red puede dividirse, en términos generales, en funcionalidad del plano de datos y funcionalidad del plano de control. ¿Cuáles son las principales funciones del plano de datos? ¿Y las del plano de control?
- R3. Hemos hecho la distinción entre la función de reenvío y la función de enrutamiento realizadas en la capa de red. ¿Cuáles son las diferencias fundamentales entre el enrutamiento y el reenvío?
- R4. ¿Cuál es el papel de la tabla de reenvío dentro de un router?
- R5. Hemos dicho que el modelo de servicio de una capa de red “define las características del transporte extremo a extremo de paquetes entre los hosts emisor y receptor”. ¿Cuál es el modelo de servicio de la capa de red de Internet? ¿Qué garantías proporciona el modelo de servicio de Internet, en lo que respecta a la entrega de datagramas de un host a otro host?

SECCIÓN 4.2

- R6. En la Sección 4.2 hemos visto que un router está compuesto, normalmente, por puertos de entrada, puertos de salida, un entramado de comutación y un procesador de enrutamiento. ¿Cuáles de estos elementos se implementan en hardware y cuáles en software? ¿Por qué? Volviendo a las nociones de plano de datos y plano de control de la capa de red, ¿cuál de ellos se implementa en hardware y cuál en software? ¿Por qué?
- R7. Explique por qué cada puerto de entrada de un router de alta velocidad almacena una copia de la tabla de reenvío.
- R8. ¿Qué se entiende por reenvío basado en el destino? ¿En qué difiere del reenvío generalizado? Tras leer la Sección 4.4, ¿cuál de esas dos técnicas es la adoptada por la tecnología SDN (redes definidas por software)?
- R9. Suponga que un paquete entrante se corresponde con dos o más entradas de la tabla de reenvío de un router. Con el reenvío tradicional basado en el destino, ¿qué norma aplica un router para determinar cuál de las dos reglas hay que invocar a la hora de determinar el puerto de salida hacia el que debe ser comutado el paquete entrante?
- R10. En la Sección 4.3 se han abordado tres tipos de entramados de comutación. Enumere y describa brevemente cada uno de ellos. ¿Cuál de ellos, si es que hay alguno, permite enviar múltiples paquetes en paralelo a través del entramado?
- R11. Describa cómo pueden perderse paquetes en los puertos de entrada. Describa cómo puede eliminarse la pérdida de paquetes en los puertos de entrada (sin utilizar buffers de capacidad infinita).
- R12. Describa cómo puede producirse una pérdida de paquetes en los puertos de salida. ¿Puede evitarse esta pérdida incrementando la velocidad del entramado de comutación?
- R13. ¿Qué es el bloqueo HOL? ¿Se produce en los puertos de entrada o en los puertos de salida?
- R14. En la Sección 4.2 hemos estudiado las siguientes disciplinas de planificación de paquetes: FIFO, prioridad, Round Robin (RR) y cola equitativa ponderada (WFQ, Weighted Fair

Queueing). ¿Cuáles de estas disciplinas de colas garantizan que todos los paquetes salgan en el mismo orden en que llegaron?

- R15. Proporcione un ejemplo que muestre por qué un operador de red podría querer dar prioridad a una clase de paquetes con respecto a otra clase de paquetes.
- R16. ¿Qué diferencia esencial hay entre las planificaciones de paquetes RR y WFQ? ¿Hay algún caso (*Sugerencia:* piense en las ponderaciones WFQ) en el que RR y WFQ se comporten exactamente igual?

SECCIÓN 4.3

- R17. Suponga que el host A envía al host B un segmento TCP encapsulado en un datagrama IP. Cuando el host B recibe el datagrama, ¿cómo sabe la capa de red del host B que debería pasar el segmento (es decir, la carga útil del datagrama) a TCP, en lugar de a UDP o a cualquier otro protocolo?
- R18. ¿Qué campo de la cabecera IP puede usarse para garantizar que un paquete se reenvíe a través de no más de N routers?
- R19. Recuerde que, como hemos visto, la suma de comprobación Internet se utiliza tanto en los segmentos de la capa de transporte (en las cabeceras UDP y TCP; Figuras 3.7 y 3.29, respectivamente), como en los datagramas de la capa de red (cabecera IP; Figura 4.16). Considere ahora un segmento de la capa de transporte encapsulado en un datagrama IP. ¿Existen bytes comunes en el datagrama IP sobre los cuales se calculen las sumas de comprobación de la cabecera del segmento y de la cabecera del datagrama? Explique su respuesta.
- R20. Cuando se fragmenta un datagrama de gran tamaño en múltiples datagramas más pequeños, ¿dónde se reensamblan estos datagramas más pequeños para obtener un único datagrama de mayor tamaño?
- R21. ¿Tienen direcciones IP los routers? En caso afirmativo, ¿cuántas?
- R22. ¿Cuál es el equivalente binario de 32 bits de la dirección IP 223.1.3.27?
- R23. Visite un host que utilice DHCP para obtener su dirección IP, su máscara de red, su router predeterminado y la dirección IP de su servidor DNS local. Enumere estos valores.
- R24. Suponga que hay tres routers entre un host de origen y un host de destino. Ignorando la fragmentación, se envía un datagrama IP desde el host de origen al host de destino. ¿A través de cuántas interfaces pasará? ¿Cuántas tablas de reenvío habrá que consultar para transportar el datagrama desde el origen al destino?
- R25. Suponga que una aplicación genera fragmentos de 40 bytes de datos cada 20 milisegundos y que cada fragmento se encapsula en un segmento TCP y luego en un datagrama IP. ¿Qué porcentaje de cada datagrama será información administrativa y qué porcentaje será datos de aplicación?
- R26. Suponga que adquiere un router inalámbrico y que lo conecta a su módem por cable. Suponga también que su ISP asigna dinámicamente una dirección IP a su dispositivo conectado (es decir, a su router inalámbrico). Además, suponga que tiene cinco equipos PC en su domicilio que utilizan 802.11 para conectarse de forma inalámbrica a su router inalámbrico. ¿Cómo se asignan las direcciones IP a los cinco PC? ¿Utiliza NAT el router inalámbrico? ¿Por qué o por qué no?
- R27. ¿Qué quiere decir el término “agregación de rutas”? ¿Por qué resulta útil para un router realizar la agregación de rutas?
- R28. ¿Qué es un protocolo “plug-and-play” o de configuración cero (*zeroconf*)?
- R29. ¿Qué es una dirección de red privada? ¿Debe haber datagramas con direcciones de red privadas en la red Internet pública? Explique su respuesta.
- R30. Compare y contraste los campos de cabecera de IPv4 e IPv6. ¿Tienen campos en común?

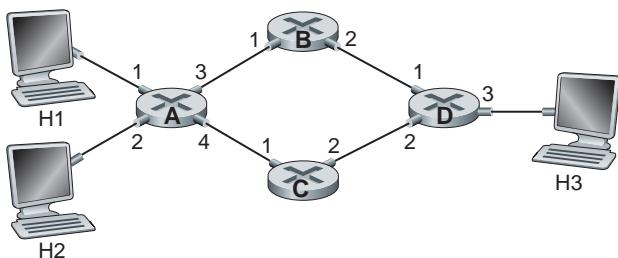
- R31. A veces se dice que cuando IPv6 se tuneliza a través de routers IPv4, IPv6 trata los túneles IPv4 como protocolos de la capa de enlace. ¿Está de acuerdo con esta afirmación? ¿Por qué o por qué no?

SECCIÓN 4.4

- R32. ¿En qué se diferencia el reenvío generalizado del reenvío basado en el destino?
- R33. ¿Cuál es la diferencia entre las tablas de reenvío basado en el destino, con las que nos hemos encontrado en la Sección 4.1, y la tabla de flujo de OpenFlow de las que hemos hablado en la Sección 4.4?
- R34. ¿Qué queremos decir al hablar de la operación “correspondencia más acción” de un router o un switch? En el caso de un conmutador de paquetes con reenvío basado en el destino, ¿qué correspondencia se busca y cuál es la acción realizada? En el caso de una SDN, enumere tres campos con los que puedan buscarse correspondencias y tres acciones que puedan tomarse.
- R35. Enumere tres campos de la cabecera de un datagrama IP con los que pueda buscarse una correspondencia en el reenvío generalizado compatible con OpenFlow 1.0. Indique, asimismo, tres campos de cabecera de un datagrama IP con los que *no pueda* buscarse una correspondencia en OpenFlow.

Problemas

- P1. Utilice la red mostrada en la parte inferior de la página.
- Especifique la tabla de reenvío del router A, de modo que todo el tráfico destinado al host H3 sea reenviado a través de la interfaz 3.
 - ¿Puede escribir una tabla de reenvío para el router A, de manera que todo el tráfico de H1 destinado al host H3 sea reenviado a través de la interfaz 3, mientras todo el tráfico de H2 destinado al host H3 sea reenviado a través de la interfaz 4? (*Sugerencia:* esta pregunta tiene truco.)
- P2. Suponga que dos paquetes llegan a dos puertos de entrada distintos de un router exactamente al mismo tiempo. Suponga también que no hay ningún otro paquete en ninguna parte del router.
- Suponga que los dos paquetes deben ser reenviados a dos puertos de salida diferentes. ¿Es posible reenviar los dos paquetes a través del entrampado de conmutación al mismo tiempo, cuando el entrampado utiliza un bus compartido?
 - Suponga que los dos paquetes deben ser reenviados a dos puertos de salida diferentes. ¿Es posible reenviar los dos paquetes a través del entrampado de conmutación al mismo tiempo, cuando el entrampado utiliza la conmutación vía memoria?
 - Suponga que los dos paquetes deben ser reenviados al mismo puerto de salida. ¿Es posible reenviar los dos paquetes a través del entrampado de conmutación al mismo tiempo, cuando el entrampado utiliza una malla?

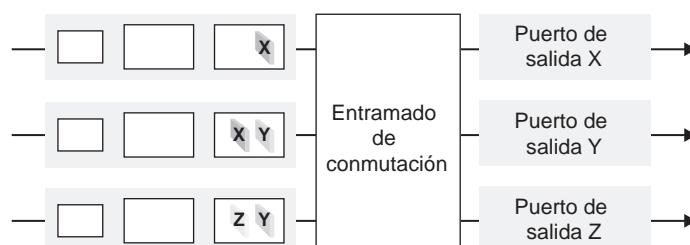


- P3. En la Sección 4.2 dijimos que el retardo máximo de cola es $(n-1)D$ si el entramado de conmutación es n veces más rápido que las velocidades de las líneas de entrada. Suponga que todos los paquetes tienen la misma longitud, que n paquetes llegan simultáneamente a los n puertos de entrada y que los n paquetes desean ser reenviados a diferentes puertos de salida. ¿Cuál será el retardo máximo de un paquete para los entramados de conmutación (a) de memoria, (b) de bus y (c) de malla?
- P4. Considere el dispositivo de conmutación mostrado en la parte inferior de la página. Suponga que todos los datagramas tienen la misma longitud fija, que el dispositivo opera de forma síncrona y particionada y que en una partición temporal se puede transferir un datagrama desde un puerto de entrada a un puerto de salida. El entramado de conmutación emplea una estructura de malla, por lo que, como máximo, se puede transferir un datagrama a un puerto de salida determinado en una partición de tiempo, pero distintos puertos de salida pueden recibir datagramas procedentes de distintos puertos de entrada en una misma partición de tiempo. ¿Cuál es el número mínimo de particiones de tiempo necesarias para transferir los paquetes mostrados desde los puertos de entrada a sus puertos de salida, suponiendo cualquier orden de planificación de la cola de entrada que desee (es decir, no tiene por qué existir el bloqueo HOL)? ¿Cuál es el número máximo necesario de particiones de tiempo, suponiendo el orden de planificación de caso peor que pueda imaginar y suponiendo que una cola de entrada no vacía nunca está inactiva?
- P5. Considere una red de datagramas que utiliza direcciones de host de 32 bits. Suponga que un router tiene cuatro enlaces, numerados de 0 a 3, y que los paquetes deben ser reenviados a las interfaces de los enlaces como sigue:

Rango de direcciones de destino	Interfaz de enlace
11100000 00000000 00000000 00000000 hasta 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 hasta 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 hasta 11100001 01111111 11111111 11111111	2
en otro caso	3

- Proporcione una tabla de reenvío con cinco entradas, que utilice la regla de coincidencia con el prefijo más largo y que reenvíe los paquetes a las interfaces de enlace correctas.
- Describa cómo determina su tabla de reenvío la interfaz de enlace apropiada para los datagramas con las siguientes direcciones de destino:

11001000 10010001 01010001 01010101
 11100001 01000000 11000011 00111100
 11100001 10000000 00010001 01110111



- P6. Considere una red de datagramas que utiliza direcciones de host de 8 bits. Suponga un router que utiliza las coincidencias con el prefijo más largo y cuya tabla de reenvío es la siguiente:

Coincidencia de prefijo	Interfaz
00	0
010	1
011	2
10	2
11	3

Para cada una de las cuatro interfaces, proporcione el rango asociado de direcciones del host de destino y el número de direcciones contenidas en el rango.

- P7. Considere una red de datagramas que utiliza direcciones de host de 8 bits. Suponga un router que utiliza las coincidencias con el prefijo más largo y cuya tabla de reenvío es la siguiente:

Coincidencia de prefijo	Interfaz
1	0
10	1
111	2
En otro caso	3

Para cada una de las cuatro interfaces, proporcione el rango asociado de direcciones del host de destino y el número de direcciones contenidas en el rango.

- P8. Sea un router que interconecta tres subredes: Subred 1, Subred 2 y Subred 3. Suponga que se requiere que todas las interfaces de cada una de estas tres subredes tengan el prefijo 223.1.17/24. Suponga también que se requiere que la Subred 1 admita al menos 6 interfaces, que la Subred 2 tiene que admitir al menos 90 interfaces y que la Subred 3 debe admitir hasta 12 interfaces. Determine tres direcciones de red (de la forma a.b.c.d/x) que satisfagan estas restricciones.
- P9. En la Sección 4.2.2 se ha proporcionado una tabla de reenvío de ejemplo (utilizando la coincidencia con el prefijo más largo). Vuelva a escribir esta tabla de reenvío utilizando la notación a.b.c.d/x en lugar de la notación en binario.
- P10. En el Problema P5 se le ha pedido que proporcione una tabla de reenvío (utilizando la regla de coincidencia con el prefijo más largo). Escriba de nuevo esta tabla de reenvío utilizando la notación a.b.c.d/x en lugar de la notación en binario.
- P11. Considere un subred cuyo prefijo es 128.119.40.128/26. Proporcione un ejemplo de una dirección IP (de la forma xxx.xxx.xxx.xxx) que pueda ser asignada a esta red. Suponga que un ISP posee el bloque de direcciones 128.119.40.64/26. Suponga que desea crear cuatro subredes a partir de este bloque de direcciones, teniendo cada bloque el mismo número de direcciones IP. ¿Cuáles serán los prefijos (expresados en formato a.b.c.d/x) para las cuatro subredes?
- P12. Considere la topología mostrada en la Figura 4.20. Denomine a las tres subredes con hosts (comenzando en el sentido horario a partir de las 12:00) como redes A, B y C. Denomine a las subredes que no tienen hosts como redes D, E y F.
- Asigne direcciones de red a cada una de estas seis subredes, teniendo en cuenta las siguientes restricciones: todas las direcciones tienen que ser asignadas a partir de 214.97.254/23;

la subred A tendrá que disponer de las direcciones suficientes como para dar soporte a 250 interfaces; la subred B tendrá que disponer de las direcciones suficientes como para dar soporte a 120 interfaces y la subred C tendrá que disponer de las direcciones suficientes como para dar soporte a 120 interfaces. Por supuesto, las subredes D, E y F deberían poder dar soporte a dos interfaces. Para cada subred, la asignación debería hacerse empleando el formato a.b.c.d/x o a.b.c.d/x – e.f.g.h/y.

- b. Utilizando su respuesta al apartado (a), proporcione las tablas de reenvío (utilizando la regla de coincidencia con el prefijo más largo) para cada uno de los tres routers.
- P13. Utilice el servicio whois del ARIN—American Registry for Internet Numbers— (<http://www.arin.net/whois>) - para determinar los bloques de direcciones IP de tres universidades. ¿Pueden usarse los servicios de whois para determinar con certidumbre la ubicación geográfica de una dirección IP específica? Utilice www.maxmind.com para determinar las ubicaciones de los servidores web de cada una de esas universidades.
- P14. Se envía un datagrama de 2.400 bytes por un enlace que tiene una MTU de 700 bytes. Suponga que el datagrama original está marcado con el número de identificación 422. ¿Cuántos fragmentos se generan? ¿Cuáles son los valores de los distintos campos relacionados con la fragmentación, en los datagramas IP generados?
- P15. Suponga que el tamaño de los datagramas está limitado a 1.500 bytes (incluyendo la cabecera) entre el host A y el host de destino B. Suponiendo una cabecera IP de 20 bytes, ¿cuántos datagramas se necesitarían para enviar un archivo MP3 de 5 millones de bytes? Explique los cálculos que haya realizado para dar una respuesta.
- P16. Considere la red de la Figura 4.25. Suponga que el ISP asigna al router la dirección 24.34.112.235, en lugar de la que se muestra en la figura, y que la dirección de la red doméstica es 192.168.1/24.
- Asigne direcciones a todas las interfaces de la red doméstica.
 - Suponga que cada host tiene dos conexiones TCP activas, todas ellas con el puerto 80 del host 128.119.40.86. Proporcione las seis entradas correspondientes de la tabla de traducciones NAT.
- P17. Suponga que está interesado en detectar el número de hosts que hay detrás de un traductor NAT y que observa que la capa IP marca secuencialmente con un número de identificación cada paquete IP. El número de identificación del primer paquete IP generado por un host es un número aleatorio y los números de identificación de los siguientes paquetes IP se asignan de forma secuencial. Suponga que todos los paquetes IP generados por los hosts que hay detrás del traductor NAT se envían al exterior.
- Basándose en esta observación y suponiendo que puede husmear todos los paquetes enviados por el traductor NAT al exterior, ¿puede esbozar una técnica sencilla para detectar el número de hosts diferentes que hay detrás del traductor NAT? Justifique su respuesta.
 - Si los números de identificación no se asignan secuencialmente, sino aleatoriamente, ¿serviría su técnica? Justifique su respuesta.
- P18. En este problema se explora el impacto de NAT sobre las aplicaciones P2P. Suponga que un par cuyo nombre de usuario es Arnold descubre mediante una consulta que otro par con el nombre de usuario Bernard tiene un archivo que desea descargar. Suponga también que Bernard y Arnold están detrás de un traductor NAT. Intente deducir una técnica que permita a Arnold establecer una conexión TCP con Bernard sin realizar una configuración NAT específica de la aplicación. Si tiene dificultades para definir una técnica así, explique por qué.
- P19. Considere la red SDN OpenFlow mostrada en la Figura 4.30. Suponga que el comportamiento de reenvío deseado para los datagramas que llegan a s2 es el siguiente:
- Los datagramas que lleguen de los hosts h5 o h6 a través del puerto de entrada 1 y que estén destinados a los hosts h1 o h2, deben reenviarse a través del puerto de salida 2.

- Los datagramas que lleguen de los hosts h1 o h2 a través del puerto de entrada 2 y que estén destinados a los hosts h5 o h6, deben reenviarse a través del puerto de salida 1.
- Los datagramas que lleguen a través de los puertos de entrada 1 o 2 y que estén destinados a los hosts h3 o h4, deben entregarse al host especificado.
- Los hosts h3 y h4 deben poder intercambiarse datagramas.

Especifique las entradas de la tabla de flujo de s2 que permiten implementar este comportamiento de reenvío.

- P20. Considere de nuevo la red SDN OpenFlow mostrada en la Figura 4.30. Suponga que el comportamiento de reenvío deseado para los datagramas que llegan a s2 desde los hosts h3 o h4 es el siguiente:

- Los datagramas que lleguen del host h3 y que estén destinados a los hosts h1, h2, h5 o h6, deben reenviarse través de la red en el sentido de las agujas del reloj.
- Los datagramas que lleguen del host h4 y que estén destinados a los hosts h1, h2, h5 o h6, deben reenviarse través de la red en el sentido contrario a las agujas del reloj.

Especifique las entradas de la tabla de flujo de s2 que permiten implementar este comportamiento de reenvío.

- P21. Considere de nuevo el escenario del Problema P19. Especifique las entradas de las tablas de flujo de los conmutadores de paquetes s1 y s3, de modo que los datagramas entrantes con una dirección de origen igual a h3 o h4 se enruten hacia los hosts de destino especificados en el campo de dirección de destino del datagrama IP. (*Sugerencia:* Las reglas de sus tablas de reenvío deben incluir tanto el caso de que un datagrama entrante esté destinado a un host directamente conectado, como el caso de que haya que reenviarlo a un router vecino para su eventual entrega allí a un host.)

- P22. Considere de nuevo la red SDN OpenFlow mostrada en la Figura 4.30. Suponga que queremos que el conmutador s2 funcione como un cortafuegos. Especifique la tabla de flujo de s2 que implemente los siguientes comportamientos de cortafuegos para la entrega de datagramas destinados a los hosts h3 y h4 (proporcione una tabla de flujo diferente para cada uno de los cuatro comportamientos de cortafuegos indicados). No hace falta que especifique el comportamiento de reenvío de s2 que se encarga de reenviar tráfico hacia otros routers.

- Solo el tráfico que llega de los hosts h1 y h6 debe entregarse a los hosts h3 o h4 (es decir, se bloquea el tráfico que llegue de los hosts h2 y h5).
- Solo se permite entregar a los hosts h3 o h4 el tráfico TCP (es decir, se bloquea el tráfico UDP).
- Solo hay que entregar el tráfico destinado a h3 (es decir, se bloquea todo el tráfico destinado a h4).
- Solo hay que entregar el tráfico UDP de h1 destinado a h3. Todo el tráfico restante se bloquea.

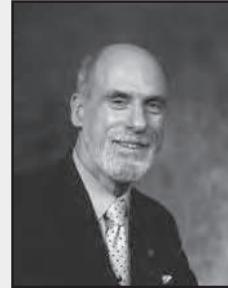
Práctica de laboratorio con Wireshark

En el sitio web del libro, <http://www.pearsonhighered.com/cs-recources>, encontrará una práctica de laboratorio con Wireshark que examina el funcionamiento del protocolo IP y el formato del datagrama IP en concreto.

UNA ENTREVISTA CON...

Vinton G. Cerf

Vinton G. Cerf es Vicepresidente y Jefe de Estrategia de Internet de Google. Ha trabajado durante más de 16 años en MCI en distintos puestos, acabando allí su carrera profesional como Vicepresidente senior de Estrategia Tecnológica. Es muy conocido por haber sido co-diseñador de los protocolos TCP/IP y de la arquitectura de Internet. Durante el tiempo que trabajó, entre 1976 y 1982, en la Agencia de Proyectos de Investigación Avanzada de Defensa (DARPA, Department of Defense Advanced Research Projects Agency) de Estados Unidos, desempeñó un papel crucial dirigiendo el desarrollo de Internet y de tecnologías de seguridad y de empaquetado de datos relacionadas con Internet. Recibió la Medalla Presidencial de la Libertad en 2005 y la Medalla Nacional de Tecnología en 1997. Es licenciado en Matemáticas por la Universidad de Stanford y doctor en Ciencias de la Computación por UCLA.



¿Qué le hizo especializarse en el campo de las redes?

Estaba trabajando como programador en UCLA a finales de la década de 1960. Mi trabajo estaba financiado por la Agencia de Proyectos de Investigación Avanzada de Defensa de Estados Unidos (que entonces se llamada ARPA y ahora DARPA). Trabajaba en el laboratorio del profesor Leonard Kleinrock en el Centro de Medidas de Red para la recientemente creada red ARPAnet. El primer nodo de ARPAnet fue instalado en UCLA el 1 de septiembre de 1969. Yo era responsable de programar una computadora que se utilizaba para capturar información de rendimiento acerca de ARPAnet y de devolver esa información para compararla con los modelos matemáticos y las predicciones relativas al rendimiento de la red.

A algunos otros estudiantes y a mí mismo nos encargaron trabajar en los denominados protocolos de nivel de host de ARPAnet, es decir, en los procedimientos y formatos que permitirían que muchos tipos distintos de computadoras interactuaran entre sí a través de la red. Fue una investigación fascinante en un mundo nuevo (para mí) de comunicación y computación distribuidas.

Cuando diseñó el protocolo, ¿se imaginaba que IP llegaría a ser tan ubicuo como lo es hoy día?

Cuando Bob Kahn y yo comenzamos a trabajar en el tema en 1973, creo que estábamos muy concentrados en el problema fundamental: cómo hacer que una serie de redes de paquetes heterogéneas interoperaran, partiendo del supuesto de que no podíamos modificar las propias redes. Intentábamos encontrar una forma de poder interconectar un conjunto arbitrario de redes de comutación de paquetes de manera transparente, de modo que las computadoras host pudieran comunicarse extremo a extremo sin necesidad de realizar ninguna traducción intermedia. Creo que éramos conscientes de que estábamos trabajando con una tecnología potente y ampliable, pero me parece que no teníamos una imagen clara de lo que podía llegar a ser nuestro mundo con centenares de millones de computadoras interconectadas a través de Internet.

¿Cuál cree que será el futuro de las redes y de Internet? ¿Cuáles cree que son los principales desafíos/ obstáculos a los que se enfrentarán?

Creo que la propia Internet y las redes en general continuarán expandiéndose. Ya existen suficientes evidencias para afirmar que pronto habrá miles de millones de dispositivos compatibles con Internet, incluyendo equipos tales como teléfonos móviles, frigoríficos, asistentes digitales personales (PDA), servidores domésticos, televisiones, además de la colección habitual de computadoras portátiles, servidores, etc. Entre los principales desafíos se incluyen el soporte para la movilidad, la capacidad de las baterías, la capacidad de los enlaces de acceso a la red y la capacidad de ampliar el núcleo óptico de la red de forma ilimitada. El diseño de una extensión interplanetaria de Internet es un proyecto en el que estoy profundamente involucrado en el Jet Propulsion Laboratory. Asimismo, necesitaremos hacer la transición desde IPv4 [direcciones de 32 bits] a IPv6 [direcciones de 128 bits]. ¡Hay una larga lista de desafíos!

¿Quién le ha servido de inspiración profesionalmente?

Mi colega Bob Kahn; mi director de tesis, Gerald Estrin; mi mejor amigo, Steve Crocker (¡nos conocimos en la facultad y fue él el que me introdujo en el mundo de las computadoras en 1960!); y los miles de ingenieros que hacen que Internet continúe evolucionando hoy día.

¿Qué consejo le daría a los estudiantes que inician su andadura en el campo de las redes/ Internet?

Que tienen que pensar sin tener en cuenta las limitaciones de los sistemas existentes. Deben tratar de imaginar qué cosas son posibles y a continuación hacer el trabajo duro de intentar averiguar cómo ir hasta allí desde el estado actual de las cosas. Es preciso ser capaz de soñar: media docena de colegas y yo hemos estado trabajando en el Jet Propulsion Laboratory en el diseño de una extensión interplanetaria de la Internet terrestre. Puede que se tarden décadas en implementar esto, misión a misión, pero parafraseando un conocido dicho: “El hombre debe poder llegar más allá de donde su mano alcanza; ¿o para qué son los cielos, si no?”.

La capa de red: el plano de control

En este capítulo, completaremos nuestro viaje a través de la capa de red hablando del segundo de sus constituyentes, el **plano de control**: la lógica de la red que controla no solo cómo se reenvía un datagrama de un router a otro, a lo largo de una ruta extremo a extremo desde el host de origen al host de destino, sino también cómo se configuran y gestionan los servicios y componentes de la capa de red. En la Sección 5.2 hablaremos de los algoritmos tradicionales de enrutamiento utilizados para calcular las rutas de coste mínimo en un grafo; estos algoritmos son la base de dos protocolos de enrutamiento ampliamente implantados en Internet: OSPF y BGP, de los que hablaremos en las secciones 5.3 y 5.4, respectivamente. Como veremos, OSPF es un protocolo de enrutamiento que opera dentro de la red de un único ISP. BGP es un protocolo de enrutamiento que sirve para interconectar todas las redes de Internet; por ello se suele decir que BGP es el “pegamento” que mantiene unida Internet. Tradicionalmente, los protocolos de enrutamiento del plano de control se han implementado junto con las funciones de reenvío del plano de datos monolíticamente, dentro de un router. Como vimos en la introducción al Capítulo 4, las redes definidas por software (SDN) separan claramente los planes de datos y de control, implementando las funciones del plano de control mediante un servicio “controlador” separado que es distinto, y remoto, con respecto a los componentes de reenvío de los routers que controla. Hablaremos de los controladores SDN en la Sección 5.5.

En las secciones 5.6 y 5.7, hablaremos un poco de las interioridades de la gestión de una red IP, presentando los protocolos ICMP (*Internet Control Message Protocol*, protocolo de mensajes de control de Internet) y SNMP (*Simple Network Management Protocol*, protocolo simple de gestión de red).

5.1 Introducción

Establezcamos rápidamente el contexto para nuestro estudio del plano de control de la red, recordando las Figuras 4.2 y 4.3. Allí vimos que la tabla de reenvío (en el caso del reenvío basado en el destino) y la tabla de flujo (en el caso del reenvío generalizado) eran los principales elementos que

enlazaban los planos de datos y de control de la capa de red. Dijimos que estas tablas especifican el comportamiento de reenvío del plano de datos local de un router. Vimos que, en el caso del reenvío generalizado, las acciones tomadas (Sección 4.4.2) podían incluir no solo reenviar un paquete a través de un puerto de salida del router, sino también eliminar un paquete, duplicar un paquete y/o reescribir los campos de cabecera de las capas 2, 3 o 4 del paquete.

En este capítulo estudiaremos cómo se calculan, mantienen e instalan estas tablas de reenvío y de flujo. En nuestra introducción a la capa de red en la Sección 4.1 aprendimos que existen dos posibles enfoques para hacerlo.

- *Control por router*. La Figura 5.1 ilustra el caso en el que se ejecuta un algoritmo de enrutamiento en todos y cada uno de los routers; cada router contiene tanto una función de reenvío como una función de enrutamiento. En cada router hay un componente de enrutamiento que se comunica con los componentes de enrutamiento de otros routers, con el fin de calcular los valores de su tabla de reenvío. Esta técnica del control por router se ha estado utilizando en Internet durante décadas. Los protocolos OSPF y BGP que estudiaremos en las Secciones 5.3 y 5.4 están basados en este enfoque de control por router.
- *Control lógicamente centralizado*. La Figura 5.2 ilustra el caso en que un controlador lógicamente centralizado calcula y distribuye las tablas de reenvío que tienen que usar todos y cada uno de los routers. Como vimos en la Sección 4.4, la abstracción generalizada correspondencia-acción permite al router llevar a cabo tanto el reenvío IP tradicional, como un rico conjunto de otras funciones (compartición de carga, cortafuegos y NAT) que anteriormente se habían venido implementando mediante dispositivos intermediarios separados.

El controlador interactúa con un agente de control (AC) que reside en cada router, a través de un protocolo bien definido, para configurar y gestionar la tabla de flujo de ese router. Normalmente, el AC tiene una funcionalidad mínima; su trabajo consiste en comunicarse con el controlador y en hacer lo que el controlador le ordene. A diferencia de los algoritmos de enrutamiento de la Figura 5.1, los AC no interactúan directamente entre sí, ni participan activamente en el cálculo de la tabla de reenvío. Esta es una distinción fundamental entre el control por router y el control lógicamente centralizado.

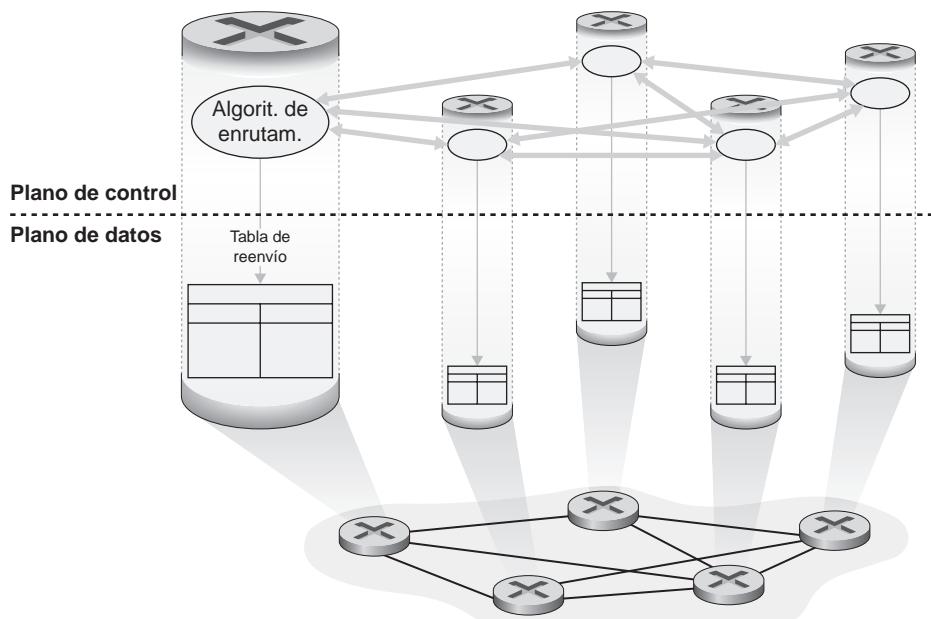


Figura 5.1 ♦ Control por router: los componentes individuales del algoritmo de enrutamiento interactúan en el plano de control.

Por control “lógicamente centralizado” [Levin 2012] queremos decir que al servicio de control de enrutamiento se accede como si fuera un único punto central de servicio, aunque es probable que el servicio se implemente mediante múltiples servidores, por razones de tolerancia a fallos y de escalabilidad del rendimiento. Como veremos en la Sección 5.5, la tecnología SDN adopta esta noción de un controlador lógicamente centralizado, una técnica que se está empleando cada vez más en instalaciones de producción. Google utiliza SDN para controlar los routers de su red de área extensa global B4, que interconecta sus centros de datos [Jain 2013]. SWAN [Hong 2013], de Microsoft Research, utiliza un controlador lógicamente centralizado para gestionar el enrutamiento y el reenvío entre una red de área extensa y una red de centros de datos. China Telecom y China Unicom están empleando SDN tanto dentro de los centros de datos como entre los centros de datos [Li 2015]. AT&T ha señalado [AT&T 2013] que “soporta muchas capacidades SDN, así como mecanismos propietarios, definidos de forma independiente, que encajan en el marco arquitectónico de SDN”.

5.2 Algoritmos de enrutamiento

En esta sección estudiaremos los **algoritmos de enrutamiento**, cuyo objetivo es determinar buenas rutas desde los emisores a los receptores, a través de la red de routers. Normalmente, una “buena ruta” es aquella que tiene el coste mínimo. Sin embargo, veremos que, en la práctica, ciertos problemas del mundo real, como las políticas utilizadas (por ejemplo, una regla que establezca que “el router x , que pertenece a la organización Y , no debería reenviar ningún paquete

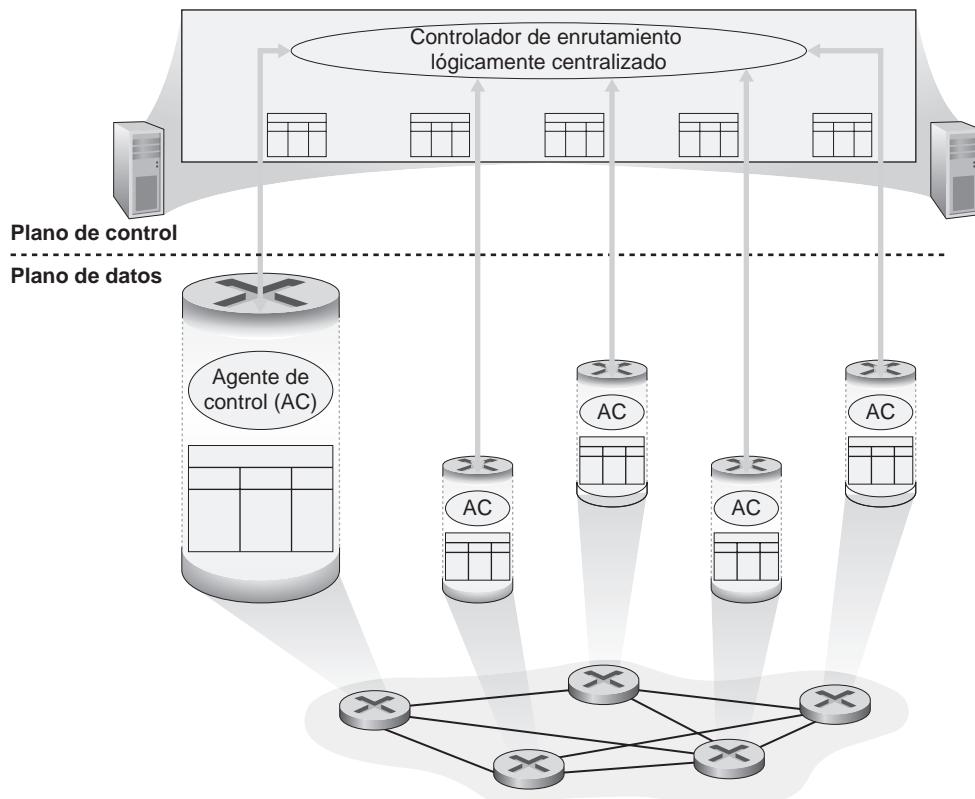


Figura 5.2 ♦ Control lógicamente centralizado: un controlador diferenciado, normalmente remoto, interactúa con agentes de control (AC) locales.

cuyo origen sea la red de la organización Z'), también entran en juego. Hay que resaltar que, independientemente de si el plano de control de la red adopta una solución de control por router o de control lógicamente centralizado, siempre debe existir una secuencia de routers bien definida que el paquete atravesará en su viaje desde el host emisor al host receptor. Por ello, los algoritmos de enrutamiento que calculan estas rutas tienen una importancia fundamental, siendo otro de los candidatos para nuestra lista de los 10 conceptos de redes más importantes.

Para formular los problemas de enrutamiento se utilizan grafos. Recuerde que un **grafo** $G = (N, E)$ es un conjunto N de nodos y una colección E de aristas, donde cada arista es una pareja de nodos de N . En el contexto del enrutamiento de la capa de red, los nodos del grafo representan los routers (los puntos en los que se toman las decisiones acerca del reenvío de los paquetes), mientras que las aristas que conectan esos nodos representan los enlaces físicos entre los routers. En la Figura 5.3 se muestra la abstracción en forma de grafo de una red de computadoras. Si desea ver algunos grafos que representan mapas de redes reales, consulte [Dodge 2016, Cheswick 2000]; para ver un estudio de cómo diferentes modelos basados en grafos permiten modelar Internet, consulte [Zegura 1997, Faloutsos 1999, Li 2004].

Como se muestra en la Figura 5.3, una arista también tiene un valor que representa su coste. Normalmente, el coste de una arista puede reflejar la longitud física del enlace correspondiente (por ejemplo, un enlace transoceánico tendría un coste mayor que un enlace terrestre de corto alcance), la velocidad del enlace o el coste monetario asociado con el enlace. Para nuestros propósitos, simplemente utilizaremos el coste del enlace como algo que nos viene dado, sin preocuparnos por cómo se determina. Para cualquier arista (x, y) de E , denominaremos $c(x, y)$ al coste de la arista que conecta los nodos x e y . Si el par (x, y) no pertenece a E , hacemos $c(x, y) = \infty$. Asimismo, solo vamos a considerar en nuestra exposición los grafos no dirigidos (es decir, grafos cuyas aristas no tienen una dirección), de modo que la arista (x, y) es la misma que la arista (y, x) y además $c(x, y) = c(y, x)$; sin embargo, los algoritmos que estudiaremos pueden generalizarse fácilmente para el caso de enlaces dirigidos que tengan un coste diferente en cada dirección. Por último, un nodo y se dice que es un **vecino** del nodo x si (x, y) pertenece a E .

Dado que las distintas aristas del grafo abstracto tienen costes asignados, un objetivo natural de un algoritmo de enrutamiento es identificar las rutas de coste mínimo entre los orígenes y los destinos. Con el fin de definir este problema de manera más precisa, recuerde que una **ruta** en un grafo $G = (N, E)$ es una secuencia de nodos (x_1, x_2, \dots, x_p) tal que cada una de las parejas $(x_1, x_2), (x_2, x_3), \dots, (x_{p-1}, x_p)$ son aristas pertenecientes a E . El coste de una ruta (x_1, x_2, \dots, x_p) es simplemente la suma del coste de todas las aristas que componen la ruta; es decir, $c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$. Dados dos nodos cualesquiera x e y , normalmente existen muchas rutas entre los dos nodos, teniendo cada una de ellas un coste. Una o más de estas rutas será una **ruta de coste mínimo**. Por tanto, el problema del coste mínimo está claro: hallar una ruta entre el origen y el destino que tenga el mínimo coste. Por ejemplo, en la Figura 5.3 la ruta de coste mínimo entre el nodo de origen u y el nodo de destino w es (u, x, y, w) , con un coste de ruta igual a 3. Observe que si todas las aristas

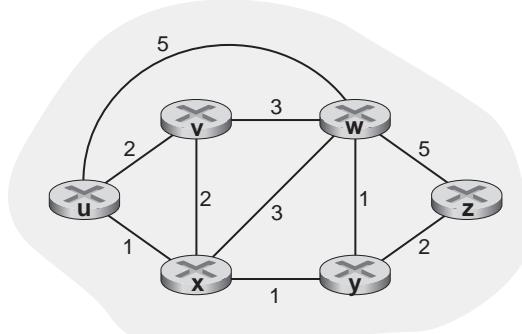


Figura 5.3 ♦ Modelo de grafo abstracto de una red de computadoras.

del grafo tienen el mismo coste, la ruta de coste mínimo es también la **ruta más corta** (es decir, la ruta con el número mínimo de enlaces entre el origen y el destino).

Como ejercicio sencillo, intente encontrar la ruta de coste mínimo desde el nodo u al z en la Figura 5.3 y reflexione sobre cómo ha calculado esa ruta. Si es usted como la mayor parte de la gente, habrá determinado la ruta de u a z examinando la Figura 5.3, trazando unas pocas rutas de u a z , y llegando de alguna manera al convencimiento de que la ruta que ha elegido es la de coste mínimo de entre todas las posibles rutas. (¿Ha comprobado las 17 rutas posibles entre u y z ? ¡Probablemente no!). Este cálculo es un ejemplo de algoritmo de enrutamiento centralizado (el algoritmo de enrutamiento se ejecutó en un lugar, su cerebro, que dispone de la información completa de la red). En términos generales, una forma de clasificar los algoritmos de enrutamiento es dependiendo de si son centralizados o descentralizados.

- Un **algoritmo de enrutamiento centralizado** calcula la ruta de coste mínimo entre un origen y un destino utilizando un conocimiento global y completo acerca de la red. Es decir, el algoritmo toma como entradas la conectividad entre todos los nodos y todos los costes de enlace. Esto requiere, por tanto, que el algoritmo obtenga de alguna manera esta información antes de realizar el propio cálculo. El cálculo en sí puede hacerse en un solo sitio (por ejemplo, un controlador lógicamente centralizado, como el de la Figura 5.2), o replicarse en el componente de enrutamiento de todos y cada uno de los routers (por ejemplo, como en la Figura 5.1). La característica distintiva aquí, sin embargo, es que un algoritmo global dispone de toda la información acerca de la conectividad y de los costes de los enlaces. Los algoritmos con información de estado global a menudo se denominan **algoritmos de estado de enlaces (LS, Link-State)**, ya que el algoritmo tiene que ser consciente del coste de cada enlace de la red. Estudiaremos los algoritmos LS en la Sección 5.2.1.
- En un **algoritmo de enrutamiento descentralizado**, el cálculo de la ruta de coste mínimo se realiza por parte de los routers de manera iterativa y distribuida. Ningún nodo tiene toda la información acerca del coste de todos los enlaces de la red. En lugar de ello, al principio, cada nodo solo conoce los costes de sus propios enlaces directamente conectados. Después, a través de un proceso iterativo de cálculo e intercambio de información con sus nodos vecinos, cada nodo calcula gradualmente la ruta de coste mínimo hacia un destino o conjunto de destinos. El algoritmo de enrutamiento descentralizado que estudiaremos más adelante en la Sección 5.2.2 se denomina **algoritmo de vector de distancias (DV, Distance-Vector)**, porque cada nodo mantiene un vector de estimaciones de los costes (distancias) a todos los demás nodos de la red. Dichos algoritmos descentralizados, con intercambio interactivo de mensajes entre routers vecinos, está quizás adaptado de una forma más natural a los planos de control en los que los routers interactúan directamente entre sí, como en la Figura 5.1.

Una segunda forma general de clasificar los algoritmos de enrutamiento es según sean estáticos o dinámicos. En los **algoritmos de enrutamiento estático**, las rutas cambian muy lentamente con el tiempo, a menudo como resultado de una intervención humana (por ejemplo, una persona que edita manualmente la tabla de reenvío de un router). Los **algoritmos de enrutamiento dinámico** modifican los caminos de enrutamiento a medida que la carga de tráfico o la topología de la red cambian. Un algoritmo dinámico puede ejecutarse periódicamente o como respuesta directa a cambios en la topología o en el coste de los enlaces. Aunque los algoritmos dinámicos responden mejor a los cambios de la red, también son más susceptibles a problemas como los bucles de enrutamiento y la oscilación de rutas.

Una tercera forma de clasificar los algoritmos de enrutamiento es según sean sensibles o no a la carga. En un **algoritmo sensible a la carga**, los costes de enlace varían de forma dinámica para reflejar el nivel actual de congestión en el enlace subyacente. Si se asocia un coste alto con un enlace que actualmente esté congestionado, el algoritmo de enrutamiento tenderá a elegir rutas que eviten tal enlace congestionado. Aunque los primeros algoritmos de enrutamiento de ARPAnet eran sensibles a la carga [McQuillan 1980], se encontró una serie de dificultades [Huitema 1998].

Los algoritmos de enrutamiento actuales de Internet (como RIP, OSPF y BGP) son **algoritmos no sensibles a la carga**, ya que el coste de un enlace no refleja explícitamente su nivel actual (o reciente) de congestión.

5.2.1 Algoritmo de enrutamiento de estado de enlaces (LS)

Recuerde que en un algoritmo de estado de enlaces, la topología de la red y el coste de todos los enlaces son conocidos; es decir, están disponibles como entradas para el algoritmo LS. En la práctica, esto se consigue haciendo que cada nodo difunda paquetes del estado de los enlaces a *todos* los demás nodos de la red, conteniendo cada paquete de estado de enlaces las identidades y costes de sus enlaces conectados. En la práctica (por ejemplo, con el protocolo de enrutamiento OSPF de Internet, que estudiaremos en la Sección 5.3), esto suele realizarse mediante un **algoritmo de difusión de estado de los enlaces** [Perlman 1999]. El resultado de difundir la información por parte de los nodos es que todos los nodos tienen una visión completa e idéntica de la red. Cada nodo puede entonces ejecutar el algoritmo LS y calcular el mismo conjunto de rutas de coste mínimo que cualquier otro nodo.

El algoritmo de enrutamiento de estado de enlaces que presentamos a continuación se conoce como *algoritmo de Dijkstra*, en honor a su inventor. Un algoritmo estrechamente relacionado con él es el algoritmo de Prim; consulte [Cormen 2001] para ver una exposición de carácter general sobre los algoritmos de grafos. El algoritmo de Dijkstra calcula la ruta de coste mínimo desde un nodo (el origen, al que denominaremos u) hasta todos los demás nodos de la red. El algoritmo de Dijkstra es iterativo y tiene la propiedad de que después de la k -ésima iteración del algoritmo se conocen las rutas de coste mínimo hacia k nodos de destino, y entre las rutas de coste mínimo a todos los nodos de destino, estas k rutas tendrán los k costes más pequeños. Definimos la siguiente notación:

- $D(v)$: coste de la ruta de coste mínimo desde el nodo de origen al destino v , para esta iteración del algoritmo.
- $p(v)$: nodo anterior (vecino de v) a lo largo de la ruta de coste mínimo desde el origen hasta v .
- N' : subconjunto de nodos; v pertenece a N' si la ruta de coste mínimo desde el origen hasta v se conoce de forma definitiva.

El algoritmo de enrutamiento centralizado consta de un paso de inicialización seguido de un bucle. El número de veces que se ejecuta el bucle es igual al número de nodos de la red. Al terminar, el algoritmo habrá calculado las rutas más cortas desde el nodo de origen u hasta cualquier otro nodo de la red.

Algoritmo de estado de enlaces (LS) para el nodo de origen u

```

1  Inicialización:
2  N' = {u}
3  for todo nodo v
4      if v es un vecino de u
5          then D(v) = c(u,v)
6      else D(v) =
7
8  Loop
9  encontrar w no perteneciente a N' tal que D(w) sea un mínimo
10 añadir w a N'
11 actualizar D(v) para cada vecino v de w que no pertenezca a N':
12     D(v) = min(D(v), D(w)+ c(w,v) )
13 /* el nuevo coste a v es o bien el antiguo coste a v o el coste
14 de la ruta de coste mínimo a w más el coste desde w a v */
15 until N' = N

```

Como ejemplo, consideremos la red de la Figura 5.3 y calculemos las rutas de coste mínimo desde u a todos los destinos posibles. En la Tabla 5.1 se muestra una tabla de resumen de los cálculos del algoritmo, donde cada línea de la tabla proporciona los valores de las variables del algoritmo al final de la iteración. Veamos en detalle los primeros pasos.

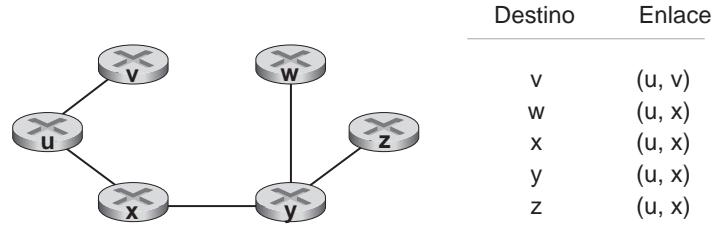
- En el paso de inicialización, las rutas de coste mínimo actualmente conocidas desde u a sus vecinos conectados directamente, v , x y w , se inicializan a 2, 1 y 5, respectivamente. En particular, fíjese en que el coste a w se define como 5 (aunque pronto veremos que, de hecho, existe una ruta de coste más pequeño), ya que este es el coste del enlace directo (un salto) de u a w . Los costes a y y z se hacen igual a infinito porque no están directamente conectados a u .
- En la primera iteración, buscamos entre aquellos nodos que todavía no se han añadido al conjunto N' y localizamos el nodo que tiene el coste mínimo después de finalizar la iteración previa. Dicho nodo es x , con un coste de 1, y por tanto x se añade al conjunto N' . La línea 12 del algoritmo LS se ejecuta entonces para actualizar $D(v)$ para todos los nodos v , proporcionando los resultados mostrados en la segunda línea (Paso 1) de la Tabla 5.1. El coste de la ruta a v no varía. Se determina que el coste de la ruta a w (que era 5 al final de la inicialización) a través del nodo x es ahora igual a 4. Por ello, se selecciona esta ruta de coste mínimo y el predecesor de w a lo largo de la ruta más corta desde u se establece en x . De forma similar, se calcula el coste a y (a través de x) y se obtiene que es 2, actualizándose la tabla de acuerdo con ello.
- En la segunda iteración, se determina que los nodos v e y tienen las rutas de coste mínimo (2) y se deshace el empate arbitrariamente, añadiendo y al conjunto N' , de modo que ahora N' contiene a u , x e y . El coste de los restantes nodos que todavía no pertenecen a N' , es decir, los nodos v , w y z , se actualiza en la línea 12 del algoritmo LS, dando los resultados mostrados en la tercera fila de la Tabla 5.1.
- Y así sucesivamente...

Cuando el algoritmo LS termina, tenemos para cada nodo su predecesor a lo largo de la ruta de coste mínimo desde el nodo de origen. Para cada predecesor, también tenemos su predecesor, y así de este modo podemos construir la ruta completa desde el origen a todos los destinos. La tabla de reenvío de un nodo, por ejemplo del nodo u , puede entonces construirse a partir de esta información, almacenando para cada destino el nodo del siguiente salto en la ruta de coste mínimo desde u al destino. La Figura 5.4 muestra las rutas de coste mínimo resultantes y la tabla de reenvío de u para la red de la Figura 5.3.

¿Cuál es la complejidad de cálculo de este algoritmo? Es decir, dados n nodos (sin contar el origen) ¿cuántos cálculos hay que realizar, en el caso peor, para determinar las rutas de coste mínimo desde el origen a todos los destinos? En la primera iteración, tenemos que buscar a través de los n nodos para determinar el nodo w que no pertenece a N' y que tiene el coste mínimo. En la segunda iteración, tenemos que comprobar $n - 1$ nodos para determinar el coste mínimo; en la tercera iteración, hay que comprobar $n - 2$ nodos, y así sucesivamente. En general, el número total

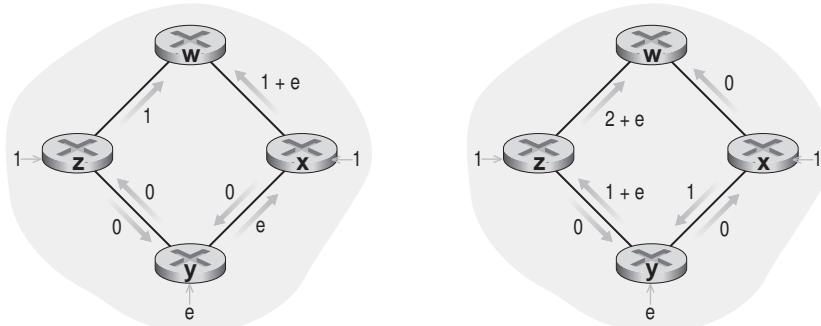
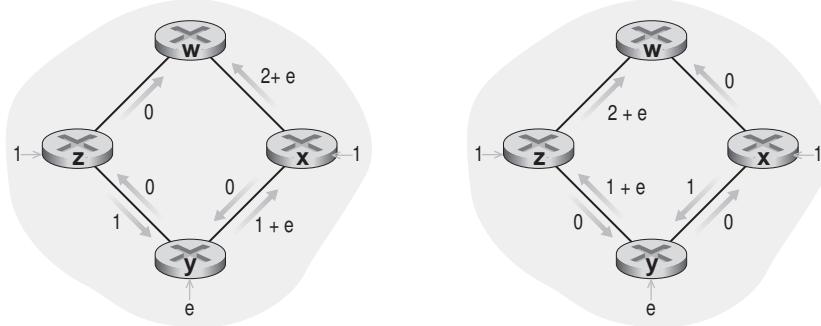
Paso	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	∞	∞
1	ux	2, u	4, x		2, x	∞
2	uxy	2, u	3, y		4, y	
3	$uxyv$		3, y		4, y	
4	$uxyw$				4, y	
5	$uxywz$					

Tabla 5.1 ♦ Ejecución del algoritmo de estado de enlaces para la red de la Figura 5.3.

**Figura 5.4** ♦ Rutas de coste mínimo y tabla de reenvío para el nodo u.

de nodos a través de los que tenemos que buscar, teniendo en cuenta todas las iteraciones, es igual a $n(n + 1)/2$ y, por tanto, decimos que la anterior implementación del algoritmo LS tiene, en el caso peor, una complejidad de orden n al cuadrado: $O(n^2)$. Una implementación más sofisticada de este algoritmo, que utiliza una estructura de datos conocida como montón (*heap*), puede calcular el mínimo en la línea 9 en tiempo logarítmico en lugar de lineal, reduciendo así la complejidad.

Antes de terminar nuestro estudio del algoritmo LS, consideremos una patología que puede surgir. La Figura 5.5 muestra una topología de red simple donde el coste de cada enlace es igual a la carga transportada por el enlace, reflejando, por ejemplo, el retardo que se experimentaría en ese enlace. En este ejemplo, los costes de los enlaces no son simétricos; es decir, $c(u,v)$ es igual a $c(v,u)$ solo si la carga transportada en ambas direcciones del enlace (u,v) es la misma. En este ejemplo, el nodo z origina una unidad de tráfico destinada a w, el nodo x también origina una unidad de

**b. x, y detectan una ruta mejor hacia w, en sentido horario****Figura 5.5** ♦ Oscilaciones con enrutamiento sensible a la congestión.

tráfico destinada a w , y el nodo y inyecta una cantidad de tráfico igual a e , también destinado a w . El enrutamiento inicial se muestra en la Figura 5.5(a), correspondiendo los costes de los enlaces a la cantidad de tráfico transportado.

Cuando se vuelve a ejecutar el algoritmo LS, el nodo y determina, basándose en los costes de enlace mostrados en la Figura 5.5(a), que la ruta en sentido horario a w tiene un coste de 1, mientras que la ruta en sentido antihorario a w (que era la que había estado utilizando) tiene un coste de $1 + e$. Por tanto, la ruta de coste mínimo de y a w ahora va en sentido horario. De forma similar, x determina que ahora su nueva ruta de coste mínimo a w va en sentido horario, dando como resultado los costes mostrados en la Figura 5.5(b). Cuando el algoritmo LS se ejecuta otra vez, los nodos x , y y z detectan una ruta de coste cero a w en el sentido antihorario y todos ellos envían su tráfico a las rutas en sentido antihorario. La siguiente vez que se ejecuta el algoritmo LS, x , y y z enrutan su tráfico a las rutas en sentido horario.

¿Qué podemos hacer para evitar tales oscilaciones (que pueden producirse en cualquier algoritmo, no solo el LS, que utilice una métrica de enlace basada en la congestión o el retardo)? Una solución sería obligar a que los costes de enlace no dependan de la cantidad de tráfico transportado (una solución inaceptable, ya que uno de los objetivos del enrutamiento es evitar los enlaces altamente congestionados; por ejemplo, los enlaces con un alto retardo). Otra solución consiste en garantizar que no todos los routers ejecuten el algoritmo LS al mismo tiempo. Esta parece una solución más razonable, ya que podríamos esperar que, aunque los routers ejecuten el algoritmo LS con la misma periodicidad, la instancia de ejecución del algoritmo no sería la misma en cada uno de los nodos. Es interesante observar que los investigadores han comprobado que los routers de Internet pueden auto-sincronizarse entre ellos [Floyd Synchronization 1994]. Es decir, incluso aunque inicialmente ejecuten el algoritmo con el mismo periodo pero en distintos instantes de tiempo, la instancia de ejecución del algoritmo puede llegar a sincronizarse en los routers y permanecer sincronizada. Una forma de evitar tal auto-sincronización es que cada router elija aleatoriamente el instante en el que enviar un anuncio de enlace.

Ahora que ya hemos estudiado el algoritmo LS, vamos a abordar el otro algoritmo de enrutamiento importante que se utiliza actualmente en la práctica: el algoritmo de enrutamiento por vector de distancias.

5.2.2 Algoritmo de enrutamiento por vector de distancias (DV)

Mientras que el algoritmo LS es un algoritmo que emplea información global, el **algoritmo por vector de distancias (DV)** es iterativo, asíncrono y distribuido. Es *distribuido* en el sentido de que cada nodo recibe información de uno o más de sus vecinos *directamente conectados*, realiza un cálculo y luego distribuye los resultados de su cálculo de vuelta a sus vecinos. Es *iterativo* porque este proceso continúa hasta que ya no se intercambia más información entre los vecinos. (Resulta interesante observar que el algoritmo también finaliza por sí mismo, es decir, no existe ninguna señal que indique que los cálculos deberían detenerse; simplemente se detienen.) El algoritmo es *asíncrono* en el sentido de que no requiere que todos los nodos operen sincronizados entre sí. Como tendremos oportunidad de ver, ¡un algoritmo asíncrono, iterativo, distribuido y que finaliza por sí mismo es mucho más interesante y divertido que un algoritmo centralizado!

Antes de presentar el algoritmo de vector de distancias, es conveniente que veamos una relación importante que existe entre los costes de las rutas de coste mínimo. Sea $d_x(y)$ el coste de la ruta de coste mínimo desde el nodo x al nodo y . Entonces, los costes mínimos están relacionados mediante la conocida ecuación de Bellman-Ford,

$$d_x(y) = \min_v \{c(x, v) + d_v(y)\} \quad (5.1)$$

donde \min_v se calcula para todos los vecinos de x . La ecuación de Bellman-Ford es bastante intuitiva. De hecho, después de viajar de x a v , si tomamos la ruta de coste mínimo de v a y , el coste de la ruta será $c(x, v) + d_v(y)$. Puesto que hay que comenzar viajando a algún vecino v , el coste mínimo de x a y será el mínimo de $c(x, v) + d_v(y)$, calculado para todos los vecinos v .

Pero para aquellos que sean escépticos en cuanto a la validez de la ecuación, vamos a comprobarla para el nodo de origen u y el nodo de destino z de la Figura 5.3. El nodo de origen u tiene tres vecinos: los nodos v , x y w . Recorriendo las distintas rutas del grafo, es fácil ver que $d_v(z) = 5$, $d_x(z) = 3$ y $d_w(z) = 3$. Introduciendo estos valores en la Ecuación 5.1, junto con los costes $c(u,v) = 2$, $c(u,x) = 1$ y $c(u,w) = 5$, se obtiene $d_u(z) = \min\{2 + 5, 5 + 3, 1 + 3\} = 4$, que obviamente es cierto y es exactamente lo que nos proporcionó el algoritmo de Dijkstra para la misma red. Esta rápida verificación debería ayudarle a disipar cualquier duda que pueda tener.

La ecuación de Bellman-Ford no es únicamente una curiosidad intelectual; realmente tiene una importancia práctica significativa. En concreto, la solución de la ecuación de Bellman-Ford proporciona las entradas de la tabla de reenvío del nodo x . Para ver esto, sea v^* cualquier nodo vecino que alcanza el mínimo dado por la Ecuación 5.1. Entonces, si el nodo x desea enviar un paquete al nodo y a lo largo de la ruta de coste mínimo, debería en primer lugar reenviar el paquete al nodo v^* . Por tanto, la tabla de reenvío del nodo x especificaría el nodo v^* como el router del siguiente salto para el destino final y . Otra importante contribución práctica de la ecuación de Bellman-Ford es que sugiere la forma en que tendrá lugar la comunicación vecino a vecino en el algoritmo de vector de distancias.

La idea básica es la siguiente: cada nodo x comienza con $D_x(y)$, una estimación del coste de la ruta de coste mínimo desde sí mismo al nodo y , para todos los nodos y de N . Sea $\mathbf{D}_x = [D_x(y): y \in N]$ el vector de distancias del nodo x , que es el vector de coste estimado desde x a todos los demás nodos y pertenecientes a N . Con el algoritmo de vector de distancias, cada nodo x mantiene la siguiente información de enrutamiento:

- Para cada vecino v , el coste $c(x,v)$ desde x al vecino directamente conectado v .
- El vector de distancias del nodo x , es decir, $\mathbf{D}_x = [D_x(y): y \in N]$, que contiene la estimación que x hace de su coste hacia todos los destinos y de N .
- Los vectores de distancias de cada uno de sus vecinos, es decir, $\mathbf{D}_v = [D_v(y): y \in N]$ para cada vecino v de x .

En el algoritmo asíncrono distribuido, cada nodo envía de vez en cuando una copia de su vector de distancias a cada uno de sus vecinos. Cuando un nodo x recibe un nuevo vector de distancias procedente de cualquiera de sus vecinos w , guarda dicho vector de w y luego utiliza la ecuación de Bellman-Ford para actualizar su propio vector de distancias como sigue:

$$D_x(y) = \min_v \{c(x, v) + D_v(y)\} \quad \text{para cada nodo } y \text{ perteneciente a } N$$

Si el vector de distancias del nodo x ha cambiado como resultado de este paso de actualización, entonces el nodo x enviará su vector de distancias actualizado a cada uno de sus vecinos, lo que puede a su vez actualizar sus propios vectores de distancias. De forma bastante milagrosa, siempre y cuando todos los nodos continúen intercambiando sus vectores de distancia de forma asíncrona, ¡cada coste estimado $D_x(y)$ converge a $d_x(y)$, el coste real de la ruta de coste mínimo del nodo x al nodo y [Bertsekas 1991]!

Algoritmo por vector de distancias (DV)

En cada nodo x :

```

1  Inicialización:
2  for todos los destinos  $y$  pertenecientes a  $N$ :
3       $D_x(y) = c(x,y)$  /* si  $y$  no es un vecino, entonces  $c(x,y) = \infty$  */
4  for cada vecino  $w$ 
5       $D_w(y) = ?$  for todos los destinos  $y$  pertenecientes a  $N$ 
6  for cada vecino  $w$ 
7      enviar vector de distancias  $\mathbf{D}_x = [D_x(y): y \in N]$  a  $w$ 
```

```

8
9 loop
10    wait (hasta ver una variación en el coste de enlace a un vecino w
11        o hasta recibir un vector de distancias de algún vecino w)
12
13    for cada y perteneciente a N:
14         $D_x(y) = \min_v \{c(x,v) + D_v(y)\}$ 
15
16 if  $D_x(y)$  ha variado para cualquier destino y
17     enviar vector de distancia  $\mathbf{D}_x = [D_x(y): y \text{ perteneciente a } N]$  a todos los vecinos
18
19 forever

```

En el algoritmo de vector de distancias, un nodo x actualiza la estimación de su vector de distancias si se produce un cambio en el coste de uno de sus enlaces directamente conectados o si recibe una actualización de un vector de distancias de algún vecino. Pero para actualizar su propia tabla de reenvío para un determinado destino y , lo que realmente necesita saber el nodo x no es la distancia de la ruta más corta a y , sino el nodo vecino $v^*(y)$, que es el router del siguiente salto a lo largo de la ruta más corta a y . Como es lógico, el router del siguiente salto $v^*(y)$ es el vecino v que alcanza el mínimo en la línea 14 del algoritmo DV. (Si existen varios vecinos v que alcanzan el mínimo, entonces $v^*(y)$ puede ser cualquiera de esos vecinos.) Por tanto, en las líneas 13–14, para cada destino y el nodo x también determina $v^*(y)$ y actualiza su tabla de reenvío para el destino y .

Recuerde que al algoritmo LS es un algoritmo centralizado, en el sentido de que requiere que cada nodo obtenga en primer lugar un mapa completo de la red antes de ejecutar el algoritmo de Dijkstra. El algoritmo DV es un algoritmo *descentralizado* y no utiliza dicha información global. De hecho, la única información que tendrá un nodo es el coste de los enlaces a los vecinos a los que está directamente conectado y la información que recibe de esos vecinos. Cada nodo espera a recibir una actualización de cualquier vecino (líneas 10–11), calcula su nuevo vector de distancias cuando recibe una actualización (línea 14) y distribuye su nuevo vector de distancias a sus vecinos (líneas 16–17). En la práctica, los algoritmos del tipo vector de distancias se utilizan en muchos protocolos de enrutamiento, entre los que se incluyen los protocolos RIP y BGP de Internet, ISO IDRP, Novell IPX y el ARPAnet original.

La Figura 5.6 ilustra el funcionamiento del algoritmo DV para el caso de la red simple de tres nodos mostrada en la parte superior de la figura. El funcionamiento del algoritmo se ilustra de manera síncrona, en la que todos los nodos reciben simultáneamente vectores de distancias de sus vecinos, calculan sus nuevos vectores de distancias e informan a sus vecinos si sus vectores de distancias han cambiado. Después de estudiar este ejemplo, puede comprobar usted mismo que el algoritmo también funciona correctamente en el modo asíncrono, es decir, efectuándose cálculos en los nodos y generándose y recibiéndose actualizaciones en cualquier instante.

La columna de más a la izquierda de la figura muestra tres **tablas de enrutamiento** iniciales para cada uno de los tres nodos. Por ejemplo, la tabla de la esquina superior izquierda corresponde a la tabla de enrutamiento inicial del nodo x . Dentro de una tabla de enrutamiento concreta, cada fila es un vector de distancias (específicamente, la tabla de enrutamiento de cada nodo incluye su propio vector de distancias y el de cada uno de sus vecinos). Por tanto, la primera fila de la tabla de enrutamiento inicial del nodo x es $\mathbf{D}_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$. La segunda y tercera filas de esta tabla son los vectores de distancias recibidos más recientemente de los nodos y y z , respectivamente. Puesto que durante la inicialización el nodo x no ha recibido nada aún del nodo y ni del z , las entradas de la segunda y de la tercera filas se inicializan con infinito.

Después de la inicialización, cada nodo envía su vector de distancias a cada uno de sus dos vecinos. Esto se indica en la Figura 5.6 mediante las flechas que salen de la primera columna de las tablas y van hasta la segunda columna. Por ejemplo, el nodo x envía su vector de distancias

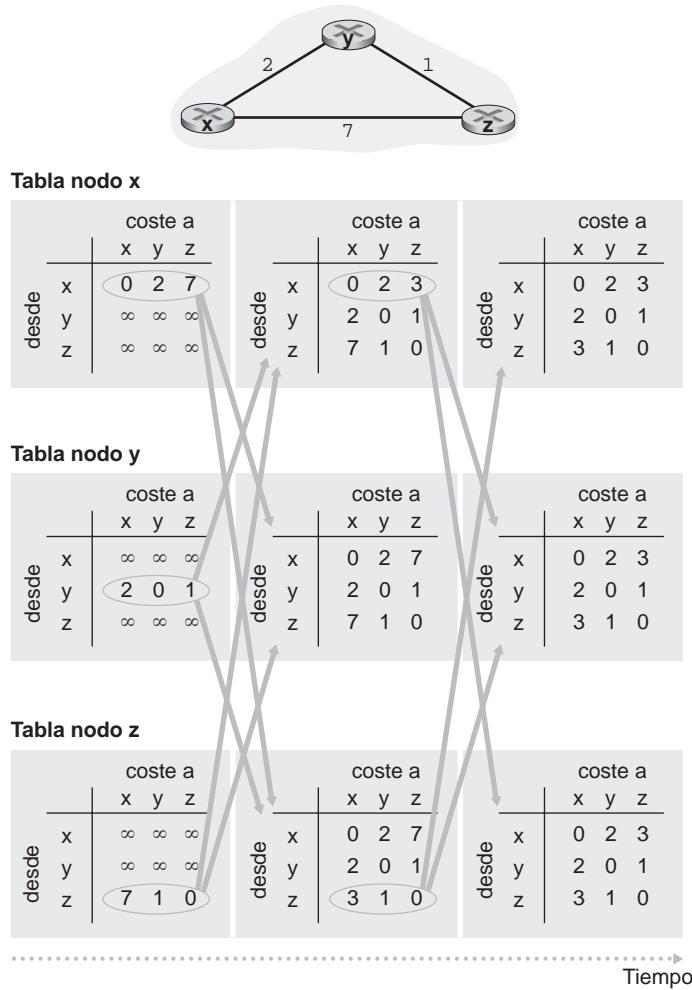


Figura 5.6 ♦ Algoritmo de vector de distancias (DV).

$D_x = [0, 2, 7]$ a los nodos y y z. Después de recibir las actualizaciones, cada nodo recalculará su propio vector de distancias. Por ejemplo, el nodo x calcula

$$D_x(x) = 0$$

$$D_x(y) = \min\{c(x, y) + D_y(y), c(x, z) + D_z(y)\} = \min\{2 + 0, 7 + 1\} = 2$$

$$D_x(z) = \min\{c(x, y) + D_y(z), c(x, z) + D_z(z)\} = \min\{2 + 1, 7 + 0\} = 3$$

Por tanto, la segunda columna muestra, para cada nodo, el nuevo vector de distancias del nodo, junto con los vectores de distancias que acaba de recibir de sus vecinos. Observe, por ejemplo, que la estimación del nodo x para el coste mínimo al nodo z, $D_x(z)$, ha cambiado de 7 a 3. Fíjese también en que para el nodo x, el nodo vecino y alcanza el mínimo en la línea 14 del algoritmo de vector de distancias; luego en esta etapa del algoritmo tenemos que, en el nodo x, $v^*(y) = y$ y $v^*(z) = y$.

Una vez que los nodos recalculan sus vectores de distancias, envían de nuevo los valores actualizados a sus vecinos (si se ha producido un cambio). Esto se indica en la Figura 5.6 mediante las flechas que van desde la segunda columna a la tercera columna de las tablas. Observe que únicamente los nodos x y z envían actualizaciones: el vector de distancias del nodo y no ha cambiado, por lo que no envía ninguna actualización. Después de recibir las actualizaciones, los nodos recalculan sus vectores de distancias y actualizan sus tablas de enrutamiento, lo que se muestra en la tercera columna.

El proceso de recibir vectores de distancias actualizados de los vecinos, recalcular las entradas de la tabla de enrutamiento e informar a los vecinos de los costes modificados de la ruta de coste mínimo hacia un destino continúa hasta que ya no se envían mensajes de actualización. En esta situación, puesto que no se envían mensajes de actualización, no se realizarán más cálculos de la tabla de enrutamiento y el algoritmo entrará en un estado de reposo; es decir, todos los nodos se encontrarán realizando la espera indicada por las líneas 10–11 del algoritmo del vector de distancias. El algoritmo permanece en el estado de reposo hasta que el coste de un enlace cambia, como se explica a continuación.

Algoritmo de vector de distancias: cambios en el coste de los enlaces y fallo de los enlaces

Cuando un nodo que ejecuta el algoritmo DV detecta un cambio en el coste del enlace desde sí mismo a un vecino (líneas 10–11), actualiza su vector de distancias (líneas 13–14) y, si existe un cambio en el coste de la ruta de coste mínimo, informa a sus vecinos (líneas 16–17) de su nuevo vector de distancias. La Figura 5.7(a) ilustra un escenario en el que el coste del enlace de y a x cambia de 4 a 1. Aquí vamos a centrarnos únicamente en las entradas de la tabla de distancias de y y z al destino x . El algoritmo de vector de distancias da lugar a la siguiente secuencia de sucesos:

- En el instante t_0 , y detecta el cambio en el coste del enlace (el coste ha cambiado de 4 a 1), actualiza su vector de distancias e informa a sus vecinos de este cambio, puesto que su vector de distancias ha cambiado.
- En el instante t_1 , z recibe la actualización de y y actualiza su tabla. Calcula el nuevo coste mínimo a x (ha disminuido de 5 a 2) y envía su nuevo vector de distancias a sus vecinos.
- En el instante t_2 , y recibe la actualización de z y actualiza su tabla de distancias. El coste mínimo de y no cambia y, por tanto, y no envía ningún mensaje a z . El algoritmo entra en el estado de reposo.

Así, solo se han necesitado dos iteraciones para que el algoritmo DV alcance un estado de reposo. Las buenas noticias acerca de la disminución del coste entre x e y se han propagado rápidamente a través de la red.

Consideremos ahora lo que ocurre cuando el coste de un enlace *aumenta*. Suponga que el coste del enlace entre x e y aumenta de 4 a 60, como se muestra en la Figura 5.7(b).

1. Antes de que el coste del enlace varíe, $D_y(x) = 4$, $D_y(z) = 1$, $D_z(y) = 1$ y $D_z(x) = 5$. En el instante t_0 y detecta el cambio en el coste del enlace (el coste ha variado de 4 a 60). El nodo y calcula su nueva ruta de coste mínimo a x , obteniendo un coste de:

$$D_y(x) = \min\{c(y, x) + D_x(x), c(y, z) + D_z(x)\} = \min\{60 + 0, 1 + 5\} = 6$$

Por supuesto, con nuestra visión global de la red, podemos ver que este nuevo coste a través de z es *erróneo*. Pero la única información que tiene el nodo y es que su coste directo a x es 60 y que z le ha dicho a y que z podría alcanzar x con un coste de 5. Por tanto, para llegar a x , y ahora

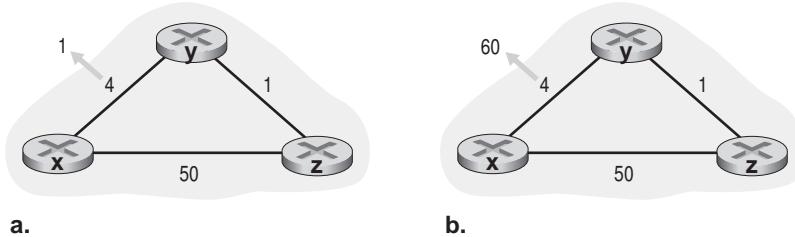


Figura 5.7 ♦ Cambios en el coste del enlace.

enrutaría a través de z , confiando plenamente en que z será capaz de llegar hasta x con un coste de 5. En t_1 tenemos por tanto un **bucle de enrutamiento** (para llegar a x , y enruta a través de z , y z enruta a través de y). Un bucle de enrutamiento es como un agujero negro (un paquete destinado a x que llega a y o z en t_1 rebotará entre estos dos nodos permanentemente, o hasta que las tablas de reenvío cambien).

2. Dado que el nodo y ha calculado un nuevo coste mínimo a x , informa a z de este nuevo vector de distancias en el instante t_1 .
3. En algún momento posterior a t_1 , z recibe un nuevo vector de distancias de y , que indica que el coste mínimo de y a x es 6. z sabe que puede llegar a y con un coste de 1 y, por tanto, calcula un nuevo coste mínimo a x que será igual a $D_z(x) = \min\{50 + 0, 1 + 6\} = 7$. Puesto que el coste mínimo de z a x ha aumentado, a continuación informa a y de su nuevo vector de distancias en t_2 .
4. De forma similar, después de recibir el nuevo vector de distancias de z , el nodo y determina que $D_y(x) = 8$ y envía a z su vector de distancias. El nodo z determina entonces que $D_z(x) = 9$ y envía a y su vector de distancias, y así sucesivamente.

¿Durante cuánto tiempo continuará el proceso? Puede comprobar por sí mismo que el bucle se mantendrá durante 44 iteraciones (intercambios de mensajes entre y y z), hasta que z finalmente calcule que el coste de su ruta a través de y es mayor que 50. En ese punto, z (¡finalmente!) determinará que su ruta de coste mínimo a x es a través de su conexión directa con x . El nodo y entonces enrutará hacia x a través de z . ¡Como puede ver, las malas noticias acerca del aumento del coste del enlace han tardado mucho en propagarse! ¿Qué habría ocurrido si el coste del enlace $c(y, x)$ hubiera cambiado de 4 a 10.000 y el coste $c(z, x)$ hubiera sido 9.999? A causa de los escenarios de este tipo, en ocasiones se designa a este problema con el nombre de problema de la cuenta hasta infinito.

Algoritmo de vector de distancias: adición de la inversa envenenada

El escenario concreto de bucle que acabamos de describir puede evitarse utilizando una técnica conocida como *inversa envenenada (poisoned reverse)*. La idea es simple: si z enruta a través de y para llegar al destino x , entonces z anunciará a y que $D_z(x) = \infty$ (incluso aunque z sepa que, en realidad, $D_z(x) = 5$). El nodo z continuará contando esta pequeña mentira a y mientras continúe enrutando hacia x a través de y . Dado que y cree que z no dispone de una ruta hacia x , el nodo y nunca intentará enrutar hacia x a través de z , mientras z continúe enrutando hacia x a través de y (y mintiendo acerca de ello).

Veamos ahora cómo la inversa envenenada resuelve el problema concreto del bucle encontrado antes, en la Figura 5.5(b). Como resultado de la inversa envenenada, la tabla de distancias de y indica que $D_z(x) = \infty$. Cuando el coste del enlace (x, y) cambia de 4 a 60 en el instante t_0 , el nodo y actualiza su tabla y continúa enrutando directamente a x , aunque a un coste mayor, de 60, e informa a z de su nuevo coste a x , es decir, $D_y(x) = 60$. Después de recibir la actualización en t_1 , z cambia inmediatamente su ruta a x para que sea a través del enlace directo (z, x) con un coste de 50. Puesto que esta es la nueva ruta de coste mínimo a x , y dado que la ruta ya no pasa a través de y , ahora z informa en t_2 a y de que $D_z(x) = 50$. Después de recibir la actualización de z , el nodo y actualiza su tabla de distancias con $D_y(x) = 51$. Además, dado que ahora z está en la ruta de coste mínimo de y a x , el nodo y envenena la ruta inversa de z a x , informando a z en el instante t_3 de que $D_y(x) = \infty$ (aunque y sepa que, en realidad, $D_y(x) = 51$).

¿Resuelve la inversa envenenada el problema general de la cuenta hasta infinito? No. El lector puede comprobar por sí mismo que los bucles que implican a tres o más nodos (en lugar de simplemente a dos nodos vecinos) no serán detectados por la técnica de la inversa envenenada.

Comparación de los algoritmos de enrutamiento LS y DV

Los algoritmos de vector de distancias y de estado de enlaces utilizan métodos complementarios para el cálculo de las rutas. Con el algoritmo de vector de distancias, cada nodo *solo* se comunica

con sus vecinos directamente conectados, pero les proporciona sus estimaciones de coste mínimo desde sí mismo a *todos* los demás nodos (conocidos) de la red. El algoritmo de estado de enlaces requiere información global. En consecuencia, cuando se lo implemente en todos y cada uno de los routers, como por ejemplo en las Figuras 4.2 y 5.1, cada nodo tendrá que comunicarse con todos los restantes nodos (vía difusión), pero solo les informará de los costes de sus enlaces directamente conectados. Vamos a terminar este estudio sobre los algoritmos de estado de enlaces y de vector de distancias haciendo una rápida comparación de algunos de sus atributos. Recuerde que N es el conjunto de nodos (routers) y E es el conjunto de aristas (enlaces).

- *Complejidad del mensaje.* Hemos visto que el algoritmo LS requiere que cada nodo conozca el coste de cada enlace de la red. Esto requiere el envío de $O(|N| |E|)$ mensajes. Además, cuando el coste de un enlace cambia, el nuevo coste tiene que enviarse a todos los nodos. El algoritmo de vector de distancias requiere intercambios de mensajes entre los vecinos directamente conectados en cada iteración. Hemos visto que el tiempo necesario para que el algoritmo converja puede depender de muchos factores. Cuando los costes de los enlaces cambian, el algoritmo de vector de distancias propagará los resultados del coste del enlace que ha cambiado solo si el nuevo coste de enlace da lugar a una ruta de coste mínimo distinta para uno de los nodos conectados a dicho enlace.
- *Velocidad de convergencia.* Hemos visto que nuestra implementación del algoritmo de estado de enlaces es un algoritmo $O(|N|^2)$ que requiere enviar $O(|N| |E|)$ mensajes. El algoritmo de vector de distancias puede converger lentamente y pueden aparecer bucles de enrutamiento mientras está convergiendo. Este algoritmo también sufre el problema de la cuenta hasta infinito.
- *Robustez.* ¿Qué puede ocurrir si un router falla, funciona mal o es saboteado? Con el algoritmo de estado de enlaces, un router podría difundir un coste incorrecto para uno de sus enlaces conectados (pero no para los otros). Un nodo también podría corromper o eliminar cualquier paquete recibido como parte de un mensaje de difusión LS. Pero, con el algoritmo LS, un nodo solo calcula su propia tabla de reenvío, mientras que otros nodos realizan cálculos similares por sí mismos. Esto significa que los cálculos de rutas son hasta cierto punto independientes en LS, proporcionando un mayor grado de robustez. Con el algoritmo de vector de distancias, un nodo puede anunciar rutas de coste mínimo incorrectas a uno o a todos los destinos. (De hecho, en 1997, un router que funcionaba mal en un pequeño ISP proporcionó a los routers troncales nacionales información de enrutamiento errónea. Esto hizo que otros routers inundaran con una gran cantidad de tráfico al router que funcionaba mal y provocó que amplias partes de Internet estuvieran desconectadas durante varias horas [Neumann 1997].) En un sentido más general, observamos que, en cada iteración, los cálculos de un nodo con el algoritmo de vector de distancias se pasan a sus vecinos y luego, indirectamente, al vecino del vecino en la siguiente iteración. En este sentido, con el algoritmo de vector de distancias, un cálculo de nodo incorrecto puede difundirse a través de toda la red.

En resumen, ningún algoritmo es el ganador evidente; de hecho, ambos algoritmos se utilizan en Internet.

5.3 Enrutamiento dentro de un sistema autónomo en Internet: OSPF

En nuestro estudio de los algoritmos de estado de enlaces y de vector de distancias, hemos visto la red simplemente como una colección de routers interconectados. Un router era indistinguible de otro, en el sentido de que los routers ejecutaban el mismo algoritmo de enrutamiento para calcular las rutas a través de la red completa. En la práctica, este modelo y la imagen de un conjunto homogéneo de routers que ejecutan todos ellos el mismo algoritmo de enrutamiento es un poco simplista, por al menos dos razones importantes:

- *Escala.* Cuando el número de routers comienza a hacerse grande, la sobrecarga implicada en la comunicación, cálculo y almacenamiento de la información de enrutamiento se hace prohibitiva. Actualmente, Internet consta de cientos de millones de routers. Almacenar la información de enrutamiento para todos los posibles destinos en cada uno de estos routers evidentemente requeriría enormes cantidades de memoria. ¡La sobrecarga requerida para difundir las actualizaciones de conectividad y de coste de los enlaces entre todos los routers sería inmensa! Seguramente, un algoritmo de vector de distancias que iterara entre tal enorme cantidad de routers nunca llegaría a converger. Evidentemente, es preciso hacer algo para reducir la complejidad del cálculo de rutas en una red tan grande como Internet.
- *Autonomía administrativa.* Como se describe en la Sección 1.3, Internet es una red de ISPs, estando compuesto cada ISP por su propia red de routers. Un ISP generalmente desea operar su red a su antojo (por ejemplo, emplear cualquier algoritmo de enrutamiento que elija dentro de su red) u ocultar al mundo exterior ciertos aspectos de la organización interna de su red. Idealmente, una organización debería poder operar y administrar su red como desee, sin por ello dejar de conectar su red a otras redes externas.

Estos dos problemas pueden resolverse organizando los routers en **sistemas autónomos (AS, Autonomous System)**, estando cada AS formado por un grupo de routers que se encuentran bajo el mismo control administrativo. A menudo, los routers de un ISP, y los enlaces que los interconectan, constituyen un único AS. Otros ISP, sin embargo, dividen su red en múltiples AS. En particular, algunos ISP de nivel 1 utilizan un solo AS gigante para toda su red, mientras que otros ISP dividen su red en decenas de AS interconectados. Cada sistema autónomo se identifica mediante su número de sistema autónomo (ASN, *Autonomous System Number*), que es único globalmente [RFC 1930]. Los números de AS, como las direcciones IP, son asignados por los registros regionales de ICANN [ICANN 2016].

Los routers de un mismo AS ejecutan todos ellos el mismo algoritmo de enrutamiento y disponen de información unos de otros. El algoritmo de enrutamiento que se ejecuta dentro de un sistema autónomo se denomina **protocolo de enrutamiento interno del sistema autónomo**.

OSPF

El enrutamiento OSPF (*Open Shortest Path First*, protocolo abierto de preferencia para la ruta más corta) y su pariente próximo, IS-IS, se utilizan ampliamente para el enrutamiento interno a los sistemas autónomos en Internet. El ‘Open’ de OSPF indica que la especificación del protocolo de enrutamiento está disponible públicamente (a diferencia, por ejemplo, del protocolo EIGRP de Cisco, que solo recientemente se ha convertido en abierto [Savage 2015], después de ser durante unos 20 años un protocolo propiedad de Cisco). La versión más reciente de OSPF, la versión 2, está definida en [RFC 2328], un documento público.

OSPF es un protocolo de estado de enlaces que utiliza la técnica de inundación de información de estado de los enlaces y un algoritmo de la ruta de coste mínimo de Dijkstra. Con OSPF, un router construye un mapa topológico completo (es decir, un grafo) del sistema autónomo entero. A continuación, cada router ejecuta localmente el algoritmo de la ruta más corta de Dijkstra para determinar un árbol de rutas más cortas a todas las *subredes*, con él mismo como nodo raíz. El administrador de la red configura los costes de los enlaces individuales (consulte el recuadro En la práctica: configuración de los pesos de los enlaces en OSPF). El administrador puede decidir hacer igual a 1 el coste de todos los enlaces, proporcionando así un enrutamiento con un número mínimo de saltos, o puede definir los pesos de los enlaces para que sean inversamente proporcionales a la capacidad de los mismos, con el fin de disuadir al tráfico de utilizar los enlaces con poco ancho de banda. OSPF no establece una política para definir el peso de los enlaces (esta tarea le corresponde al administrador de la red), sino que proporciona el mecanismo (el protocolo) para determinar el enrutamiento de coste mínimo para el conjunto dado de pesos de los enlaces.



EN LA PRÁCTICA

CONFIGURACIÓN DE LOS PESOS DE LOS ENLACES EN OSPF

En nuestra exposición sobre el enrutamiento de estado de enlaces hemos supuesto implícitamente que los pesos de los enlaces están definidos, que se está ejecutando un algoritmo de enrutamiento como OSPF y que el tráfico fluye de acuerdo con las tablas de enrutamiento calculadas por el algoritmo LS. En términos de causa y efecto, los pesos de los enlaces nos vienen dados (es decir, se conocen de antemano) y dan como resultado (mediante el algoritmo de Dijkstra) las rutas que minimizan el coste global. Desde este punto de vista, los pesos de los enlaces reflejan el coste de utilizar un enlace (por ejemplo, si los pesos son inversamente proporcionales a la capacidad, entonces los enlaces de alta capacidad tendrían asociados pesos más pequeños y, por tanto, serían más atractivos desde el punto de vista del enrutamiento) y el algoritmo de Dijkstra sirve para minimizar el coste global.

En la práctica, la relación causa-efecto entre el peso de los enlaces y las rutas puede invertirse, cuando los operadores de red configuran los pesos de los enlaces de manera que se obtengan rutas que permitan alcanzar determinados objetivos de ingeniería de tráfico [Fortz 2000, Fortz 2002]. Por ejemplo, suponga que un operador de red tiene una estimación del flujo de tráfico que entra en la red por cada punto de entrada y que está destinado a cada punto de salida. El operador podría querer entonces implementar un enrutamiento específico para los flujos de entrada-a-salida que minimice la tasa máxima de utilización de los enlaces de la red. Pero con un algoritmo de enrutamiento como OSPF, la principal herramienta de la que dispone el operador para alterar el enrutamiento de los flujos a través de la red son los pesos de los enlaces. Por tanto, para alcanzar el objetivo de minimizar la tasa máxima de utilización de los enlaces, el operador tiene que encontrar el conjunto de pesos de los enlaces que permita alcanzar este objetivo. Esto constituye una inversión de la relación causa-efecto: el enrutamiento deseado de los flujos es conocido y tienen que determinarse los pesos de los enlaces OSPF de manera que el algoritmo de enrutamiento OSPF dé como resultado el enrutamiento de flujos deseado.

Con OSPF, un router difunde la información de enrutamiento a *todos* los demás routers del sistema autónomo, no solo a sus routers vecinos. Un router difunde la información de estado de los enlaces cuando se produce un cambio en el estado de un enlace (por ejemplo, un cambio en el coste o en su estado activo/inactivo). También difunde periódicamente el estado de un enlace (al menos una vez cada 30 minutos), incluso aunque el estado del mismo no haya cambiado. El documento RFC 2328 destaca que “esta actualización periódica de los anuncios del estado de los enlaces añade robustez al algoritmo LS”. Los anuncios OSPF están contenidos en mensajes OSPF que son transportados directamente por IP, siendo el número del protocolo de la capa superior para OSPF igual a 89. Por tanto, el protocolo OSPF tiene que implementar por sí mismo funcionalidades tales como la de transferencia fiable de mensajes y la de envío de mensajes de difusión acerca del estado de los enlaces. El protocolo OSPF también comprueba que los enlaces estén operativos (mediante un mensaje HELLO que se envía a un vecino conectado) y permite al router OSPF obtener de un vecino la base de datos de estado de los enlaces de toda la red.

Algunas de las funcionalidades avanzadas incluidas en OSPF son las siguientes:

- *Seguridad.* Los intercambios entre routers OSPF (por ejemplo, actualizaciones de estado de los enlaces) pueden ser autenticados. Con la autenticación, solo pueden participar en el protocolo OSPF los routers de confianza del sistema autónomo, impidiendo así que intrusos maliciosos (o estudiantes de redes que apliquen sus conocimientos recién adquiridos sin permiso) inyecten información incorrecta en las tablas de los routers. De manera predeterminada, los paquetes OSPF entre routers no son autenticados y podrían ser alterados. Pueden configurarse dos tipos de autenticación: simple y MD5 (consulte el Capítulo 8 para obtener información sobre MD5 y la autenticación en general). Con la autenticación simple, se configura la misma contraseña en todos los routers. Cuando un router envía un paquete OSPF, incluye la contraseña en texto plano

(“legible”). Evidentemente, la autenticación simple no es muy segura. La autenticación MD5 está basada en claves secretas compartidas que están configuradas en todos los routers. Para cada paquete OSPF que se envía, el router calcula el hash MD5 del contenido del paquete OSPF, al que se añade la clave secreta (consulte el Capítulo 8 para ver una explicación acerca de los códigos de autenticación de mensajes). A continuación, el router incluye el valor hash resultante en el paquete OSPF. El router receptor, utilizando la clave secreta preconfigurada, calculará un hash MD5 del paquete y lo comparará con el valor hash que transporta el paquete, verificando de este modo la autenticidad del mismo. En la autenticación MD5 también se utilizan números de secuencia para protegerse frente a ataques por repetición.

- *Varias rutas de igual coste.* Cuando varias rutas a un destino tienen el mismo coste, OSPF permite utilizar varias rutas (es decir, no es necesario elegir una misma ruta para transportar todo el tráfico cuando existen varias rutas con igual coste).
- *Soporte integrado para enrutamiento por unidifusión y por multidifusión.* OSPF multidifusión (MOSPF, *Multicast OSPF*) [RFC 1584] añade extensiones simples a OSPF para permitir el enrutamiento por multidifusión. MOSPF utiliza la base de datos de enlaces OSPF existente y añade un nuevo tipo de anuncio de estado de enlaces al mecanismo de difusión de estado de los enlaces de OSPF.
- *Soporte para definir una jerarquía dentro de un mismo AS.* Un sistema autónomo OSPF puede configurarse jerárquicamente en áreas. Cada área ejecuta su propio algoritmo OSPF de enruteamiento por estado de enlaces, encargándose cada router de un área de difundir su información de estado de enlaces a todos los restantes routers del área. Dentro de un área, uno o más routers de frontera de área son responsables de enrutar los paquetes hacia fuera del área. Por último, una y solo una de las áreas OSPF del AS se configura como área troncal. El papel principal del área troncal es enrutar el tráfico entre las restantes áreas del AS. El área troncal es la que contiene siempre todos los routers de frontera de área del AS y puede contener también routers que no sean de frontera. El enrutamiento entre áreas dentro del AS requiere que el paquete se enrute primero hacia router de frontera de área (enrutamiento interno al área), que luego se enrute a través del área troncal hasta el router de frontera perteneciente al área de destino y que después se enrute hasta su destino final.

OSPF es un protocolo relativamente complejo, por lo que aquí lo hemos cubierto de forma necesariamente breve; en [Huitema 1998; Moy 1998; RFC 2328] se proporcionan detalles adicionales.

5.4 Enrutamiento entre los ISP: BGP

Acabamos de ver que OSPF es un ejemplo de protocolo de enrutamiento dentro de un sistema autónomo. Cuando se enruta un paquete entre un origen y un destino que se encuentran en un mismo sistema autónomo, la ruta que el paquete sigue está completamente determinada por el protocolo de enrutamiento dentro del sistema autónomo. Sin embargo, para enrutar un paquete entre múltiples sistemas autónomos (por ejemplo, desde un teléfono inteligente en Tombuctú hasta un servidor situado en un centro de datos de Silicon Valley, California), necesitamos un **protocolo de enrutamiento entre sistemas autónomos**. Puesto que un protocolo de enrutamiento entre sistemas autónomos requiere la coordinación de múltiples AS, los sistemas autónomos que se estén comunicando deberán ejecutar el mismo protocolo de enrutamiento entre sistemas autónomos. De hecho, en Internet, todos los AS utilizan el mismo protocolo de enrutamiento entre sistemas autónomos, denominado *Border Gateway Protocol*, más conocido como **BGP** [RFC 4271; Stewart 1999].

Se podría sostener que BGP es el más importante de todos los protocolos de Internet (el único otro competidor sería el protocolo IP que hemos estudiado en la Sección 4.3), ya que es el que une a los miles de ISP existentes en Internet. Como pronto veremos, BGP es un protocolo descentralizado

y asíncrono, en la línea del enrutamiento por vector de distancias descrito en la Sección 5.2.2. Aunque BGP es un protocolo complejo y difícil, si queremos comprender en profundidad Internet necesitamos familiarizarnos con sus principios fundamentales y su funcionamiento. El tiempo que dediquemos a aprender BGP estará bien invertido.

5.4.1 El papel de BGP

Para comprender las responsabilidades de BGP, pensemos en un sistema autónomo y en un router cualquiera dentro de ese sistema autónomo. Recuerde que todo router dispone de una tabla de reenvío, que juega el papel central en el proceso de reenviar los paquetes entrantes hacia los enlaces de salida del router. Como hemos visto, para los destinos que se encuentran dentro del mismo AS, las entradas de la tabla de reenvío del router están determinadas por el protocolo de enrutamiento dentro del sistema autónomo. ¿Pero qué ocurre con los destinos que se encuentran fuera del AS? Es precisamente aquí donde BGP viene al rescate.

En BGP, los paquetes no se enrutan hacia una dirección de destino específica, sino hacia prefijos CIDR, representando cada prefijo a una subred o a una colección de subredes. En el mundo de BGP, un destino puede tener la forma 138.16.68/22, que en este ejemplo incluiría 1.024 direcciones IP. Por tanto, la tabla de reenvío de un router dispondrá de entradas de la forma (x, I) , donde x es un prefijo (como por ejemplo 138.16.68/22) e I es el número de interfaz de una de las interfaces del router.

Como protocolo de enrutamiento entre sistemas autónomos, BGP proporciona a cada router mecanismos para:

1. *Obtener de los sistemas autónomos vecinos información acerca de la alcanzabilidad de los prefijos.* En particular, BGP permite a cada subred anunciar su existencia al resto de Internet. Una subred vocifera “Existo y estoy aquí”, y BGP garantiza que todos los routers de Internet sepan sobre ella. Si no fuera por BGP, las subredes estarían aisladas, resultando desconocidas e inalcanzables para el resto de Internet.
2. *Determinar las “mejores” rutas hacia los distintos prefijos.* Un router puede llegar a conocer dos o más rutas hacia un prefijo específico. Para determinar la mejor ruta, el router ejecutará localmente un procedimiento de selección de rutas de BGP (utilizando la información de alcanzabilidad de prefijos que ha obtenido de los routers vecinos). La mejor ruta se determinará basándose tanto en las políticas existentes, como en la información de alcanzabilidad.

Veamos ahora cómo lleva a cabo BGP estas dos tareas.

5.4.2 Anuncio de la información de rutas BGP

Considere la red mostrada en la Figura 5.8. Como se puede ver, esta sencilla red tiene tres sistemas autónomos: AS1, AS2 y AS3. En la figura se muestra que AS3 incluye una subred con prefijo x . En un sistema autónomo determinado, cada router puede ser un **router de pasarela** o un **router interno**. Un router de pasarela es aquel que está situado en la frontera de un sistema autónomo y se conecta directamente a uno o más routers de otros sistemas autónomos. Un router interno solo está conectado a hosts y routers pertenecientes a su propio sistema autónomo. En AS1, por ejemplo, el router 1c es un router de pasarela; los routers 1a, 1b y 1d son routers internos.

Pensemos en la tarea de anunciar la información de alcanzabilidad del prefijo x a todos los routers mostrados en la Figura 5.8. A alto nivel, esta tarea resulta sencilla. En primer lugar, AS3 envía un mensaje BGP a AS2, informándole de que x existe y de que se encuentra en AS3; llamemos a este mensaje “AS3 x ”. A continuación, AS2 envía un mensaje BGP a AS1, informándole de que x existe y de que se puede llegar a x pasando primero por AS2 y luego yendo a AS3; llamemos a este mensaje “AS2 AS3 x ”. De esta forma, los distintos sistemas autónomos no solo aprenderán que x existe, sino también una ruta de sistemas autónomos que conduce hasta x .

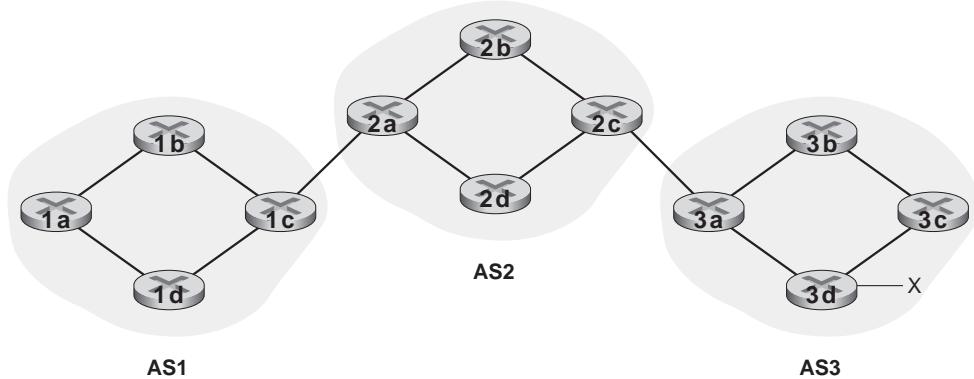


Figura 5.8 ♦ Red con tres sistemas autónomos. AS3 incluye una subred con prefijo x.

Aunque la explicación del párrafo anterior sobre el anuncio de información de alcanzabilidad BGP transmite correctamente la idea general, no es una explicación precisa, en el sentido de que no son los sistemas autónomos los que se intercambian mensajes, sino que en realidad quien se los intercambia son los routers. Para entender este punto, volvamos a examinar el ejemplo de la Figura 5.8. En BGP, las parejas de routers intercambian información de enrutamiento a través de conexiones TCP semipermanentes, utilizando el puerto 179. Cada una de esas conexiones TCP, junto con todos los mensajes BGP enviados a través de la conexión, se denomina **conexión BGP**. Además, una conexión BGP que abarca dos sistemas autónomos se llama conexión **BGP externa (eBGP)**, mientras que una sesión BGP entre routers de un mismo sistema autónomo se denomina conexión **BGP interna (iBGP)**. En la Figura 5.9 se muestran ejemplos de conexiones BGP para la red de la Figura 5.8. Normalmente habrá una conexión eBGP por cada enlace que conecte directamente routers de pasarela situados en diferentes sistemas autónomos; así, en la Figura 5.9 hay una conexión eBGP entre los routers de pasarela 1c y 2a, y otra conexión eBGP entre los routers de pasarela 2c y 3a.

Existen también conexiones iBGP entre los routers que forman cada uno de los sistemas autónomos. En particular, la Figura 5.9 muestra una configuración común, en la que hay una conexión BGP por cada pareja de routers interna a un sistema autónomo, creando una malla de conexiones

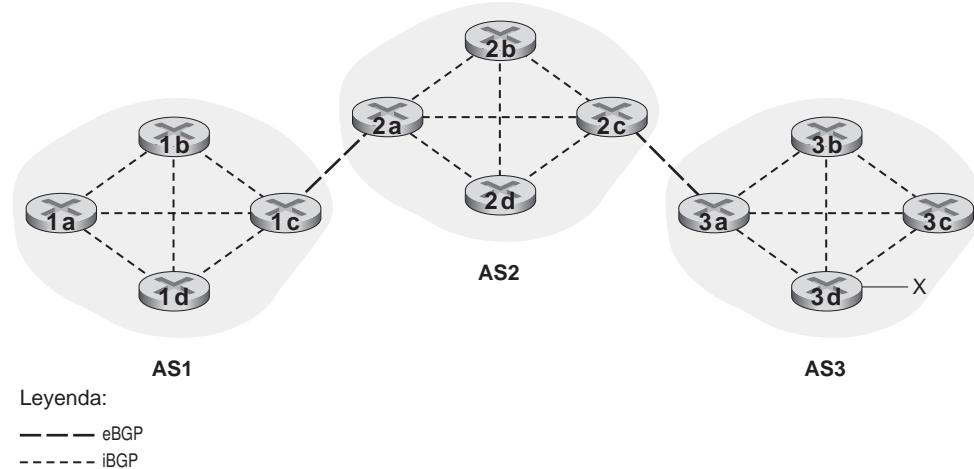


Figura 5.9 ♦ Conexiones eBGP e iBGP.

TCP dentro de cada AS. En la Figura 5.9, las conexiones eBGP se muestran con punteado más largo; las conexiones iBGP se ilustran mediante punteado más corto. Observe que las conexiones iBGP no siempre se corresponden con enlaces físicos.

Para propagar la información de alcanzabilidad se utilizan sesiones tanto iBGP como eBGP. Consideremos de nuevo la tarea de anunciar la información de alcanzabilidad para el prefijo x a todos los routers de AS1 y AS2. En este proceso, el router de pasarela 3a envía primero un mensaje eBGP “AS3 x” al router de pasarela 2c. A continuación, el router de pasarela 2c envía el mensaje iBGP “AS3 x” a todos los restantes routers de AS2, incluyendo el router de pasarela 2a. Despues, el router 2a envía el mensaje eBGP “AS2 AS3 x” al router de pasarela 1c. Finalmente, el router de pasarela 1c utiliza iBGP para enviar el mensaje “AS2 AS3 x” a todos los routers de AS1. Tras completar este proceso, todos los routers de AS1 y AS2 conocen la existencia de x y conocen también una ruta de sistemas autónomos que conduce hasta x.

Por supuesto, en una red real puede haber muchas rutas diferentes desde un cierto router hasta un destino determinado, cada una de ellas a través de una secuencia distinta de sistemas autónomos. Considere, por ejemplo, la red de la Figura 5.10, que es la red original de la Figura 5.8, pero con un enlace físico adicional que une el router 1d con el router 3d. En este caso, hay dos rutas desde AS1 hasta x: la ruta “AS2 AS3 x” a través del router 1c, y la nueva ruta “AS3 x” a través del router 1d.

5.4.3 Determinación de las mejores rutas

Como acabamos de ver, puede haber muchas rutas desde un cierto router hasta una subred de destino. En Internet, de hecho, los routers reciben a menudo información de alcanzabilidad acerca de docenas de diferentes rutas posibles. ¿Cómo elige un router entre estas rutas (y luego configura correspondientemente su tabla de reenvío)?

Antes de entrar en esta cuestión crítica, necesitamos presentar algo más de terminología BGP. Cuando un router anuncia un prefijo a través de una conexión BGP, incluye con el prefijo varios **atributos BGP**. En la jerga de BGP, un prefijo junto con sus atributos se denomina **ruta**. Dos de los atributos más importantes son AS-PATH y NEXT-HOP. El atributo AS-PATH contiene la lista de sistemas autónomos a través de los que ha pasado el anuncio, como hemos visto en los ejemplos anteriores. Para generar el valor AS-PATH, cuando se pasa un prefijo a un sistema autónomo, el sistema añade su ASN a la lista contenida en AS-PATH. Por ejemplo, en la Figura 5.10 hay dos rutas desde AS1 hasta la subred x: una que utiliza el valor AS-PATH “AS2 AS3” y otra que emplea el valor AS-PATH “AS3”. Los routers BGP también utilizan el atributo AS-PATH para detectar e impedir los bucles de anuncio; en concreto, si un router ve que su propio sistema autónomo está en la lista de la ruta, rechazará el anuncio.

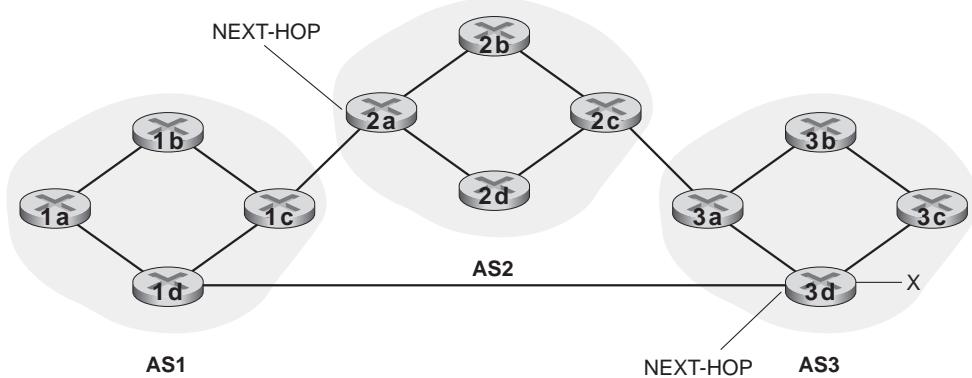


Figura 5.10 ♦ Red ampliada con un enlace directo entre AS1 y AS3.

El atributo NEXT-HOP proporciona el enlace crítico entre el protocolo de enrutamiento interno del sistema autónomo y el protocolo de enrutamiento entre sistemas autónomos, y tiene un sutil aunque importante uso. El siguiente salto (NEXT-HOP) es la dirección IP de la interfaz de router que inicia la secuencia de sistemas autónomos (AS-PATH). Para comprender el uso de este atributo, vamos a hacer referencia de nuevo a la Figura 5.10. Como se indica en ella, el atributo NEXT-HOP para la ruta “AS2 AS3 x”, que va de AS1 hasta x y pasa por AS2, es la dirección IP de la interfaz izquierda del router 2a. El atributo NEXT-HOP para la ruta “AS3 x”, que va de AS1 hasta x y que evita pasar por AS2, es la dirección IP de la interfaz izquierda del router 3d. En resumen, en este sencillo ejemplo, cada router de AS1 aprende que existen dos rutas BGP hacia el prefijo x:

Dirección IP de la interfaz izquierda del router 2a; AS2 AS3; x
Dirección IP de la interfaz izquierda del router 3d; AS3; x

Aquí, hemos escrito cada ruta BGP en forma de lista con tres componentes: NEXT-HOP; AS-PATH; prefijo de destino. En la práctica, una ruta BGP incluye atributos adicionales, que por el momento vamos a ignorar. Observe que el atributo NEXT-HOP es una dirección IP de un router que *no* pertenece a AS1; sin embargo, la subred que contiene esta dirección IP está directamente conectada a AS1.

Enrutamiento de la patata caliente

Ya estamos *finalmente* en posición de hablar acerca de los algoritmos de enrutamiento BGP de una forma precisa. Comenzaremos con uno de los algoritmos más simples, el denominado **enrutamiento de la patata caliente**.

Fíjese en el router 1b de la Figura 5.10. Como acabamos de describir, este router aprenderá que existen dos rutas BGP posibles hasta el prefijo x. En el enrutamiento de la patata caliente, la ruta elegida (de entre todas las posibles) será aquella que tenga el mínimo coste hasta el router NEXT-HOP que da comienzo a la ruta. En este ejemplo, el router 1b consultará su información de enrutamiento interna al sistema autónomo para encontrar la ruta interna de mínimo coste hasta el router NEXT-HOP 2a y la ruta interna de mínimo coste hasta el router NEXT-HOP 3d, y luego seleccionará la que tenga un menor coste de las dos. Por ejemplo, suponga que definimos el coste como el número de enlaces atravesados. Entonces el mínimo coste entre el router 1b y el router 2a será 2; el mínimo coste entre el router 1b y el router 3d será 3 y, por tanto, se seleccionaría el router 2a. A continuación, el router 1b consultaría su tabla de reenvío (configurada por su algoritmo de enrutamiento interno al sistema autónomo) y localizaría la interfaz I que se encuentra en la ruta de mínimo coste hacia el router 2a. Despues, añadirá (x, I) a su tabla de reenvío.

Los pasos para añadir a la tabla de reenvío de un router un prefijo externo al sistema autónomo (en el caso del enrutamiento de la patata caliente) se resumen en la Figura 5.11. Es importante observar que, a la hora de añadir a una tabla de reenvío un prefijo externo al sistema autónomo, se utilizan tanto el protocolo de enrutamiento entre sistemas autónomos (BGP) como el protocolo de enrutamiento interno al sistema autónomo (por ejemplo, OSPF).

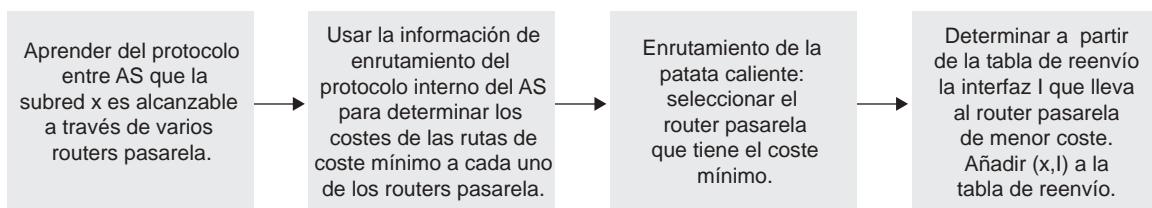


Figura 5.11 ♦ Pasos para añadir a la tabla de reenvío de un router un destino externo al sistema autónomo.

La idea que subyace al enrutamiento de la patata caliente es que el router 1b debe preocuparse de conseguir que los paquetes salgan de su propio sistema autónomo lo más rápidamente posible (o, para ser más específicos, con el menor coste posible), sin preocuparse del coste que tengan las partes restantes de la ruta hasta el destino, situadas fuera de su propio sistema autónomo. En el nombre “enrutamiento de la patata caliente” subyace la idea de que un paquete es análogo a una patata caliente, que nos está quemando las manos. Puesto que quema, queremos pasársela a otra persona (a otro sistema autónomo) lo más rápidamente posible. El enrutamiento de la patata caliente es, por tanto, un algoritmo egoísta: trata de reducir el coste dentro de su propio sistema autónomo, ignorando los restantes componentes de los costes extremo a extremo que están fuera de su sistema autónomo. Fíjese en que, con el enrutamiento de la patata caliente, dos routers pertenecientes a un mismo sistema autónomo podrían elegir dos rutas de sistemas autónomos diferentes para un mismo prefijo. Por ejemplo, acabamos de ver que el router 1b enviaría los paquetes a través de AS2 para alcanzar x. Sin embargo, el router 1d evitaría AS2 y enviaría los paquetes directamente a AS3 para alcanzar x.

Algoritmo de selección de ruta

En la práctica, BGP usa un algoritmo que es más complicado que el enrutamiento de la patata caliente, pero de todos modos incorpora también dicho enrutamiento. Para cualquier prefijo de destino especificado, la entrada al algoritmo de selección de ruta de BGP es el conjunto de todas las rutas hasta ese prefijo que han sido aprendidas y aceptadas por el router. Si solo existe una ruta, obviamente BGP seleccionará esa ruta. Si existen dos o más rutas hacia el mismo prefijo, entonces BGP invoca secuencialmente las siguientes reglas de eliminación hasta quedarse con una ruta:

1. Se asigna un valor de **preferencia local** a las rutas, como uno de sus atributos (además de los atributos AS-PATH y NEXT-HOP). La preferencia local de una ruta podría haber sido definida por el router o podría haber sido aprendida de otro router perteneciente al mismo sistema autónomo. El valor del atributo de preferencia local es una decisión política que se deja completamente en manos del administrador de red del sistema autónomo (en breve veremos en detalle las políticas de BGP). Se eligen las rutas con los valores de preferencia local más altos.
2. De las rutas que quedan (todas ellas con el mismo valor de preferencia local), se selecciona la ruta con la secuencia de sistemas autónomos (AS-PATH) más corta. Si esta regla fuera la única para seleccionar la ruta, entonces BGP estaría aplicando un algoritmo de vector de distancias para determinar la ruta, siendo la métrica de distancia utilizada el número de saltos entre sistemas autónomos, en lugar del número de saltos entre routers.
3. Para las restantes rutas (todas con el mismo valor de preferencia local y la misma longitud de AS-PATH), se utiliza el enrutamiento de la patata caliente, es decir, se selecciona la ruta con el router del siguiente salto (NEXT-HOP) más próximo.
4. Si siguiera quedando más de una ruta, el router utilizaría identificadores BGP para seleccionar la ruta; véase [Stewart 1999].

Como ejemplo, consideremos de nuevo el router 1b de la Figura 5.10. Recuerde que hay exactamente dos rutas BGP hacia el prefijo x, una que pasa a través de AS2 y otra que evita pasar a través de AS2. Recuerde también que si solo usáramos el enrutamiento de la patata caliente, entonces BGP enrutaría los paquetes hacia x a través de AS2. Pero en el anterior algoritmo de selección de ruta, la regla 2 se aplica antes de la regla 3, haciendo que BGP seleccione la ruta que evita pasar por AS2, ya que esa ruta tiene un AS-PATH más corto. Así que, como puede ver, con el anterior algoritmo de selección de ruta BGP ya no es un algoritmo egoísta: elige preferentemente las rutas con secuencias sistemas autónomos cortas (reduciendo así el retardo extremo a extremo).

Como hemos mencionado anteriormente, BGP es el estándar *de facto* para el enrutamiento entre sistemas autónomos de Internet. Para ver el contenido de varias tablas de enrutamiento BGP (¡muy grandes!) extraídas de routers de los ISP de nivel 1, consulte <http://www.routeviews.org>. Frecuentemente, las tablas de enrutamiento BGP contienen más de medio millón de rutas (es

decir, prefijos y sus atributos correspondientes). Puede ver estadísticas acerca del tamaño y las características de las tablas de enrutamiento BGP en [Potaroo 2016].

5.4.4 IP-Anycast

Además de ser el protocolo de enrutamiento entre sistemas autónomos de Internet, BGP se usa a menudo para implementar el servicio IP-anycast [RFC 1546, RFC 7094], utilizado comúnmente en DNS. Para entender la motivación de IP-anycast, piense que, en muchas aplicaciones, nos interesa (1) duplicar el mismo contenido en diferentes servidores, situados en muchas ubicaciones distintas, geográficamente dispersas; y (2) hacer que cada usuario acceda al contenido usando el servidor que tenga más próximo. Por ejemplo, una red CDN podría replicar los videos y otros objetos en servidores situados en distintos países. De forma similar, el sistema DNS puede replicar los registros DNS en servidores DNS dispersos por todo el mundo. Cuando un usuario quiera acceder a este contenido replicado, resulta conveniente dirigir al usuario al servidor “más próximo” que disponga de ese contenido. El algoritmo de selección de rutas de BGP proporciona un mecanismo sencillo y natural para hacer esto.

Para concretar nuestra exposición, veamos de qué modo podría una red CDN utilizar IP-anycast. Como se muestra en la Figura 5.12, durante la etapa de configuración de IP-anycast la empresa CDN asigna la *misma* dirección IP a cada uno de sus servidores, y utiliza BGP estándar para anunciar esa dirección IP de cada uno de los servidores. Cuando un router BGP recibe múltiples anuncios de ruta para esta dirección IP, trata esos anuncios como si proporcionaran diferentes rutas hacia una misma ubicación física (cuando, de hecho, los anuncios son para diferentes rutas hacia distintas ubicaciones físicas). Al configurar su tabla de enrutamiento, cada router utilizará localmente el algoritmo de selección de rutas de BGP para elegir la “mejor” ruta hacia esa dirección IP (por ejemplo, la ruta más cercana, midiendo según el número de saltos de

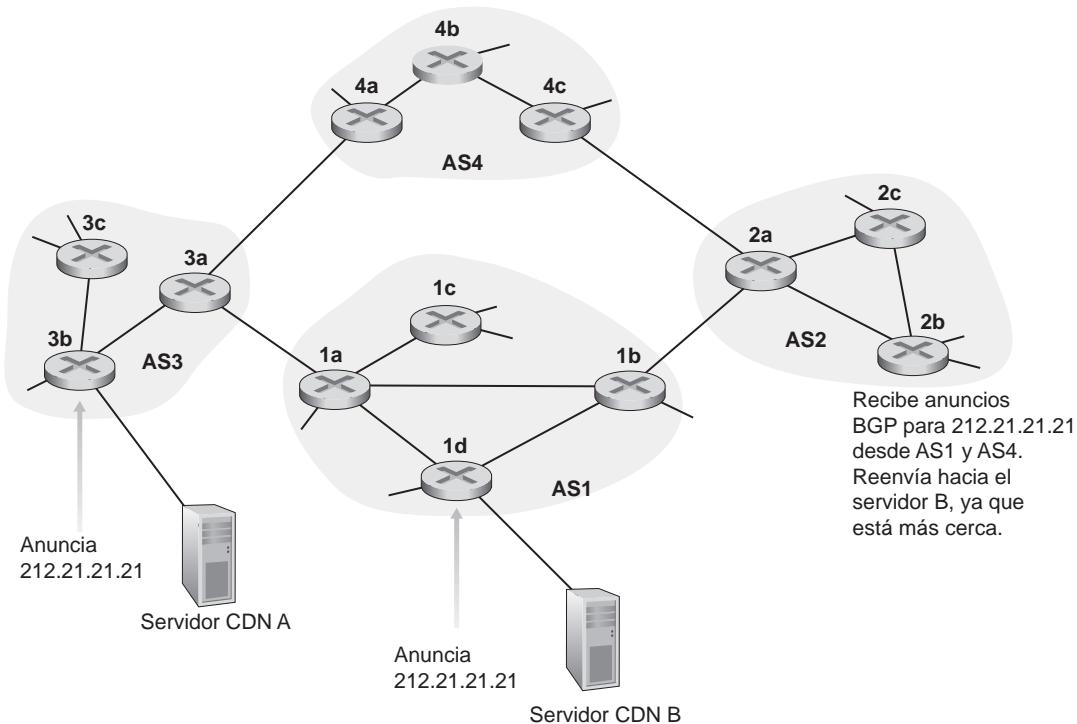


Figura 5.12 ♦ Utilización de IP-anycast para dirigir a los usuarios al servidor CDN más próximo.

sistema autónomo). Por ejemplo, si una ruta BGP (correspondiente a una ubicación) está a una distancia de un solo salto de sistema autónomo con respecto al router y todas las restantes rutas BGP (correspondientes a otras ubicaciones) están a una distancia de dos o más saltos de sistema autónomo, entonces el router BGP decidirá enrutar los paquetes hacia la ubicación que está a un solo salto de distancia. Después de esta fase inicial BGP de anuncio de direcciones, la red CDN puede dedicarse a su tarea principal: distribuir contenido. Cuando un cliente solicita el vídeo, la CDN devuelve al cliente la dirección IP común utilizada por los servidores geográficamente dispersos, con independencia de dónde esté situado el cliente. Cuando el cliente envía una solicitud a esa dirección IP, los routers de Internet reenvían ese paquete de solicitud al servidor “más próximo”, según determine el algoritmo de selección de rutas de BGP.

Aunque el ejemplo anterior de CDN ilustra adecuadamente cómo puede usarse IP-anycast, en la práctica las redes CDN no suelen utilizar IP-anycast, porque los cambios en el enrutamiento BGP pueden hacer que diferentes paquetes de una misma conexión TCP lleguen a diferentes instancias del servidor web. Pero IP-anycast sí que es ampliamente utilizado por el sistema DNS para dirigir las consultas DNS al servidor DNS raíz más cercano. Recuerde, de la Sección 2.4, que actualmente existen 13 direcciones IP para los servidores DNS raíz. Pero para cada una de estas direcciones hay múltiples servidores DNS raíz, teniendo algunas de estas direcciones más de 100 servidores DNS raíz dispersos por todo el mundo. Cuando se envía una consulta DNS a una de estas 13 direcciones IP, se utiliza IP-anycast para enrutar la consulta hacia el servidor DNS raíz más cercano que sea responsable de esa dirección.

5.4.5 Política de enrutamiento

Cuando un router selecciona una ruta hacia un destino, la política de enrutamiento del sistema autónomo puede prevalecer sobre todas las restantes consideraciones, como la secuencia más corta de sistemas autónomos o el enrutamiento de la patata caliente. De hecho, en el algoritmo de selección de ruta, las rutas se seleccionan primero de acuerdo con el atributo de preferencia local, cuyo valor está fijado por la política del sistema autónomo local.

Vamos a ilustrar algunos de los conceptos básicos de la política de enrutamiento de BGP con un ejemplo sencillo. La Figura 5.13 muestra seis sistemas autónomos interconectados: A, B, C, W, X e Y. Es importante observar que A, B, C, W, X e Y son sistemas autónomos, no routers. Supongamos que los sistemas autónomos W, X e Y son ISP de acceso, y que A, B y C son redes troncales de empresas operadoras. Supongamos también que A, B y C se intercambian tráfico directamente y que proporcionan información BGP completa a sus redes clientes. Todo el tráfico que entra en la red de un ISP de acceso tiene que estar destinado a esa red, y todo el tráfico que sale de la red de un ISP de acceso tiene que haber sido originado en esa red. W e Y son claramente proveedores ISP de acceso. X es un **ISP de acceso multidomiciliado**, ya que está conectado al resto de la red a través de dos proveedores diferentes (un escenario cada vez más común en la práctica). Sin embargo, al igual que sucede con W e Y, X tiene que ser el origen/destino de todo el tráfico que sale/entra en X. Pero, ¿cómo puede implementarse este comportamiento de la red de acceso? ¿Cómo impedir que X reenvíe tráfico entre B y C? Esto se puede conseguir fácilmente controlando la forma en que

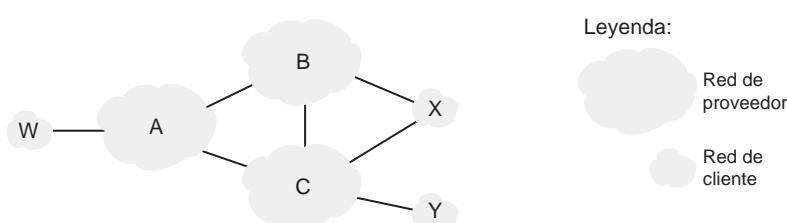


Figura 5.13 ♦ Un escenario simple de política BGP.

EN LA PRÁCTICA

¿POR QUÉ LOS SISTEMAS AUTÓNOMOS DISPONEN DE DIFERENTES PROTOCOLOS DE ENRUTAMIENTO PARA USO INTERNO Y PARA COMUNICARSE ENTRE ELLOS?

Una vez estudiados los detalles de una serie de protocolos específicos de enrutamiento interno a los sistemas autónomos y de enrutamiento entre sistemas autónomos, implantados actualmente en Internet, vamos a concluir considerando quizá la cuestión más importante que podríamos plantearnos acerca de estos protocolos (¡esperamos que le haya surgido esta duda a lo largo del capítulo y que los árboles no le hayan impedido ver el bosque!): ¿por qué se utilizan protocolos de enrutamiento distintos dentro de los sistemas autónomos y para comunicarse entre unos sistemas autónomos y otros?

La respuesta a esta pregunta se encuentra en las diferencias de objetivos entre el enrutamiento dentro de un sistema autónomo y el enrutamiento entre sistemas autónomos:

- *Política.* Entre sistemas autónomos, prevalecen las políticas. Puede ser importante que el tráfico originado en un determinado sistema autónomo no pueda atravesar otro sistema autónomo específico. De forma similar, un sistema autónomo dado puede desear controlar el tráfico de tránsito que transporta entre otros sistemas autónomos. Hemos visto que BGP transporta atributos de ruta y permite la distribución controlada de información de enrutamiento, de manera que pueden tomarse ese tipo decisiones de enrutamiento basadas en políticas. Dentro de un sistema autónomo, todo está en teoría bajo el mismo control administrativo y, por tanto, las políticas desempeñan un papel mucho menos importante en la elección de rutas dentro del sistema autónomo.
- *Escala.* La capacidad de ampliación de un algoritmo de enrutamiento y de sus estructuras de datos, con el fin de gestionar el enrutamiento hacia/entre una gran cantidad de redes, es un problema crítico en el enrutamiento entre sistemas autónomos. Dentro de un sistema autónomo, la escalabilidad es un problema menor, aunque sólo sea porque si un ISP se hace demasiado grande, siempre es posible dividirlo en dos sistemas autónomos y realizar el enrutamiento entre los dos nuevos sistemas. (Recuerde que OSPF permite crear una jerarquía de ese estilo, dividiendo un sistema autónomo en áreas.)
- *Rendimiento.* Puesto que el enrutamiento entre sistemas autónomos está muy orientado a las políticas, la calidad (por ejemplo, el rendimiento) de las rutas utilizadas suele ser una cuestión secundaria (es decir, una ruta más larga o más costosa que satisfaga determinados criterios políticos puede perfectamente tener preferencia sobre una ruta más corta, pero que no cumpla dichos criterios). De hecho, hemos visto que entre sistemas autónomos no existe ni siquiera el concepto de coste asociado con las rutas (salvo por el recuento de saltos entre sistemas autónomos). Sin embargo, dentro de un sistema autónomo tales cuestiones políticas tienen una importancia menor, lo que permite al enrutamiento centrarse más en el rendimiento que se puede alcanzar en una ruta.

son anunciadas las rutas BGP. En concreto, X operará como la red de un ISP de acceso si anuncia (a sus vecinos B y C) que no tiene ninguna ruta a ningún otro destino excepto a ella misma. Es decir, incluso aunque X pueda conocer una determinada ruta, por ejemplo XCY, para llegar a la red Y, no anunciará este camino a B. Puesto que B no es consciente de que X dispone de un camino hasta Y, B nunca reenviará tráfico destinado a Y (o a C) a través de X. Este sencillo ejemplo ilustra cómo se puede utilizar una política selectiva de anuncio de rutas para implementar las relaciones de enrutamiento cliente/proveedor.

A continuación, vamos a centrarnos en la red de un proveedor, por ejemplo, en el sistema autónomo B. Suponga que B ha aprendido (de A) que A dispone de la ruta AW hacia W. B puede, por tanto, incluir la ruta AW en su base de información de enrutamiento. Evidentemente, B también quiere anunciar la ruta BAW a su cliente, X, para que X sepa que puede llegar a W a través de B. Pero, ¿debería B anunciar la ruta BAW a C? Si lo hace, entonces C podría enviar tráfico a W por la ruta BAW. Si A, B y C son proveedores troncales, entonces B podría pensar, con razón, que no

tendría que asumir la carga (¡y el coste!) de transportar el tráfico en tránsito entre A y C. B podría pensar, con razón, que es el trabajo (¡y el coste!) de A y C asegurarse de que C puede enrutar hacia/ desde los clientes de A a través de una conexión directa entre A y C. Actualmente, no existe ningún estándar oficial que gobierne cómo los ISP troncales deben llevar a cabo el enrutamiento entre ellos. Sin embargo, una regla heurística seguida por los ISP comerciales es que cualquier tráfico que fluya a través de la red troncal de un ISP tiene que tener su origen o su destino (o ambos) en una red que sea un cliente de dicho ISP; si fuera de otro modo, ese tráfico estaría obteniendo un viaje gratis a través de la red del ISP. Los acuerdos bilaterales individuales (que regulan cuestiones como las mencionadas más arriba) normalmente son negociados entre parejas de proveedores ISP y suelen ser confidenciales; [Huston 1999a] proporciona una interesante explicación acerca de esos acuerdos bilaterales. Si desea ver una descripción detallada de cómo las políticas de enrutamiento reflejan las relaciones comerciales entre los ISP, consulte [Gao 2001; Dmitriopoulos 2007]. Para ver una descripción de las políticas de enrutamiento BGP desde el punto de vista de un ISP, consulte [Caesar 2005b].

Con esto completamos nuestra breve introducción a BGP. Es importante comprender BGP porque juega un papel crucial en Internet. Le animamos a consultar las referencias [Griffin 2012; Stewart 1999; Labovitz 1997; Halabi 2000; Huitema 1998; Gao 2001; Feamster 2004; Caesar 2005b; Li 2007] para aprender más acerca de BGP.

5.4.6 Cómo encajan las piezas: obtención de presencia en Internet

Aunque esta subsección no trata acerca de BGP *per se*, engarza muchos de los protocolos y conceptos que hemos visto hasta ahora, incluyendo el direccionamiento IP, DNS y BGP.

Suponga que acaba de crear una pequeña empresa que dispone de una serie de servidores, incluyendo un servidor web público que describe los productos y servicios de su empresa, un servidor de correo desde el que sus empleados obtienen sus mensajes de correo electrónico y un servidor DNS. Naturalmente, lo que querrá es que todo el mundo pueda visitar su sitio web, para conocer sus excitantes productos y servicios. Además, querrá que sus empleados sean capaces de enviar y recibir correos electrónicos de clientes potenciales repartidos por todo el mundo.

Para conseguir estos objetivos, primero necesitará obtener conectividad Internet, lo cual se consigue firmando un contrato con un ISP local y conectándose a él. Su empresa dispondrá de un router de pasarela, que estará conectado a un router de su ISP local. Esta conexión podría tratarse de una conexión DSL a través de la infraestructura telefónica existente, de una línea arrendada que conecte con el router del ISP o de cualquiera de las otras muchas soluciones de acceso descritas en el Capítulo 1. Su ISP local también le proporcionará un rango de direcciones IP, como por ejemplo un rango de direcciones /24 compuesto por 256 direcciones. Una vez que disponga de la conectividad física y del rango de direcciones IP, tendrá que asignar una de las direcciones IP de su rango a su servidor web, otra a su servidor de correo, otra a su servidor DNS, otra a su router de pasarela y otras direcciones IP a otros servidores y dispositivos de la red de su empresa.

Además de firmar un contrato con su ISP, también tendrá que pagar a una entidad de registro de Internet para obtener un nombre de dominio para su empresa, como se describe en el Capítulo 2. Por ejemplo, si el nombre de su empresa es Xanadu Inc., lógicamente intentaría adquirir el nombre de dominio xanadu.com. Su empresa deberá obtener también presencia en el sistema DNS. Específicamente, como los usuarios externos querrán contactar con su servidor DNS para obtener las direcciones IP de sus servidores, también tendrá que proporcionar a su entidad de registro la dirección IP de su servidor DNS. Su entidad de registro añadirá entonces una entrada para su servidor DNS (nombre de dominio y dirección IP correspondiente) en los servidores del dominio de nivel superior .com, como se describe en el Capítulo 2. Una vez completado este paso, cualquier usuario que conozca su nombre de dominio (por ejemplo, xanadu.com), será capaz de obtener la dirección IP de su servidor DNS a través del sistema DNS.

Para que la gente pueda descubrir las direcciones IP de su servidor web, tendrá que incluir entradas en su servidor DNS que establezcan la correspondencia entre el nombre de host de su servidor web (por ejemplo, xanadu.com) y su dirección IP. También deberá tener entradas similares para otros servidores de su empresa que estén públicamente disponibles, incluyendo su servidor de correo. Con todo esto, si Alicia quiere acceder a su servidor web, el sistema DNS contactará con su servidor DNS, averiguará la dirección IP de su servidor web y se la proporcionará a Alicia. Alicia podrá entonces establecer una conexión TCP directamente con su servidor web.

Sin embargo, todavía queda otro paso necesario y crucial para permitir que usuarios de todo el mundo accedan a su servidor web. Piense en lo que sucedería cuando Alicia, que conoce la dirección IP de su servidor web, envíe un datagrama IP (por ejemplo, un segmento SYN TCP) a esa dirección IP. Ese datagrama será enrutado a través de Internet, visitando una serie de routers en muchos sistemas autónomos distintos, y terminará por llegar a su servidor web. Cuando cualquiera de los routers reciba el datagrama, buscará en su tabla de reenvío una entrada que le permita determinar a través de qué puerto de salida debe reenviar el datagrama. Por tanto, cada uno de los routers necesita conocer la existencia del prefijo /24 de su empresa (o de alguna entrada agregada que lo incluya). ¿Cómo puede un router conocer el prefijo de su empresa? ¡Como acabamos de ver, lo consigue gracias a BGP! Específicamente, cuando su empresa firma un contrato con un ISP local y se le asigna un prefijo (es decir, un rango de direcciones), su ISP local usará BGP para anunciar su prefijo a los ISP con los que esté conectado. Esos ISP usarán BGP, a su vez, para propagar el anuncio. Al final, todos los routers de Internet conocerán su prefijo (o algún agregado que lo incluya) y serán así capaces de reenviar apropiadamente los datagramas destinados a su servidor web y a su servidor de correo.

5.5 El plano de control SDN

En esta sección, vamos a sumergirnos en el plano de control SDN: la lógica de la red que controla el reenvío de paquetes entre los dispositivos compatibles con SDN de una red, así como la configuración y gestión de estos dispositivos y sus servicios. Nuestro estudio aquí parte del análisis del reenvío SDN generalizado que hemos esbozado anteriormente en la Sección 4.4, así que quizás convenga que revise esa sección, al igual que la Sección 5.1 de este capítulo, antes de continuar. Como en la Sección 4.4, adoptaremos aquí la terminología usada en la literatura sobre SDN y denominaremos a los dispositivos de reenvío de la red “comutadores de paquetes” (o simplemente comutadores, sobreentendiéndose lo de “paquetes”), ya que las decisiones de reenvío pueden tomarse basándose en las direcciones de origen/destino de la capa de red, en las direcciones de origen/destino de la capa de enlace o en muchos otros valores de los campos de las cabeceras de transporte, de red y de enlace del paquete.

Podemos identificar cuatro características fundamentales de una arquitectura SDN [Kreutz 2015]:

- *Reenvío basado en el flujo.* El reenvío de paquetes por parte de comutadores controlados mediante SDN puede estar basado en los valores de diversos campos de las cabeceras de las capas de transporte, de red o de enlace. Ya vimos en la Sección 4.4 que la abstracción OpenFlow 1.0 permite el reenvío basado en once valores diferentes de campos de cabecera. Esto contrasta fuertemente con el enfoque tradicional de reenvío utilizado por los routers y que estudiamos en las secciones 5.2-5.4, en el que el reenvío de datagramas IP se basaba exclusivamente en la dirección IP de destino del datagrama. Recuerde de la Figura 5.2 que las reglas de reenvío de paquetes están especificadas en la tabla de flujo de un comutador; es responsabilidad del plano de control SDN calcular, gestionar e instalar las entradas de las tablas de flujo en todos los comutadores de la red.

- *Separación del plano de datos y el plano de control.* Esta separación se muestra claramente en las Figuras 5.2 y 5.14. El plano de datos está compuesto por los conmutadores de la red: dispositivos relativamente simples (pero rápidos) que ejecutan las reglas de “correspondencia más acción” contenidas en sus tablas de flujo. El plano de control está compuesto por servidores y software que determinan y gestionan las tablas de flujo de los conmutadores.
- *Funciones de control de la red: externas a los conmutadores del plano de datos.* Dado que la “S” de SDN significa “software”, quizás no resulte sorprendente que el plano de control SDN esté implementado en software. A diferencia de los routers tradicionales, sin embargo, este software se ejecuta en servidores que son diferentes (y remotos) de los conmutadores de la red. Como se muestra en la Figura 5.14, el propio plano de control consta de dos componentes: un controlador SDN (o sistema operativo de red [Gude 2008]) y un conjunto de aplicaciones de control de red. El controlador mantiene información precisa sobre el estado de la red (por ejemplo, el estado de los hosts, conmutadores y enlaces remotos), proporciona esta información a las aplicaciones de control de red que se ejecutan en el plano de control y proporciona los medios con los cuales estas aplicaciones pueden monitorizar, programar y controlar los dispositivos de la red subyacente. Aunque el controlador de la Figura 5.14 se muestra como un único servidor central, en la práctica el controlador solo está centralizado desde el punto de vista lógico; normalmente se implementa mediante varios servidores, que proporcionan unas prestaciones coordinadas y escalables, así como una alta disponibilidad.
- *Una red programable.* La red se puede programar mediante las aplicaciones de control de red que se ejecutan en el plano de control. Estas aplicaciones representan el “cerebro” del plano de control SDN y utilizan las API proporcionadas por el controlador SDN con el fin de especificar y controlar el plano de datos en los dispositivos de la red. Por ejemplo, una aplicación de control de red para enrutamiento podría determinar las rutas extremo a extremo entre orígenes y destinos (por ejemplo, ejecutando el algoritmo de Dijkstra a partir de la información de estado

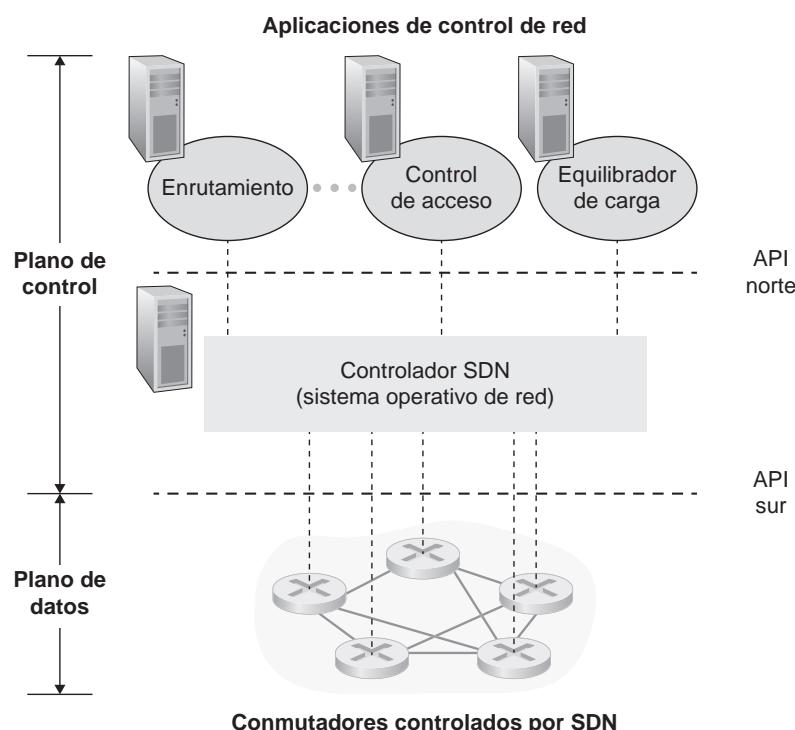


Figura 5.14 ♦ Componentes de la arquitectura SDN: conmutadores controlados por SDN, el controlador SDN y las aplicaciones de control de red.

de los nodos y de los enlaces mantenida por el controlador SDN). Otra aplicación de red podría encargarse del control de acceso, es decir, de determinar qué paquetes hay que bloquear en un conmutador, como en nuestro tercer ejemplo de la Sección 4.4.3. Otra aplicación diferente podría reenviar los paquetes de forma que se consiga un equilibrio de carga entre servidores (el segundo ejemplo que hemos considerado en la Sección 4.4.3).

Teniendo en cuenta todo esto, podemos ver que SDN representa un significativo “desempaquetamiento” de la funcionalidad de red: los conmutadores del plano de datos, los controladores SDN y las aplicaciones de control de red son entidades separadas, que pueden ser suministradas por diferentes proveedores y organizaciones. Esto contrasta con el modelo pre-SDN, en el que un router/switch (junto con su software integrado del plano de control y sus implementaciones de protocolos) era monolítico, estaba integrado verticalmente y era suministrado por un único fabricante. Este desempaquetamiento de la funcionalidad de red en SDN ha sido comparado con la evolución, en su día, desde las computadoras mainframe (en las que el hardware, el software del sistema y las aplicaciones eran suministrados por un único fabricante) a las computadoras personales (con su hardware, sistemas operativos y aplicaciones separados). El desempaquetamiento del hardware de la computadora, del software del sistema y de las aplicaciones ha dado como resultado un ecosistema abierto y rico, guiado por la innovación en esas tres áreas; la esperanza para SDN es que también traiga consigo una rica y similar ola de innovación.

Dada nuestra comprensión de la arquitectura SDN de la Figura 5.14, surgen de manera natural numerosas cuestiones. ¿Cómo y dónde se calculan en la práctica las tablas de flujo? ¿Cómo se actualizan en los dispositivos controlados por SDN esas tablas, en respuesta a sucesos (por ejemplo, un enlace conectado que se active/desactive)? ¿Y cómo se coordinan las entradas de las tablas de flujo de múltiples conmutadores, de tal manera que resulte una funcionalidad de la red orquestada y coherente (por ejemplo, rutas extremo a extremo de reenvío de paquetes desde los orígenes hasta los destinos, o cortafuegos distribuidos coordinados)? Es tarea del plano de control SDN proporcionar estas, y otras muchas, capacidades.

5.5.1 El plano de control SDN: controlador SDN y aplicaciones SDN de control de red

Comencemos nuestra exposición sobre el plano de control SDN de una forma abstracta, analizando las capacidades genéricas que el plano de control debe proporcionar. Como veremos, este enfoque abstracto, sobre los “principios fundamentales”, nos llevará a una arquitectura global que refleja el modo en que se han implementado en la práctica los planos de control.

Como hemos dicho antes, el plano de control SDN está dividido, a grandes rasgos, en dos componentes: el controlador SDN y las aplicaciones SDN de control de red. Exploraremos en primer lugar el controlador. Se han desarrollado muchos controladores SDN desde que surgiera el primero de ellos [Gude 2008]; en [Kreutz 2015] podrá encontrar un resumen exhaustivo y actualizado. La Figura 5.15 proporciona una vista más detallada de un controlador SDN genérico. La funcionalidad de un controlador puede organizarse, desde un punto de vista general, en tres capas. Veamos cuáles son estas capas en el orden tradicional abajo-arriba:

- *Una capa de comunicaciones: comunicación entre el controlador SDN y los dispositivos de red controlados.* Obviamente, si un controlador SDN tiene que controlar el funcionamiento de un conmutador, host u otro dispositivo remoto compatible con SDN, se necesita un protocolo para transferir información entre el controlador y ese dispositivo. Además, un dispositivo debe ser capaz de comunicar al controlador sucesos observados localmente (por ejemplo, un mensaje que indique que un enlace conectado se ha activado o desactivado o que un dispositivo se acaba de unir a la red, o un latido que indique que un dispositivo está activo y funcionando). Estos sucesos proporcionan al controlador SDN una visión actualizada del estado de la red. Este protocolo

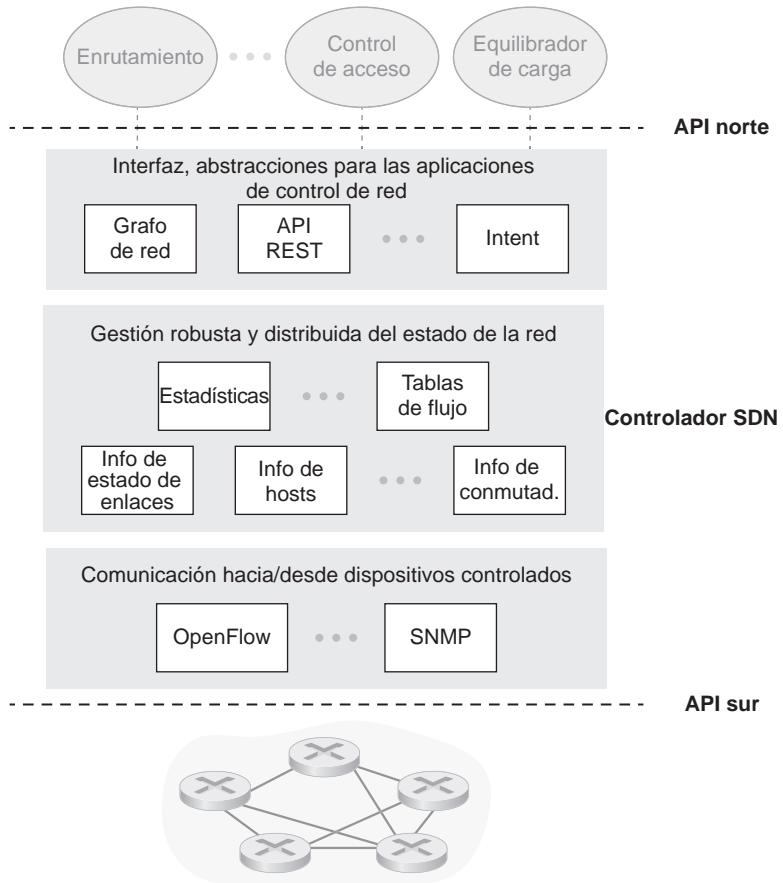


Figura 5.15 ♦ Componentes de un controlador SDN.

constituye la capa inferior de la arquitectura del controlador, como se muestra en la Figura 5.15. La comunicación entre el controlador y los dispositivos controlados cruza lo que se ha dado en llamar la interfaz “sur” del controlador (*southbound interface*). En la Sección 5.5.2 estudiaremos OpenFlow, un protocolo específico que proporciona esta funcionalidad de comunicaciones. OpenFlow está implementado en la mayoría de los controladores SDN, si no en todos.

- *Una capa de gestión del estado de la red.* Las decisiones últimas de control tomadas por el plano de control SDN —por ejemplo, configurar tablas de flujo en todos los commutadores para conseguir el reenvío extremo a extremo deseado, para implementar un equilibrado de carga o para habilitar una determinada capacidad de cortafuegos— requiere que el controlador disponga de información actualizada sobre el estado de los hosts, enlaces, commutadores y otros dispositivos controlados por SDN que haya en la red. La tabla de flujo de un commutador contiene contadores cuyos valores también pueden ser aprovechados por las aplicaciones de control de red; por tanto, estos valores deberán estar disponibles para esas aplicaciones. Puesto que el fin último del plano de control es determinar las tablas de flujo de los diversos dispositivos controlados, un controlador puede también mantener una copia de dichas tablas. Todos estos elementos de información constituyen ejemplos del “estado” de la red que el controlador SDN mantiene.
- *La interfaz con la capa de aplicaciones de control de red.* El controlador interactúa con las aplicaciones de control de red a través de su interfaz “norte” (*northbound interface*). Esta API permite a las aplicaciones de control de red leer/escribir el estado de la red y las tablas de flujo

dentro de la capa de gestión del estado. Las aplicaciones pueden registrarse para que se las notifique cuando se produzcan sucesos de cambio de estado, con el fin de poder tomar acciones en respuesta a las notificaciones de sucesos de red enviadas desde los dispositivos controlados por SDN. Pueden proporcionarse diferentes tipos de APIs; como veremos más adelante, dos controladores SDN muy populares se comunican con sus aplicaciones utilizando una interfaz solicitud-respuesta tipo REST [Fielding 2000].

Ya hemos indicado, en diversas ocasiones, que se puede considerar que un controlador SDN está “lógicamente centralizado”, es decir, que el controlador puede ser visto externamente (por ejemplo, desde el punto de vista de los dispositivos controlados por SDN y de las aplicaciones externas de control de red) como un único servicio monolítico. Sin embargo, estos servicios, y las bases de datos utilizadas para almacenar la información de estado, se implementan en la práctica mediante un conjunto *distribuido* de servidores, por razones de tolerancia ante fallos, de alta disponibilidad o de prestaciones. Al estar las funciones controladoras implementadas mediante un conjunto de servidores, hay que tomar en consideración la semántica de las operaciones internas del controlador (por ejemplo, el mantenimiento de un orden temporal lógico de los sucesos, la coherencia, el consenso y otros factores) [Panda 2013]. Esas preocupaciones son comunes en muchos sistemas distribuidos; en [Lamport 1989, Lamson 1996] puede encontrar algunas soluciones elegantes a estos desafíos. Los controladores modernos como OpenDaylight [OpenDaylight Lithium 2016] y ONOS [ONOS 2016] (véase el recuadro adjunto de En la práctica) han puesto un considerable énfasis en desarrollar la arquitectura de una plataforma controladora lógicamente centralizada, pero físicamente distribuida, que proporcione servicios escalables y alta disponibilidad, tanto a los dispositivos controlados como a las aplicaciones de control de red.

La arquitectura mostrada en la Figura 5.15 se asemeja mucho a la del controlador NOX originalmente propuesto en 2008 [Gude 2008], así como a la de los actuales controladores SDN OpenDaylight [OpenDaylight Lithium 2016] y ONOS [ONOS 2016] (véase el recuadro adjunto de En la práctica). Veremos un ejemplo de operación del controlador en la Sección 5.5.3. Pero primero vamos a examinar el protocolo OpenFlow, que reside en la capa de comunicaciones del controlador.

5.5.2 Protocolo OpenFlow

El protocolo OpenFlow [OpenFlow 2009, ONF 2016] opera entre un controlador SDN y un conmutador controlado por SDN u otro dispositivo que implemente la API OpenFlow que hemos estudiado anteriormente, en la Sección 4.4. El protocolo OpenFlow funciona sobre TCP, siendo su número de puerto predeterminado el 6653.

Entre los mensajes de mayor importancia que fluyen desde el controlador hacia el conmutador controlado podemos citar:

- *Configuración (configuration)*. Este mensaje permite al controlador consultar y configurar los parámetros de configuración del conmutador.
- *Modificar estado (modify-state)*. El controlador utiliza este mensaje para añadir/borrar o modificar entradas de la tabla de flujo del conmutador, así como para configurar las propiedades de los puertos del conmutador.
- *Leer estado (read-state)*. El controlador usa este mensaje para recopilar estadísticas y valores de contadores de los puertos y tablas de flujo del conmutador.
- *Enviar paquete (send-packet)*. El controlador usa este mensaje para enviar un paquete específico a través de un puerto de salida especificado del conmutador controlado. El propio mensaje contiene, en su carga útil, el paquete que hay que enviar.

Entre los mensajes que fluyen desde el conmutador controlado por SDN hacia el controlador podemos citar:



EN LA PRÁCTICA

RED GLOBAL DEFINIDA POR SOFTWARE DE GOOGLE

Recuerde del caso de estudio de la Sección 2.6 que Google tiene implantada una red de área extensa (WAN) dedicada, que interconecta sus centros de datos y sus clusters de servidores (en IXPs e ISPs). Esta red, denominada B4, tiene un plano de control SDN diseñado por Google y construido sobre OpenFlow. La red de Google es capaz de conseguir una utilización promedio de sus enlaces WAN del 70% (entre dos y tres veces superior a la tasa de utilización típica de los enlaces) y de dividir los flujos de aplicación entre múltiples rutas, basándose en la prioridad de las aplicaciones y en las demandas de flujo existentes [Jain 2013].

La red B4 de Google está particularmente bien adaptada a SDN: (i) Google controla todos los dispositivos, desde los servidores de frontera situados en IXPs e ISPs, hasta los routers que forman el núcleo de su red; (ii) las aplicaciones que más ancho de banda requieren son copias de datos a gran escala entre sitios, que pueden ceder la preferencia a las aplicaciones interactivas de mayor prioridad durante los períodos de congestión de recursos; (iii) al haber solo unas pocas docenas de centros de datos conectados, el control centralizado resulta factible.

La red B4 de Google utiliza comutadores hechos a medida, cada uno de los cuales implementa una versión ligeramente ampliada de OpenFlow, con un agente local OpenFlow (OFA, *OpenFlow Agent*) que es similar en espíritu al agente de control que nos encontramos en la Figura 5.2. Cada OFA se conecta, a su vez, con un controlador OpenFlow (OFC, *OpenFlow Controller*) que reside en el servidor de control de la red (NCS, *Network Control Server*) a través de una red separada “fuera de banda”, distinta de la red que transporta el tráfico entre centros de datos. El OFC proporciona, por tanto, los servicios utilizados por el NCS para comunicarse con los comutadores que controla, de forma similar en espíritu a la capa inferior de la arquitectura SDN mostrada en la Figura 5.15. En B4, el OFC también realiza funciones de gestión de estado, manteniendo el estado de los nodos y enlaces en una base de información de red (NIB, *Network Information Base*). La implementación del OFC realizada por Google está basada en el controlador SDN ONIX [Koponen 2010]. Están implementados dos protocolos de enruteamiento: BGP para el enruteamiento entre centros de datos e IS-IS (un pariente cercano de OSPF) para el enruteamiento en el interior de un centro de datos. Se utiliza Paxos [Chandra 2007] para ejecutar réplicas en caliente de los componentes del NCS, con el fin de protegerse frente a posibles fallos.

Una aplicación de control de red especializada en ingeniería de tráfico (que descansa desde un punto de vista lógico sobre el conjunto de servidores de red) interactúa con estos servidores para proporcionar a los grupos de flujos de aplicación asignaciones globales de ancho de banda, que afectan a toda la red. Gracias a B4, SDN dio un importante salto adelante, adentrándose en las redes operacionales de un proveedor de red global. En [Jain 2013] puede encontrar una descripción detallada de B4.

- *Flujo eliminado (flow-removed)*. Este mensaje informa al controlador de que se ha eliminado una entrada de una tabla de flujo, por ejemplo debido a un fin de temporización o como resultado de un mensaje *modify-state* recibido.
- *Estado de puerto (port-status)*. El comutador usa este mensaje para informar al controlador de un cambio en el estado de un puerto.
- *Paquete entrante (packet-in)*. Recuerde de la Sección 4.4 que los paquetes que llegan a un puerto del comutador y que no se corresponden con ninguna entrada de la tabla de flujo se envían al controlador para su procesamiento adicional. Los paquetes para los que sí existe correspondencia también pueden ser enviados al controlador, como una de las posibles acciones que se ejecutan al detectarse la correspondencia. El mensaje *packet-in* se emplea para enviar tales paquetes al controlador.

En [OpenFlow 2009, ONF 2016] se definen mensajes OpenFlow adicionales.

5.5.3 Interacción entre los planos de datos y de control: ejemplo

Para cimentar nuestra comprensión de la interacción entre los dispositivos controlados por SDN y el controlador SDN, consideremos el ejemplo mostrado en la Figura 5.16, en el que se utiliza el algoritmo de Dijkstra (que hemos estudiado en la Sección 5.2) para determinar las rutas más cortas. El escenario SDN de la Figura 5.16 tiene dos diferencias importantes con respecto al escenario anterior de control por router de las secciones 5.2.1 y 5.3, en donde el algoritmo de Dijkstra estaba implementado en todos y cada uno de los routers y se usaba la técnica de inundación para transmitir a todos los routers de la red las actualizaciones del estado de los enlaces:

- El algoritmo de Dijkstra se ejecuta como una aplicación separada, fuera de los commutadores de paquetes.
- Los commutadores de paquetes envían las actualizaciones de estado de los enlaces al controlador SDN, en vez de intercambiárselas unos con otros.

En este ejemplo, supongamos que el enlace entre los commutadores s1 y s2 se desactiva; supongamos también que está implementado un enrutamiento basado en la ruta más corta, por lo que las reglas de reenvío para los flujos entrantes y salientes en s1, s2 y s4 se verán afectadas; y supongamos por último que la operación de s3 no se ve afectada. Asumamos también que se utiliza OpenFlow como protocolo de la capa de comunicaciones y que la única función que realiza el plano de control es el enrutamiento basado en el estado de los enlaces.

1. El commutador s1, al experimentar un fallo en el enlace entre él mismo y s2, notifica el cambio en el estado del enlace al controlador SDN, utilizando el mensaje *port-status* de OpenFlow.

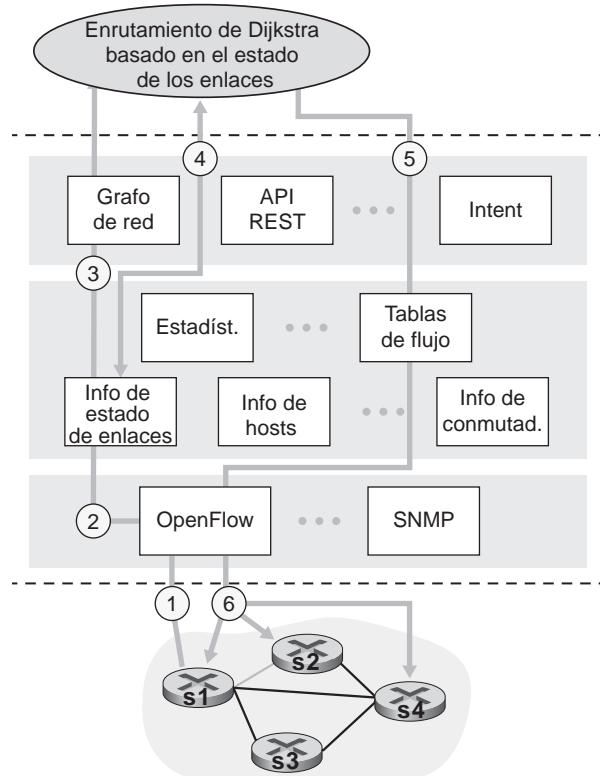


Figura 5.16 ♦ Escenario con controlador SDN: cambio en el estado de un enlace.

2. El controlador SDN recibe el mensaje OpenFlow que indica el cambio en el estado del enlace y envía una notificación al gestor del estado de los enlaces, que actualiza una base de datos de estado de los enlaces.
3. La aplicación de control de red que implementa el enrutamiento de Dijkstra basado en el estado de los enlaces, se había registrado previamente para ser notificada cuando hubiera cambios en el estado de los enlaces. Esta aplicación recibe la notificación del cambio en el estado del enlace.
4. La aplicación de enrutamiento basada en el estado de los enlaces interactúa con el gestor de estado de los enlaces para obtener información actualizada del estado de los enlaces; también puede consultar otros componentes de la capa de gestión del estado. Después calcula las nuevas rutas de coste mínimo.
5. La aplicación de enrutamiento basada en el estado de los enlaces interactúa entonces con el gestor de tablas de flujo, que determina las tablas de flujo que hay que actualizar.
6. El gestor de tablas de flujo utiliza entonces el protocolo OpenFlow para actualizar las entradas de las tablas de flujo de los comutadores afectados: s1 (que ahora enrutará a través de s4 los paquetes destinados a s2), s2 (que ahora comenzará a recibir paquetes de s1 a través del comutador intermedio s4) y s4 (que ahora debe reenviar los paquetes de s1 destinados a s2).

Este ejemplo es sencillo, pero ilustra cómo el plano de control SDN proporciona servicios del plano de control (en este caso, enrutamiento de la capa de red) que previamente se habían venido implementando mediante un control por router, ejercido en todos y cada uno de los routers de la red. Ahora podemos ver fácilmente cómo un ISP que utilice SDN podría pasar fácilmente de utilizar rutas de mínimo coste, a emplear una técnica de enrutamiento más especializada. De hecho, puesto que el controlador puede ajustar las tablas de flujo como desee, puede implementar *cualquier* forma de reenvío que quiera, simplemente cambiando su software de aplicación de control. Compare esta facilidad de modificación con el caso de un plano de control tradicional, con control por router, en el que habría que cambiar el software de todos los routers (que podrían haber sido suministrados al ISP por múltiples fabricantes independientes).

5.5.4 SDN: pasado y futuro

Aunque el enorme interés en SDN es un fenómeno relativamente reciente, las raíces técnicas de SDN, y en particular la separación de los planos de datos y de control, se remontan a mucho atrás. En 2004, diversas voces [Fearnster 2004, Lakshman 2004, RFC 3746] argumentaron en favor de la separación de los planos de datos y de control de la red. [van der Merwe 1998] describe un marco conceptual de control para redes ATM [Black 1995] con múltiples controladores, cada uno de ellos controlando una serie de comutadores ATM. El proyecto Ethane [Casado 2007] fue pionero en lo que respecta a la noción de una red de comutadores Ethernet simples basados en flujo, con tablas de flujo de tipo correspondencia-acción, un controlador centralizado que gestionara la admisión de flujo y el enrutamiento, y el reenvío de los paquetes para los que no se encontrara correspondencia, desde el comutador al controlador. Ese mismo año 2007 se puso en funcionamiento una red de más de 300 comutadores Ethane. Después, Ethane evolucionó rápidamente para dar lugar al proyecto OpenFlow y, como dice el dicho, ¡el resto es historia!

Hay numerosos proyectos de investigación dirigidos a desarrollar futuras arquitecturas y capacidades SDN. Como hemos visto, la revolución SDN está conduciendo a la rápida sustitución de comutadores y routers monolíticos y dedicados (que cuentan con plano de datos y plano de control) por hardware de comutación sencillo y no diferenciado, al que se añade un sofisticado plano de control software. Una generalización de SDN a la que se conoce con el nombre de NFV (*Network Functions Virtualization*, virtualización de funciones de red) trata también de sustituir rápidamente los dispositivos intermedios sofisticados (como por ejemplo los dispositivos con hardware dedicado y software propietario para la distribución y almacenamiento en caché de contenidos multimedia) por servidores, dispositivos de comutación y dispositivos de almacenamiento simples y no diferenciados [Gember-Jacobson 2014]. Una segunda área de investigación también muy

importante trata de ampliar los conceptos de SDN desde el entorno interno a un sistema autónomo al entorno de operación entre sistemas autónomos [Gupta 2014].

EN LA PRÁCTICA

CASOS DE ESTUDIO DE CONTROLADORES SDN: LOS CONTROLADORES OPENDAYLIGHT Y ONOS

En los primeros días de SDN había un único protocolo SDN (OpenFlow [McKeown 2008; OpenFlow 2009]) y un único controlador SDN (NOX [Gude 2008]). Desde entonces, el número de controladores SDN ha crecido significativamente [Kreutz 2015]. Algunos controladores SDN son propietarios y específicos de una empresa, como por ejemplo ONIX [Koponen 2010], Contrail, de Juniper Networks [Juniper Contrail 2016] y el controlador de Google [Jain 2013] para su red de área extensa B4. Pero hay muchos más controladores que son de código abierto, implementados en diversos lenguajes de programación [Erickson 2013]. Más recientemente, el controlador OpenDaylight [OpenDaylight Lithium 2016] y el controlador ONOS [ONOS 2016] han recibido un considerable apoyo del sector. Ambos son de código abierto y están siendo desarrollados en colaboración con la Linux Foundation.

El controlador OpenDaylight

La Figura 5.17 presenta una vista simplificada de la plataforma controladora SDN OpenDaylight Lithium [OpenDaylight Lithium 2016]. El conjunto principal de componentes del controlador ODL se corresponde estrechamente con los que hemos presentado en la Figura 5.15.

Las *aplicaciones de servicio de red* son las aplicaciones que determinan cómo se llevan a cabo el reenvío y otros servicios del plano de datos, como el de cortafuegos y el de equilibrado de carga, en los conmutadores controlados. A diferencia del controlador canónico de la Figura 5.15, el controlador ODL tiene dos interfaces a través de las cuales las aplicaciones pueden comunicarse con los servicios nativos del controlador y entre sí: las aplicaciones externas se comunican con los módulos del controlador utilizando una API REST de solicitud-respuesta que se ejecuta sobre HTTP. Las aplicaciones internas se comunican entre sí a través de la capa SAL (*Service Abstraction Layer*, capa de abstracción de servicios). Es responsabilidad del diseñador de una aplicación controladora decidir si la implementa externa o internamente; la configuración concreta de aplicaciones mostrada en la Figura 5.17 es un mero ejemplo.

Las *funciones básicas de servicio de red* de ODL forman el corazón del controlador y se corresponden estrechamente con las funciones de gestión del estado de la red que ya hemos visto en la Figura 5.15. La capa SAL es el sistema nervioso del controlador, que permite a las aplicaciones y a los componentes del controlador invocar sus mutuos servicios y suscribirse a los sucesos que generen. También proporciona una interfaz abstracta uniforme a los *protocolos de comunicaciones subyacentes* específicos de la capa de comunicaciones, incluyendo OpenFlow y SNMP (*Simple Network Management Protocol*, protocolo simple de gestión de red: un protocolo de gestión de red del que hablaremos en la Sección 5.7). OVSDB es un protocolo utilizado para gestionar la conmutación de centros de datos, un área de aplicación importante para la tecnología SDN. Hablaremos de las redes de centros de datos en el Capítulo 6.

El controlador ONOS

La Figura 5.18 presenta una vista simplificada del controlador ONOS [ONOS 2016]. De forma similar al controlador canónico de la Figura 5.15, podemos identificar tres capas en el controlador ONOS:

- *Protocolos y abstracciones norte.* Una característica original de ONOS es su infraestructura Intent, que permite a una aplicación solicitar un servicio de alto nivel (por ejemplo, establecer una conexión entre el host A y el host B o, a la inversa, no permitir que el host A y el host B se comuniquen) sin necesidad de conocer los detalles de cómo se implementa dicho servicio. La información de estado se proporciona a las aplicaciones de control de red a través de la API norte, bien síncronamente (mediante consultas) o

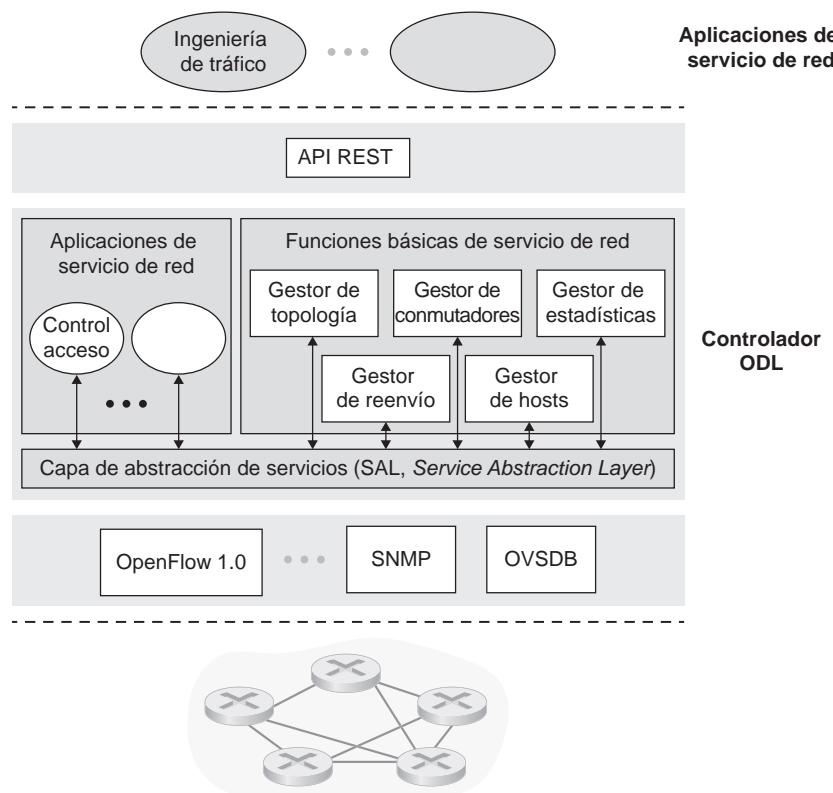


Figura 5.17 ♦ El controlador OpenDaylight.

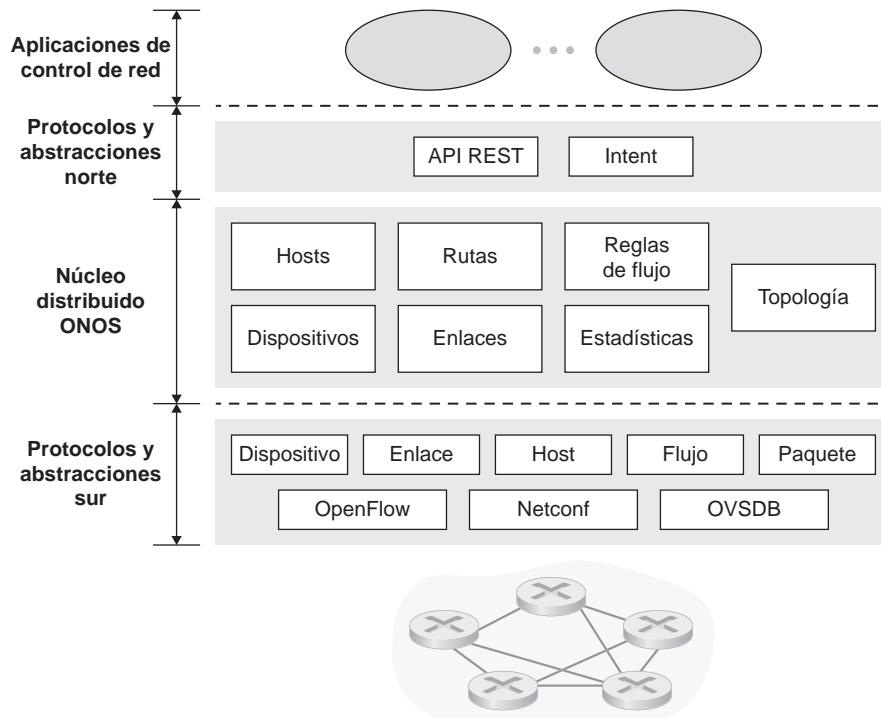


Figura 5.18 ♦ Arquitectura del controlador ONOS.

asíncronamente (a través de retrollamadas a procesos escucha, por ejemplo cuando cambia el estado de la red).

- *Núcleo distribuido.* El estado de los enlaces, hosts y dispositivos de la red se mantiene en el núcleo distribuido de ONOS. ONOS se implanta como un servicio en un conjunto de servidores interconectados, ejecutándose en cada servidor una copia idéntica del software ONOS; a mayor número de servidores, mayor capacidad de servicio. El núcleo de ONOS proporciona los mecanismos de replicación y coordinación del servicio entre instancias, proporcionando a las aplicaciones situadas por encima y a los dispositivos de red situados por debajo la abstracción de unos servicios del núcleo lógicamente centralizados.
 - *Protocolos y abstracciones sur.* Las abstracciones sur enmascaran la heterogeneidad de los hosts, enlaces, commutadores y protocolos subyacentes, permitiendo que el núcleo distribuido sea independiente de los dispositivos y de los protocolos. Debido a esta abstracción, la interfaz sur situada por debajo del núcleo distribuido está a un nivel lógico superior que en nuestro controlador canónico de la Figura 5.15 o en el controlador ODL de la Figura 5.17.
-

5.6 Protocolo de mensajes de control de Internet (ICMP)

Los hosts y los routers utilizan ICMP (*Internet Control Message Protocol*, protocolo de mensajes de control de Internet), especificado en [RFC 792], para intercambiarse información acerca de la capa de red. El uso más típico de ICMP es la generación de informes de error. Por ejemplo, al ejecutar una sesión HTTP podemos encontrarnos con un mensaje de error como “Red de destino inalcanzable”. Este mensaje tiene su origen en ICMP. En algún punto, un router IP no ha podido encontrar una ruta hasta el host especificado en la solicitud HTTP, y dicho router ha creado y enviado un mensaje ICMP a nuestro host para informarle del error.

ICMP a menudo se considera parte de IP pero, en sentido arquitectónico, se encuentra justo encima de IP, ya que los mensajes ICMP son transportados dentro de datagramas IP. Es decir, los mensajes ICMP son transportados como carga útil de IP, al igual que los segmentos TCP o UDP son transportados como carga útil de IP. De forma similar, cuando un host recibe un datagrama IP con ICMP especificado como el protocolo de la capa superior (número de protocolo de la capa superior igual a 1), demultiplexa el contenido del datagrama hacia ICMP, al igual que demultiplexaría el contenido de un datagrama hacia TCP o UDP.

Los mensajes ICMP tienen un campo de tipo y un campo de código, y contienen la cabecera y los 8 primeros bytes del datagrama IP que ha dado lugar a la generación del mensaje ICMP (para que el emisor pueda determinar qué datagrama ha producido el error). En la Figura 5.19 se muestra una serie de tipos de mensajes ICMP seleccionados. Observe que los mensajes ICMP no solo se emplean para señalizar condiciones de error.

El famoso programa ping envía un mensaje ICMP de tipo 8 y código 0 al host especificado. El host de destino, al ver la solicitud de eco, devuelve una respuesta de eco ICMP de tipo 0 y código 0. La mayoría de las implementaciones de TCP/IP soportan el servidor ping directamente en el sistema operativo; es decir, el servidor no es un proceso. El Capítulo 11 de [Stevens 1990] proporciona el código fuente del programa cliente ping. Observe que el programa cliente necesita poder instruir al sistema operativo para generar un mensaje ICMP de tipo 8 y código 0.

Otro interesante mensaje ICMP es el mensaje de regulación del origen. Este mensaje rara vez se emplea en la práctica. Su propósito original era llevar a cabo el control de congestión (permitir a un router congestionado enviar un mensaje ICMP de este tipo a un host para forzarle a reducir su velocidad de transmisión). Hemos visto en el Capítulo 3 que TCP dispone de su propio mecanismo control de congestión que opera en la capa de transporte, sin utilizar ninguna realimentación de la capa de red, como el mensaje ICMP de regulación del origen.

Tipo ICMP	Código	Descripción
0	0	respuesta de eco (para ping)
3	0	red de destino inalcanzable
3	1	host de destino inalcanzable
3	2	protocolo de destino inalcanzable
3	3	puerto de destino inalcanzable
3	6	red de destino desconocida
3	7	host de destino desconocido
4	0	regulación del origen (control de congestión)
8	0	solicitud de eco
9	0	anuncio de router
10	0	descubrimiento de router
11	0	TTL caducado
12	0	Cabecera IP errónea

Figura 5.19 ♦ Tipos de mensajes ICMP.

En el Capítulo 1 hemos introducido el programa Traceroute, el cual nos permite trazar una ruta desde un host a cualquier otro host del mundo. Merece la pena resaltar que Traceroute se implementa con mensajes ICMP. Para determinar los nombres y las direcciones de los routers existentes entre el origen y el destino, el programa Traceroute del origen envía una serie de datagramas IP ordinarios al destino. Cada uno de estos datagramas transporta un segmento UDP con un número de puerto UDP poco probable. El primero de estos datagramas tiene un TTL de 1, el segundo de 2, el tercero de 3, y así sucesivamente. El origen también inicia sendos temporizadores para cada uno de los datagramas. Cuando el datagrama n -ésimo llega al router n -ésimo, este observa que el TTL del datagrama acaba de caducar. De acuerdo con las reglas del protocolo IP, el router descarta el datagrama y envía al origen un mensaje de advertencia ICMP (tipo 11, código 0). Este mensaje de advertencia incluye el nombre del router y su dirección IP. Cuando este mensaje ICMP llega de vuelta al origen, este obtiene el tiempo de ida y vuelta del temporizador, y obtiene también del propio mensaje ICMP el nombre y la dirección IP del router n -ésimo.

¿Cómo sabe un origen Traceroute cuándo dejar de enviar segmentos UDP? Recuerde que el origen incrementa el valor del campo TTL cada vez que envía un datagrama. Por tanto, uno de los datagramas terminará recorriendo el camino completo hasta el host de destino. Dado que ese datagrama contiene un segmento UDP con un número de puerto improbable, el host de destino devuelve al origen un mensaje ICMP de puerto inalcanzable (tipo 3, código 3). Cuando el host de origen recibe este mensaje ICMP, sabe que no tiene que enviar más paquetes de sondeo. (Realmente, el programa estándar Traceroute envía conjuntos de tres paquetes con el mismo TTL; es por ello que la salida de Traceroute proporciona tres resultados para cada TTL.)

De esta forma, el host de origen obtiene el número y la identidad de los routers que existen entre él y el host de destino, así como el tiempo de ida y vuelta entre los dos hosts. Observe que el programa cliente Traceroute tiene que poder instruir al sistema operativo para generar datagramas UDP con valores TTL específicos, y que el sistema operativo también tiene que ser capaz de notificarle la llegada de mensajes ICMP. Ahora que comprende cómo funciona Traceroute, puede volver atrás y practicar con él un poco más.

En RFC 4443 se ha definido una nueva versión de ICMP para IPv6. Además de reorganizar las definiciones existentes de tipos y códigos ICMP, ICMPv6 también añade nuevos tipos y códigos, requeridos por la nueva funcionalidad IPv6. Entre estos se incluyen el tipo “Packet too big” (paquete demasiado grande) y un código de error “unrecognized IPv6 options” (opciones IPv6 no reconocidas).

5.7 Gestión de red y SNMP

Habiendo finalizado nuestro estudio de la capa de red, y cuando ya solo nos queda por delante la capa de enlace, sabemos que una red está compuesta por muchos elementos hardware y software complejos, que interactúan unos con otros: elementos que van desde los enlaces, switches, routers, hosts y otros dispositivos que constituyen los componentes físicos de la red, hasta los muchos protocolos que controlan y coordinan estos dispositivos. Cuando una organización junta centenares o miles de esos componentes para formar una red, se vuelve todo un desafío el trabajo del administrador de la red, que no es otro que mantener la red funcionando. Hemos visto en la Sección 5.5 que el controlador lógicamente centralizado puede ayudar con este proceso, en un contexto SDN. Pero el desafío representado por la gestión de la red ha existido desde bastante antes que SDN, existiendo un rico conjunto de herramientas y técnicas de gestión de red que ayudan al administrador de la red a monitorizarla, gestionarla y controlarla. En esta sección estudiaremos esas herramientas y técnicas.

A menudo suele plantearse la pregunta de “¿Qué es la gestión de red?”. Una definición de la gestión de red muy bien concebida, en una sola frase (aunque hay que reconocer que un poco larga), es la que da [Saydam 1996]:

La gestión de red incluye la implantación, integración y coordinación del hardware, el software y los elementos humanos para monitorizar, probar, sondear, configurar, analizar, evaluar y controlar los elementos y recursos de la red, con el fin de satisfacer los requisitos de tiempo real, de rendimiento operativo y de Calidad de Servicio, a un coste razonable.

Dada esta amplia definición, vamos a cubrir en esta sección únicamente los rudimentos de la gestión de red: la arquitectura, los protocolos y la base de información que un administrador de red utiliza para llevar a cabo su tarea. No hablaremos de los procesos de toma de decisiones de los administradores, en los que hay que tener en cuenta la identificación de fallos [Labovitz 1997; Steinder 2002; Feamster 2005; Wu 2005; Teixeira 2006], la detección de anomalías [Lakhina 2005; Barford 2009], el diseño/ingeniería de la red para satisfacer los acuerdos de nivel de servicio (SLA, *Service Level Agreement*) contratados [Huston 1999a] y otros temas. Nuestro enfoque será, por tanto, deliberadamente reducido; el lector interesado puede consultar las referencias indicadas, el excelente libro de gestión de red de Subramanian [Subramanian 2000] y el tratamiento más detallado de la gestión de red disponible en el sitio web del presente libro.

5.7.1 El marco conceptual de la gestión de red

La Figura 5.20 muestra los componentes clave de la gestión de red:

- El **servidor de gestión** es una aplicación, normalmente con intervención humana, que se ejecuta en una estación central de gestión de red situada en el centro de operaciones de red (NOC, *Network Operations Center*). El servidor de gestión es el punto focal de la actividad de administración de la red; controla la recopilación, procesamiento, análisis y/o visualización de la información de gestión de la red. Es aquí donde se inician las acciones para el control del comportamiento de la red y donde el administrador interactúa con los dispositivos que forman la red.
- Un **dispositivo gestionado** es un equipo de red (incluyendo su software) que forma parte de una red gestionada. Un dispositivo gestionado puede ser un host, un router, un switch, un dispositivo

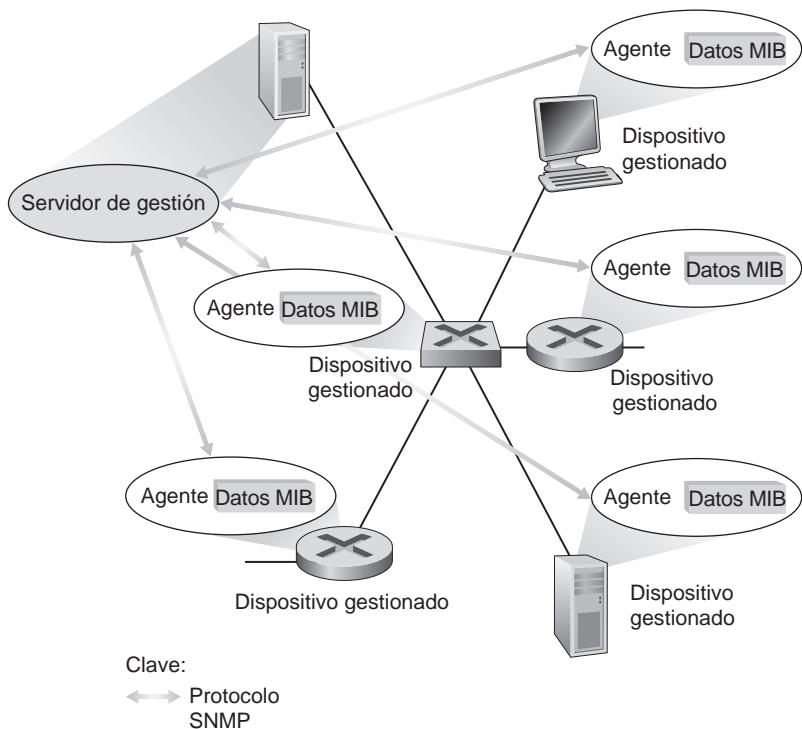


Figura 5.20 ◆ Elementos de gestión de red: servidor de gestión, dispositivos gestionados, datos MIB, agentes remotos, SNMP.

intermediario, un módem, un termómetro o cualquier otro dispositivo conectado a la red. Dentro de un dispositivo gestionado pueden existir diversos **objetos gestionados**. Estos objetos gestionados son los propios elementos hardware contenidos en el dispositivo gestionado (por ejemplo, una tarjeta de interfaz de red no es más que uno de los componentes de un host o un router) y los parámetros de configuración para los distintos elementos hardware y software (por ejemplo, un protocolo de enrutamiento interior al sistema autónomo, como OSPF).

- Cada objeto gestionado de un dispositivo gestionado tiene asociados distintos elementos de información, que se recopilan en una **base de información de gestión (MIB, Management Information Base)**; como veremos, los valores de estos elementos de información están a disposición del servidor de gestión (que en muchos casos también puede modificarlos). Un objeto de la MIB puede ser un contador, como por ejemplo el número de datagramas IP descartados en un router debido a errores en la cabecera del datagrama, o el número de segmentos UDP recibidos en un host; o puede ser información de carácter descriptivo, como por ejemplo la versión del software que se está ejecutando en un servidor DNS; o puede ser información de estado, como por ejemplo si un cierto dispositivo está funcionando correctamente; o puede ser información específica del protocolo, como por ejemplo una ruta hasta un destino. Los objetos MIB se especifican en un lenguaje de descripción de datos denominado SMI (*Structure of Management Information*, estructura de información de gestión) [RFC 2578; RFC 2579; RFC 2580]. Se utiliza un lenguaje de definición formal para garantizar que la sintaxis y la semántica de los datos de gestión de red estén bien definidas y sean no ambiguas. Los objetos MIB relacionados se agrupan en módulos MIB. A mediados de 2015, los distintos documentos RFC definían casi 400 módulos MIB, existiendo un número mucho mayor aún de módulos MIB privados (específicos del fabricante).
- En cada dispositivo gestionado reside también un **agente de gestión de red**, que es un proceso que se ejecuta en el dispositivo gestionado y que se comunica con el servidor de gestión, llevando

a cabo acciones locales en el dispositivo gestionado bajo control y por orden del servidor de gestión. El agente de gestión de red es similar al agente de enrutamiento que vimos en la Figura 5.2.

- El último componente del marco conceptual de la gestión de red es el **protocolo de gestión de red**. El protocolo se ejecuta entre el servidor de gestión y los dispositivos gestionados, permitiendo al servidor consultar el estado de los dispositivos gestionados y llevar a cabo acciones de manera indirecta en estos dispositivos a través de sus agentes. Los agentes pueden utilizar el protocolo de gestión de red para informar al servidor de gestión de que se han producido sucesos excepcionales (por ejemplo, fallos de componentes o violación de los umbrales de rendimiento). Es importante recalcar que el protocolo de gestión de red no se encarga él mismo de administrar la red; simplemente proporciona una serie de capacidades que un administrador puede utilizar para gestionar (“monitorizar, probar, sondear, configurar, analizar, evaluar y controlar”) la red. Se trata de una distinción sutil, pero de gran importancia. En la siguiente sección hablaremos del protocolo SNMP (Simple Network Management Protocol, protocolo simple de gestión de red) de Internet.

5.7.2 El protocolo SNMP

El **protocolo simple de gestión de red** versión 2 (SNMPv2) [RFC 3416] es un protocolo de la capa de aplicación que se utiliza para transportar mensajes de información y control para la gestión de red entre un servidor de gestión y un agente que actúa por cuenta del servidor de gestión. El uso más común de SNMP es el modo de solicitud-respuesta, en el que un servidor de gestión SNMP envía una solicitud a un agente SNMP, el cual la recibe, lleva a cabo ciertas acciones y envía una respuesta a dicha solicitud. Normalmente, una solicitud se utilizará para consultar (recuperar) o modificar (establecer) los valores de objetos MIB asociados con un dispositivo gestionado. Un segundo uso común de SNMP es cuando un agente envía un mensaje no solicitado, conocido como mensaje TRAP, a un servidor de gestión. Los mensajes TRAP se utilizan para notificar a un servidor de gestión una situación excepcional (por ejemplo, la activación o desactivación de un enlace) que ha dado lugar a cambios en los valores de los objetos MIB.

SNMPv2 define siete tipos de mensajes, conocidos genéricamente como unidades de datos de protocolo (PDU, *Protocol Data Unit*), que se enumeran en la Tabla 5.2 y se describen a continuación. El formato de la PDU se muestra en la Figura 5.21.

- Las PDU GetRequest, GetNextRequest y GetBulkRequest son enviadas desde un servidor de gestión a un agente para solicitar el valor de uno o más objetos MIB del dispositivo gestionado por el agente. Los objetos MIB cuyos valores están siendo solicitados se especifican en la parte de las variables de la PDU. GetRequest, GetNextRequest y GetBulkRequest difieren en la granularidad de sus solicitudes de datos. GetRequest puede solicitar un conjunto arbitrario de valores MIB; pueden utilizarse múltiples GetNextRequests para recorrer secuencialmente una lista o tabla de objetos MIB; GetBulkRequest permite devolver un gran bloque de datos, evitando la sobrecarga en que se incurre si se envían varios mensajes GetRequest o GetNextRequest. En los tres casos, el agente responde con una PDU Response que contiene los identificadores de objeto y sus valores asociados.
- Un servidor de gestión utiliza la PDU SetRequest para establecer el valor de uno o más objetos MIB en un dispositivo gestionado. Un agente responde con una PDU Response con el estado de error “noError”, con el fin de confirmar que el valor ha sido definido.
- Un servidor de gestión utiliza una PDU InformRequest para notificar a otro servidor de gestión cierta información MIB que es remota al servidor receptor.
- La PDU Response es enviada normalmente por un dispositivo gestionado al servidor de gestión en respuesta a un mensaje de solicitud recibido de ese servidor. Con esta PDU se devuelve la información solicitada.

Tipo de PDU de SNMPv2	Emisor-receptor	Descripción
GetRequest	gestor a agente	Obtener el valor de una o más instancias de objeto MIB.
GetNextRequest	gestor a agente	Obtener el valor de la siguiente instancia de objeto MIB en una lista o tabla.
GetBulkRequest	gestor a agente	Obtener valores en un bloque grande de datos, por ejemplo, en una tabla de gran tamaño.
InformRequest	gestor a gestor	Informar a la entidad gestora remota de valores MIB remotos a su acceso.
SetRequest	gestor a agente	Establecer el valor de una o más instancias de objeto MIB.
Response	agente a gestor o gestor a gestor	Generado en respuesta a: GetRequest, GetNextRequest, GetBulkRequest, SetRequest PDU, 0 InformRequest
SNMPv2-Trap	agente a gestor	Informar al gestor del número de un suceso excepcional.

Tabla 5.2 ♦ Tipos de PDU de SNMPv2.

- El último tipo de PDU de SNMPv2 es el mensaje TRAP. Los mensajes TRAP son generados asíncronamente; es decir, no son generados en respuesta a una solicitud recibida, sino en respuesta a un suceso para el que el servidor de gestión requiere una notificación. El documento RFC 3418 define tipos de mensajes TRAP bien conocidos, como el arranque en frío o caliente de un dispositivo, la activación o desactivación de un enlace, la pérdida de un vecino o un suceso de fallo de autenticación. Una solicitud TRAP recibida no requiere ninguna respuesta por parte del servidor de gestión.

Dada la naturaleza solicitud-respuesta de SNMP, es conveniente señalar aquí que, aunque las PDU SNMP pueden ser transportadas por medio de muchos protocolos de transporte distintos, normalmente son transportadas en la carga útil de un datagrama UDP. De hecho, el documento RFC 3417

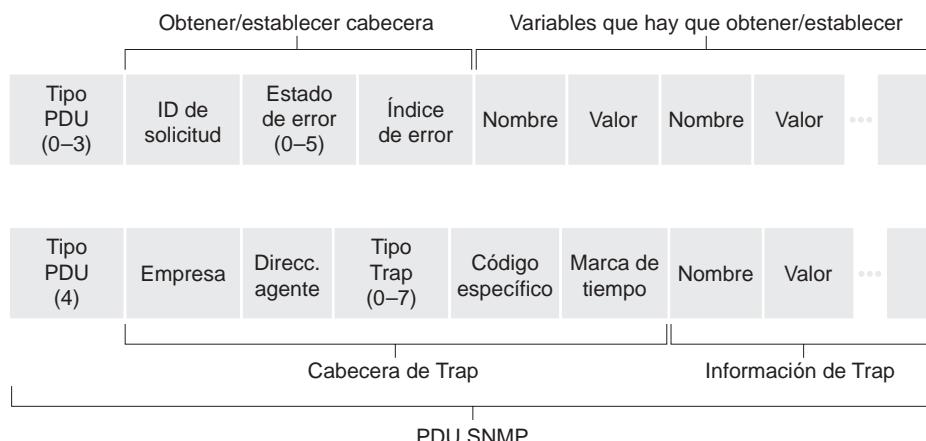


Figura 5.21 ♦ Formato de la PDU de SNMP.

establece que UDP es “la correspondencia de transporte preferida”. Sin embargo, dado que UDP es un protocolo de transporte no fiable, no existe ninguna garantía de que una solicitud, o su respuesta, sea recibida por el destino. El servidor de gestión utiliza el campo ID de solicitud de la PDU (véase la Figura 5.21) para numerar las solicitudes que envía a un agente; la respuesta del agente toma su ID de solicitud de la solicitud recibida. Por tanto, el campo ID de solicitud puede ser utilizado por el servidor de gestión para detectar las solicitudes y respuestas perdidas. Dependerá del servidor de gestión decidir si retransmite una solicitud si no ha recibido la respuesta correspondiente transcurrido un cierto periodo de tiempo. En particular, el estándar SNMP no obliga a aplicar ningún procedimiento en concreto para las retransmisiones y ni siquiera indica si la retransmisión debe llevarse a cabo. Solo obliga a que el servidor de gestión “actúe de forma responsable respecto a la frecuencia y duración de las retransmisiones”. ¡Por supuesto, esto lleva a preguntarse cómo debe actuar un protocolo “responsable”!

SNMP ha evolucionado en tres versiones. Los diseñadores de SNMPv3 han señalado que “SNMPv3 puede interpretarse como SNMPv2 con una serie de capacidades adicionales de seguridad y administración” [RFC 3410]. Realmente, existen cambios en SNMPv3 respecto de SNMPv2, pero en ninguna parte son más evidentes estos cambios que en el área de la administración y la seguridad. El papel central de la seguridad en SNMPv3 era particularmente importante, ya que la falta de una adecuada seguridad hizo que SNMP se usara principalmente para monitorizar, más que para cuestiones de control (por ejemplo, rara vez se emplea `SetRequest` en SNMPv1). De nuevo, vemos que la seguridad —un tema del que trataremos en el Capítulo 8— es un problema crítico, pero un problema cuya importancia, de nuevo, solo se ha reconocido un poco tarde, habiéndose “añadido” entonces las soluciones.

5.8 Resumen

Ya hemos completado estos dos capítulos dedicados al estudio del núcleo de la red: un estudio que comenzó con el análisis del plano de datos de la capa de red en el Capítulo 4 y que ha finalizado aquí, con el repaso al plano de control de dicha capa. Hemos visto que el plano de control es la lógica de red que controla no solo cómo se reenvía un datagrama entre routers, a lo largo de una ruta extremo a extremo que va desde el host de origen hasta el host de destino, sino también cómo se configuran y gestionan los componentes y servicios de la capa de red.

Hemos visto que hay dos enfoques generales para construir un plano de control: el *control por router* tradicional (en el que se ejecuta un algoritmo de enrutamiento en todos y cada uno de los routers y el componente de enrutamiento del router se comunica con sus homólogos de otros routers) y el control mediante *redes definidas por software* (SDN), en el que un controlador lógicamente centralizado calcula y distribuye las tablas de reenvío que deben utilizar todos y cada uno de los routers. Hemos estudiado dos algoritmos fundamentales de enrutamiento para calcular las rutas de menor coste en un grafo —el enrutamiento por estado de los enlaces y el enrutamiento de vectores de distancia— en la Sección 5.2; estos algoritmos encuentran su aplicación tanto en el control por router como en el control SDN. Dichos algoritmos son la base de dos protocolos de enrutamiento muy difundidos en Internet, OSPF y BGP, de los que hemos hablado en las secciones 5.3 y 5.4. En la Sección 5.5 hemos analizado el enfoque SDN de implementación del plano de control de la capa de red, investigando las aplicaciones de control de red SDN, el controlador SDN y el protocolo OpenFlow para la comunicación entre el controlador y los dispositivos controlados por SDN. En las secciones 5.6 y 5.7 hemos repasado algunas de las interioridades de la gestión de una red IP: ICMP (el protocolo de mensajes de control de Internet) y SNMP (el protocolo simple de gestión de red).

Habiendo completado nuestro estudio de la capa de red, vamos a descender un escalón más por la pila de protocolos, concretamente hasta la capa de enlace. Al igual que la capa de red, la capa de enlace también forma parte de todos y cada uno de los dispositivos conectados a la red. Pero en el siguiente capítulo veremos que la capa de enlace tiene la tarea mucho más localizada de transferir

los paquetes entre nodos situados en un mismo enlace o LAN. Aunque esta tarea puede parecer en principio trivial, comparada con las tareas que desempeña la capa de red, veremos que la capa de enlace implica una serie de cuestiones importantes y fascinantes, que nos mantendrán ocupados durante bastante tiempo.

Problemas y cuestiones de repaso

Capítulo 5 Cuestiones de repaso

SECCIÓN 5.1

- R1. ¿A qué nos referimos al hablar de un plano de control basado en el control por router? En tales casos, cuando decimos que los planos de datos y de control de la red están implementados “monolíticamente”, ¿qué queremos decir?
- R2. ¿A qué nos referimos al hablar de un plano de control basado en un control lógicamente centralizado? En tales casos, ¿el plano de datos y el plano de control se implementan en el mismo dispositivo o en dispositivos diferentes? Explique su respuesta.

SECCIÓN 5.2

- R3. Indique las similitudes y diferencias entre los algoritmos de enrutamiento centralizados y distribuidos. Proporcione un ejemplo de protocolo de enrutamiento que adopte un enfoque centralizado y otro descentralizado.
- R4. Indique las similitudes y diferencias entre los algoritmos de enrutamiento de estado de los enlaces y por vector de distancias.
- R5. ¿Qué es el problema de la “cuenta hasta infinito” en el enrutamiento por vector de distancias?
- R6. ¿Es necesario que todos los sistemas autónomos utilicen el mismo algoritmo de enrutamiento interno? ¿Por qué o por qué no?

SECCIONES 5.3–5.4

- R7. ¿Por qué se utilizan diferentes protocolos de enrutamiento internos de un sistema autónomo y entre sistemas autónomos en Internet?
- R8. Verdadero o falso: cuando un router OSPF envía su información de estado de los enlaces, solo la envía a los vecinos directamente conectados. Explique su respuesta.
- R9. ¿A qué se llama *área* en un sistema autónomo OSPF? ¿Por qué se introdujo el concepto de área?
- R10. Defina los siguientes términos e indique en qué se diferencian: *subred*, *prefijo* y *ruta BGP*.
- R11. ¿Cómo utiliza BGP el atributo NEXT-HOP? ¿Cómo utiliza el atributo AS-PATH?
- R12. Describa cómo un administrador de red de un ISP de nivel superior puede implementar ciertas políticas al configurar BGP.
- R13. Verdadero o falso: cuando un router BGP recibe una ruta anunciada por su vecino, debe añadir su propia identidad a la ruta recibida y luego enviar esa nueva ruta a todos sus vecinos. Explique su respuesta.

SECCIÓN 5.5

- R14. Describa el papel principal de la capa de comunicaciones, de la capa de gestión del estado de la red y de la capa de aplicaciones de control de red en un controlador SDN.
- R15. Suponga que quiere implementar un nuevo protocolo de enrutamiento en el plano de control SDN. ¿En qué capa implementaría ese protocolo? Explique su respuesta.

- R16. ¿Qué tipos de mensajes fluyen a través de las API norte y sur de un controlador SDN? ¿Quién es el receptor de los mensajes que el controlador envía a través de la interfaz sur y quién envía mensajes al controlador a través de la interfaz norte?
- R17. Describa el propósito de dos tipos de mensajes OpenFlow (a su elección) enviados desde un dispositivo controlado al controlador. Describa el propósito de dos tipos de mensajes OpenFlow (a su elección) enviados desde el controlador a un dispositivo controlado.
- R18. ¿Cuál es el propósito de la capa de abstracción de servicios en el controlador SDN OpenDaylight?

SECCIONES 5.6–5.7

- R19. Enumere cuatro tipos distintos de mensajes ICMP.
- R20. ¿Qué dos tipos de mensajes ICMP se reciben en el host emisor que está ejecutando el programa *Traceroute*?
- R21. Defina los siguientes términos en el contexto de SNMP: *servidor de gestión, dispositivo gestionado, agente de gestión de red y MIB*.
- R22. ¿Cuál es el objetivo de los mensajes SNMP *GetRequest* y *SetRequest*?
- R23. ¿Cuál es el objetivo del mensaje SNMP *Trap*?

Problemas



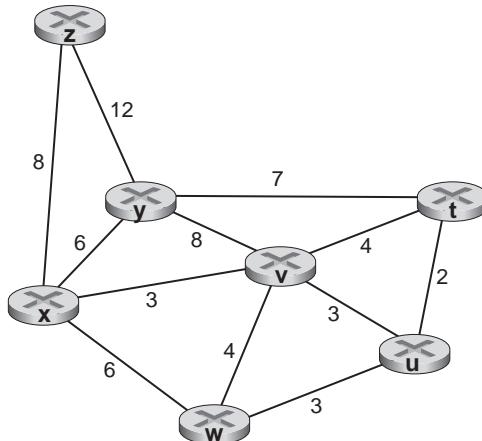
Nota de vídeo

Algoritmo de Dijkstra:
explicación y ejemplo

- P1. En la Figura 5.3, enumere las rutas desde *y* hasta *u* que no contienen bucles.

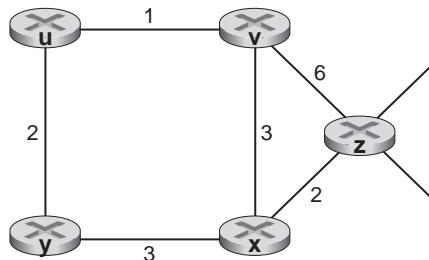
- P2. Repita el Problema P1 para las rutas de *x* a *z*, de *z* a *u* y de *z* a *w*.

- P3. Considere la siguiente red. Utilizando los costes de enlace indicados, aplique el algoritmo de la ruta más corta de Dijkstra para calcular la ruta más corta desde *x* a todos los restantes nodos de red. Indique cómo funciona el algoritmo elaborando una tabla similar a la Tabla 5.1.

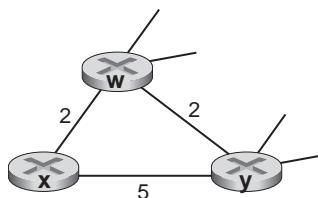


- P4. Considere la red mostrada en el Problema P3. Utilizando el algoritmo de Dijkstra y una tabla similar a la Tabla 5.1, haga lo siguiente:
- Calcule la ruta más corta desde *t* a todos los demás nodos de la red.
 - Calcule la ruta más corta desde *u* a todos los demás nodos de la red.
 - Calcule la ruta más corta desde *v* a todos los demás nodos de la red.
 - Calcule la ruta más corta desde *w* a todos los demás nodos de la red.
 - Calcule la ruta más corta desde *y* a todos los demás nodos de la red.
 - Calcule la ruta más corta desde *z* a todos los demás nodos de la red.

- P5. Utilice la red que se muestra a continuación y suponga que cada nodo inicialmente conoce los costes hasta cada uno de sus vecinos. Utilizando el algoritmo de vector de distancias, especifique las entradas de la tabla de distancias para el nodo z .



- P6. Considere una topología general (es decir, no la red concreta mostrada más arriba) y una versión síncrona del algoritmo de vector de distancias. Suponga que en cada iteración un nodo intercambia sus vectores distancia con sus vecinos y recibe los vectores distancia de ellos. Suponiendo que el algoritmo se inicia con cada nodo conociendo solo los costes de sus vecinos inmediatos, ¿cuál es el número máximo de iteraciones requerido antes de que el algoritmo distribuido converja? Justifique su respuesta.
- P7. Considere el fragmento de red mostrado a continuación. x solo tiene dos vecinos conectados, w e y . w tiene una ruta de coste mínimo al destino u (no mostrado) de 5 e y tiene una ruta de coste mínimo a u de 6. Las rutas completas desde w e y a u (y entre w e y) no se muestran. Todos los costes de enlace de la red tienen valores enteros estrictamente positivos.



- a. Indique el vector de distancias de x para los destinos w , y y u .
- b. Indique un cambio en el coste del enlace para $c(x,w)$ o $c(x,y)$ tal que x informe a sus vecinos de una nueva ruta de coste mínimo a u , como resultado de ejecutar el algoritmo de vector de distancias.
- c. Indique un cambio en el coste del enlace para $c(x,w)$ o $c(x,y)$ tal que x no informe a sus vecinos de una nueva ruta de coste mínimo a u , como resultado de ejecutar el algoritmo de vector de distancias.
- P8. Considere la topología de tres nodos mostrada en la Figura 5.6. En lugar de tener los costes de enlace mostrados en dicha figura, los costes de enlace son $c(x,y) = 3$, $c(y,z) = 6$, $c(z,x) = 4$. Calcule las tablas de distancias después del paso de inicialización y después de cada iteración de una versión síncrona del algoritmo de vector de distancias (como hemos hecho anteriormente al explicar la Figura 5.6).
- P9. Considere el problema de la cuenta hasta infinito en el enrutamiento por vector de distancias. ¿Se producirá dicho problema si disminuimos el coste de un enlace? ¿Por qué? ¿Qué ocurre si conectamos dos nodos que no tienen un enlace?
- P10. Demuestre que al aplicar el algoritmo de vector de distancias en la Figura 5.6, cada valor del vector de distancias $D(x)$ es no creciente y finalmente se estabilizará en un número finito de pasos.
- P11. Considere la Figura 5.7. Suponga que existe otro router w , conectado a los routers y y z . Los costes de todos los enlaces son los siguientes: $c(x,y) = 4$, $c(x,z) = 50$, $c(y,w) = 1$,

$c(z,w) = 1$, $c(y,z) = 3$. Suponga que se utiliza inversa envenenada en el algoritmo de enrutamiento por vector de distancias.

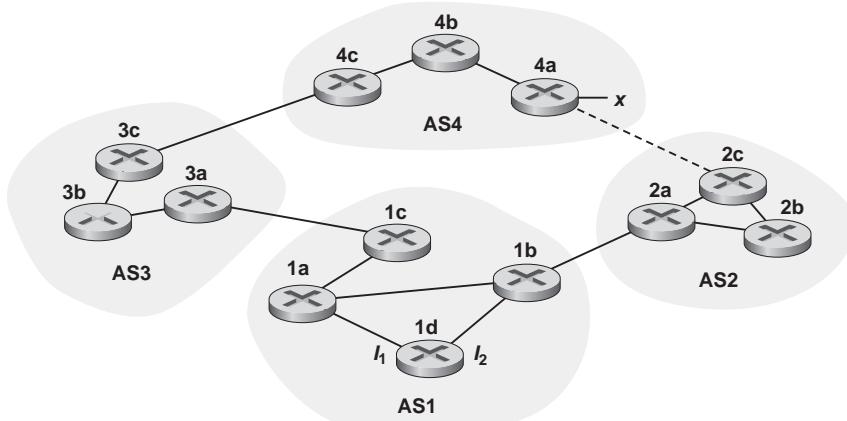
- Cuando el enrutamiento por vector de distancias se estabiliza, los routers w , y y z se informan de sus respectivas distancias a x . ¿Cuáles son los valores de esas distancias?
- Ahora suponga que el coste del enlace entre x e y aumenta a 60. ¿Se producirá un problema de cuenta hasta infinito aunque se utilice inversa envenenada? ¿Por qué? Si existe el problema de cuenta hasta infinito, entonces ¿cuántas iteraciones serán necesarias para que el enrutamiento por vector de distancias alcance de nuevo un estado estable? Justifique su respuesta.
- ¿Cómo modificaría $c(y,z)$ para que no existiera el problema de cuenta hasta infinito si $c(y,x)$ cambia de 4 a 60?

P12. Describa cómo puede detectarse en BGP la existencia de bucles en las rutas.

P13. ¿Un router BGP elegirá siempre la ruta sin bucles con la longitud más corta de la secuencia AS-PATH? Justifique su respuesta.

P14. Considere la red mostrada a continuación. Suponga que los sistemas autónomos AS3 y AS4 están ejecutando OSPF como protocolo de enrutamiento interno. Suponga que AS1 y AS4 están ejecutando RIP como protocolo de enrutamiento interno. Suponga por último que se utilizan eBGP e iBGP para el protocolo de enrutamiento entre sistemas autónomos. Además, inicialmente *no* existe enlace físico entre AS2 y AS4.

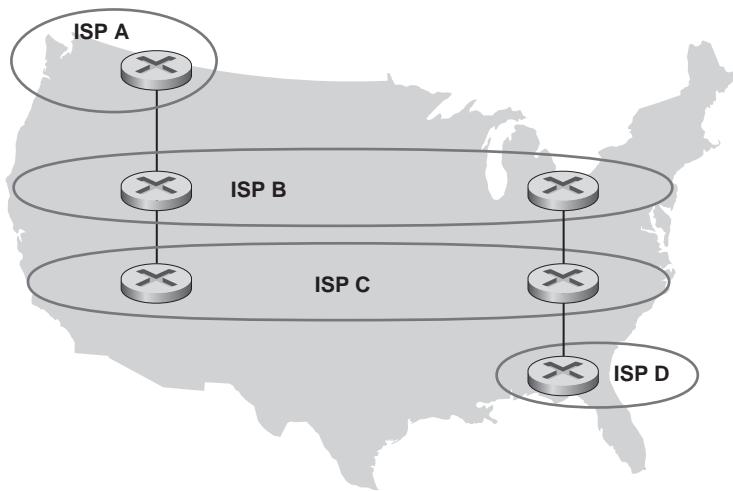
- ¿De qué protocolo de enrutamiento aprende el router 3c acerca del prefijo x : OSPF, RIP, eBGP o iBGP?
- ¿De qué protocolo de enrutamiento aprende el router 3a acerca de x ?
- ¿De qué protocolo de enrutamiento aprende el router 1c acerca de x ?
- ¿De qué protocolo de enrutamiento aprende el router 1d acerca de x ?



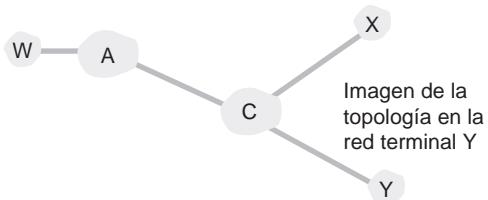
P15. Continuando con el problema anterior, una vez que el router 1d aprende acerca de x incluirá una entrada (x, I) en su tabla de reenvío.

- Para esta entrada, ¿ I será igual a I_1 o a I_2 ? Explique por qué en una frase.
- Ahora suponga que existe un enlace físico entre AS2 y AS4 (mostrado mediante una línea de puntos en la figura). Suponga que el router 1d aprende que x es accesible a través de AS2 y de AS3. ¿ I será igual a I_1 o a I_2 ? Explique por qué en una frase.
- Ahora suponga que existe otro sistema autónomo AS5, que conecta la ruta entre AS2 y AS4 (no se muestra en el diagrama). Suponga que el router 1d aprende que x es accesible a través de AS2 AS5 AS4, así como de AS3 AS4. ¿ I será igual a I_1 o a I_2 ? Explique por qué en una frase.

- P16. Considere la red mostrada a continuación. El ISP B proporciona un servicio troncal nacional al ISP A regional. El ISP C ofrece un servicio troncal nacional al ISP D regional. Cada ISP consta de un sistema autónomo. Los pares B y C se comunican entre sí por dos puntos utilizando BGP. Considere el tráfico que va de A a D. B preferiría manipular dicho tráfico hacia C por el enlace de la Costa Oeste (de modo que C tendría que absorber el coste de transportar el tráfico a través del país), mientras que C preferiría obtener ese tráfico a través del enlace con B de la Costa Este, en cuyo caso B tendría que transportar el tráfico a través del país. ¿Qué mecanismo BGP podría utilizar C para que B llevara el tráfico de A-a-D al punto de contacto de la Costa Este? Para responder a esta pregunta, tendrá que ahondar en la especificación de BGP.



- P17. En la Figura 5.13, considere la información de rutas que llega a las redes terminales (*stub*) W, X e Y. Basándose en la información disponible en W y X, ¿cuáles son sus respectivas imágenes de la topología de la red? Justifique su respuesta. La imagen de la topología en Y se muestra a continuación.



- P18. Considere la Figura 5.13. B nunca reenviaría tráfico destinado a Y a través de X basándose en el enrutamiento BGP. Pero existen algunas aplicaciones muy populares en las que los paquetes de datos se dirigen primero a X y luego fluyen hacia Y. Identifique una de tales aplicaciones y describa cómo los paquetes de datos siguen una ruta que no ha sido determinada por el enrutamiento BGP.
- P19. En la Figura 5.13, suponga que existe otra red terminal V que es un cliente del ISP A. Suponga que B y C tienen una relación de pares y que A es cliente tanto de B como de C. Suponga también que A preferiría que el tráfico destinado a W procediera solo de B, y que el tráfico destinado a V procediera de B o de C. ¿Cómo podría anunciar A sus rutas a B y C? ¿Qué rutas del sistema autónomo recibe C?
- P20. Suponga que los sistemas autónomos X y Z no están directamente conectados sino que se conectan a través del sistema autónomo Y. Suponga también que X tiene un acuerdo de comunicación entre pares con Y, y que Y tiene un acuerdo similar con Z. Por último, suponga

que Z desea transferir todo el tráfico de Y pero no el de X. ¿Permite BGP implementar esta política a Z?

- P21. Considere las dos formas en que tiene lugar la comunicación entre una entidad gestora y un dispositivo gestionado: el modo solicitud-respuesta y TRAP. ¿Cuáles son las ventajas y los inconvenientes de estos dos métodos, en términos de (1) costes, (2) tiempo de notificación cuando se producen sucesos excepcionales y (3) robustez con respecto a los mensajes perdidos entre la entidad gestora y el dispositivo?
- P22. En la Sección 5.7 hemos visto que era preferible transportar mensajes SNMP en datagramas UDP no fiables. ¿Por qué cree que los diseñadores de SNMP eligieron UDP en lugar de TCP como protocolo de transporte para SNMP?

Tarea de programación con socket

Al final del Capítulo 2 hay cuatro tareas de programación con sockets. A continuación le proponemos una quinta tarea, que utiliza ICMP, un protocolo del que hemos hablado en este capítulo.

Tarea 5: Ping ICMP

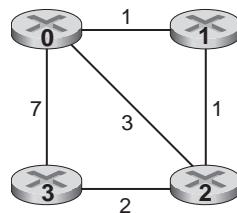
Ping es una popular aplicación de red que se utiliza para comprobar, desde una ubicación remota, si un determinado host está activo y es alcanzable. También se utiliza a menudo para medir la latencia entre el host cliente y el host de destino. Funciona enviando paquetes ICMP de “solicitud de eco” (es decir, paquetes ping) al host de destino y esperando a recibir paquetes ICMP de “respuesta de eco” (es decir, paquetes pong). Ping mide el RTT, contabiliza las pérdidas de paquete y calcula un resumen estadístico de múltiples intercambios ping-pong (el mínimo, la media, el máximo y la desviación estándar del tiempo de ida y vuelta).

En esta práctica de laboratorio tendrá que escribir su propia aplicación Ping en Python. La aplicación utilizará ICMP. Pero, para que el programa sea sencillo, no vamos a seguir exactamente la especificación oficial de RFC 1739. Observe que solo necesita escribir el lado del cliente del programa, ya que la funcionalidad necesaria en el lado del servidor está integrada en casi todos los sistemas operativos. Puede encontrar los detalles completos de esta tarea, así como fragmentos significativos del código Python, en el sitio web <http://www.pearsonhighered.com/cs-resources>.

Tarea de programación

En esta tarea de programación tendrá que escribir un conjunto “distribuido” de procedimientos que implemente un enrutamiento asíncrono distribuido por vector de distancias para la red que se muestra más abajo.

Tendrá que escribir las siguientes rutinas que se “ejecutarán” de forma asíncrona dentro del entorno emulado proporcionado para esta tarea. Para el nodo 0, escribirá las rutinas siguientes:



- *rtinit0()*. Se llamará a esta rutina una vez al principio de la emulación. *rtinit0()* no tiene argumentos. Debe inicializar su tabla de distancias en el nodo 0 para reflejar los costes directos

de 1, 3 y 7 a los nodos 1, 2 y 3, respectivamente. En la figura de encima, todos los enlaces son bidireccionales y los costes en ambas direcciones son idénticos. Después de inicializar la tabla de distancias y cualquier otra estructura de datos que necesiten sus rutinas del nodo 0, deberá entonces enviar a sus vecinos directamente conectados (en este caso, 1, 2 y 3) el coste de sus rutas de coste mínimo a los demás nodos de la red. La información de coste mínimo se envía a los nodos vecinos mediante un paquete de actualización de enrutamiento llamando a la rutina *tolayer2()*, como se describe en la tarea completa. El formato del paquete de actualización de enrutamiento también está descrito en la tarea completa.

- *rtupdate0(struct rtpkt *rcvdpkt)*. Se llamará a esta rutina cuando el nodo 0 reciba un paquete de enrutamiento que le haya enviado uno de sus vecinos directamente conectados. El parámetro **rcvdpkt* es un puntero al paquete que ha recibido. *rtupdate0()* es el “núcleo” del algoritmo de vector de distancias. Los valores recibidos en un paquete de actualización de enrutamiento procedente de algún otro nodo *i* contienen los costes de la ruta más corta actual de *i* hacia todos los demás nodos de la red. *rtupdate0()* utiliza estos valores para actualizar su propia tabla de distancias (como especifica el algoritmo de vector de distancias). Si su propio coste mínimo a otro nodo cambia como resultado de la actualización, el nodo 0 informa de este cambio del coste mínimo a sus vecinos directamente conectados enviándoles un paquete de enrutamiento. Recuerde que, en el algoritmo de vector de distancias, solo los nodos directamente conectados intercambiarán paquetes de enrutamiento. Por tanto, los nodos 1 y 2 se comunicarán entre sí, pero los nodos 1 y 3 no lo harán.

Para los nodos 1, 2 y 3 se definen rutinas similares. Por tanto, tendrá que escribir ocho procedimientos en total: *rtinit0()*, *rtinit1()*, *rtinit2()*, *rtinit3()*, *rtupdate0()*, *rtupdate1()*, *rtupdate2()* y *rtupdate3()*. Estas rutinas implementarán conjuntamente un cálculo asíncrono y distribuido de las tablas de distancias para la topología y los costes mostrados en la figura anterior.

Puede encontrar todos los detalles acerca de esta tarea de programación, así como el código C que tendrá que crear en el entorno hardware/software simulado en <http://www.pearsonhighered.com/cs-resource>. También hay disponible una versión Java de la tarea.

Práctica de laboratorio con Wireshark

En el sitio web del libro, www.pearsonhighered.com/cs-resources, encontrará una práctica de laboratorio con Wireshark que examina el uso del protocolo ICMP en los comandos ping y traceroute.

UNA ENTREVISTA CON...

Jennifer Rexford

Jennifer Rexford es profesora en el departamento de Ciencias de la Computación de la Universidad de Princeton. Sus investigaciones se centran en conseguir que las redes de computadoras sean más fáciles de diseñar y gestionar, con un énfasis especial en los protocolos de enrutamiento. Entre 1996 y 2004 formó parte del departamento de Rendimiento y Gestión de Red de AT&T Labs-Research. Mientras trabajaba en AT&T diseñó técnicas y herramientas para la realización de medidas de red, ingeniería de tráfico y configuración de routers, que fueron implantadas en la red troncal de AT&T. Jennifer es coautora del libro "Web Protocols and Practice: Networking Protocols, Caching, and Traffic Measurement," publicado por Addison-Wesley en mayo de 2001. Ha sido la moderadora de ACM SIGCOMM entre 2003 y 2007. Se graduó en Ingeniería Eléctrica en la Universidad de Princeton en 1991, doctorándose en Ingeniería Eléctrica y Ciencias de la Computación en la Universidad de Michigan en 1996. En 2004, Jennifer fue galardonada con el premio Grace Murray Hopper de la ACM por su excelencia como joven profesional en el campo de las Ciencias de la Computación y fue incluida en la lista TR-100 del MIT de principales innovadores de menos de 35 años.



¿Podría describirnos uno o dos de los proyectos más excitantes en los que haya trabajado durante su carrera? ¿Cuáles fueron los principales desafíos de diseño?

Cuando trabajaba como investigadora en AT&T, varios de nosotros diseñamos una nueva forma de gestionar el enrutamiento en las redes troncales de los ISP. Tradicionalmente, los operadores de red configuran cada router individualmente, y estos routers ejecutan protocolos distribuidos para calcular las rutas a través de la red. Estábamos convencidos de que la gestión de red sería más simple y flexible si los operadores de red pudieran ejercer un control directo sobre el modo en que los routers reenvían el tráfico, basándose en una visión global de la topología y el tráfico de toda la red. La plataforma de control de enrutamiento RCP (*Routing Control Platform*) que diseñamos y construimos podía calcular las rutas para toda la red troncal de AT&T usando una única computadora barata y permitía controlar los routers heredados sin necesidad de modificaciones. Para mí, este proyecto fue muy excitante, porque tuvimos una idea provocadora, desarrollamos un sistema funcional y, al final, realizamos una implantación real en una red que estaba en operación. Si saltamos adelante unos cuantos años, vemos que las redes definidas por software (SDN) se han convertido en una tecnología dominante, y existen protocolos estándar (como OpenFlow) que facilitan mucho la tarea de decir a los commutadores subyacentes qué tienen que hacer.

¿Cómo cree que deberían evolucionar en el futuro las redes definidas por software?

En una ruptura radical con respecto al pasado, el software del plano de control puede ser creado por muchos programadores diferentes, no solo por las empresas que desarrollan equipos de red. Sin embargo, a diferencia de las aplicaciones que se ejecutan en un servidor o en un teléfono inteligente, esas aplicaciones controladoras deben trabajar *juntas* para gestionar el mismo tráfico. Los operadores de red no desean realizar un equilibrado de carga con una parte del tráfico y un enrutamiento con otra parte del tráfico, sino que quieren realizar tanto el equilibrado de carga como el enrutamiento de un mismo tráfico. Las futuras plataformas controladoras

SDN deberían ofrecer buenas abstracciones de programación para *componer* múltiples aplicaciones controladoras escritas independientemente, de forma que trabajen juntas. Hablando en términos más generales, las buenas abstracciones de programación pueden hacer que resulte más fácil crear aplicaciones controladoras, sin tener que preocuparse de detalles de bajo nivel como las entradas de las tablas de flujo, los contadores de tráfico, los patrones de bits en las cabeceras de los paquetes, etc. Asimismo, aunque un controlador SDN está lógicamente centralizado, la red sigue consistiendo en una colección distribuida de dispositivos. Los controladores futuros deberían ofrecer buenas abstracciones para actualizar las tablas de flujo en toda la red, de modo que las aplicaciones puedan razonar acerca de lo que les sucede a los paquetes en tránsito mientras los dispositivos están siendo actualizados. La de las abstracciones de programación para el software del plano de control es un área excitante de investigación interdisciplinar entre las redes de computadoras, los sistemas distribuidos y los lenguajes de programación, con una posibilidad real de tener un impacto práctico en los años venideros.

¿Cuál cree que es el futuro de las redes y de Internet?

El de las redes es un campo excitante, porque las aplicaciones y las tecnologías subyacentes están cambiando constantemente. ¡Continuamente nos estamos reinventando a nosotros mismos! ¿Quién habría predicho hace solo diez años el predominio de los teléfonos inteligentes, que permiten a los usuarios móviles acceder tanto a las aplicaciones existentes, como a nuevos servicios basados en la ubicación? La aparición de la computación en la nube está cambiando de modo fundamental la relación entre los usuarios y la aplicación que ejecutan, y los sensores y actuadores en red (la “Internet de las cosas”) están haciendo posible una pléthora de nuevas aplicaciones (¡y de vulnerabilidades de seguridad!). El ritmo de innovación es verdaderamente fascinante.

La red subyacente es un componente crucial de todas estas innovaciones. Y, sin embargo, la red constituye un notorio “obstáculo”: limitando el rendimiento, comprometiendo la fiabilidad, restringiendo las aplicaciones y complicando la implantación y la gestión de servicios. Deberíamos intentar que la red del futuro sea tan invisible como el aire que respiramos, de modo que nunca obstaculice las nuevas ideas y los servicios valiosos. Para ello, necesitamos elevar el nivel de abstracción por encima de los protocolos y dispositivos de red individuales (¡y sus respectivos acrónimos!), de modo que podamos razonar acerca de la red y de los objetivos de alto nivel del usuario en su conjunto.

¿Qué personas le han inspirado profesionalmente?

Me he sentido inspirada durante mucho tiempo por Sally Floyd, del International Computer Science Institute. Sus investigaciones siempre eran pertinentes, centradas en los importantes desafíos a los que se enfrenta Internet. Profundizaba enormemente en cuestiones difíciles, hasta que entendía completamente el problema y el espacio de soluciones, y se dedicaba en cuerpo y alma a hacer que “las cosas avanzaran”, como por ejemplo introduciendo sus ideas en estándares de protocolo y equipos de red. Asimismo, contribuía a la comunidad, prestando servicios profesionales en numerosas organizaciones de investigación y de estandarización y creando herramientas (como los ampliamente utilizados simuladores ns-2 y ns-3) que permiten a otros investigadores avanzar. Se jubiló en 2009, pero su influencia se dejará sentir en este campo durante muchos años.

¿Qué recomendaría a los estudiantes que quieran desarrollar una carrera profesional en el campo de las ciencias de la computación y las redes?

El de las redes es un campo inherentemente interdisciplinar. A las redes se les aplican técnicas derivadas de los avances en otras muchas disciplinas, de áreas tan diversas como la teoría de colas, la teoría de juegos, la teoría de control, los sistemas distribuidos, la optimización de redes, los lenguajes de programación, el aprendizaje de máquinas, los algoritmos, las estructuras de datos, etc. Creo que ser experto en un campo relacionado, o colaborar estrechamente con expertos en esos campos, es una excelente manera de reforzar los fundamentos de las redes,

para aprender a construir redes que sean merecedoras de la confianza de la sociedad. Más allá de las disciplinas teóricas, las redes son excitantes porque creamos sistemas reales que son utilizados por la gente real. Dominar la manera de diseñar y construir sistemas —obteniendo experiencia en sistemas operativos, arquitectura de computadoras, etc.— es otra fantástica manera de ampliar nuestros conocimientos de redes para ayudar a hacer del mundo un lugar mejor.

La capa de enlace y las redes de área local

En los dos capítulos anteriores hemos visto que la capa de red proporciona un servicio de comunicaciones entre dos hosts de red *cualesquiera*. Entre los dos hosts, los datagramas viajan a través de una serie de enlaces de comunicaciones, cableados e inalámbricos, que comienzan en el host de origen, pasan a través de una serie de commutadores de paquetes (switches y routers) y terminan en el host de destino. A medida que continuamos bajando por la pila de protocolos, desde la capa de red a la capa de enlace, surge de forma natural la pregunta de cómo se envían los paquetes a través de los *enlaces individuales* que forman la ruta de comunicación de terminal a terminal. ¿Cómo se encapsulan los datagramas de la capa de red en las tramas de la capa de enlace para la transmisión a través de un enlace individual? ¿Se emplean diferentes protocolos de la capa de enlace en los distintos enlaces a lo largo de la ruta de comunicación? ¿Cómo se resuelven los conflictos de transmisión en los enlaces de difusión? ¿Existe direccionamiento en la capa de enlace y, en caso afirmativo, cómo opera el direccionamiento de la capa de enlace con el direccionamiento de la capa de red que hemos estudiado en el Capítulo 4? ¿Y cuál es exactamente la diferencia entre un switch y un router? Responderemos a estas y otras importantes cuestiones a lo largo del capítulo.

Al analizar la capa de enlace, nos encontramos con que hay dos tipos de canales fundamentalmente distintos de la capa de enlace. El primer tipo está compuesto por los canales de difusión, que conectan varios hosts en las redes de área local (LAN) inalámbricas, en las redes por satélite y en las redes de acceso híbridas de fibra y cable coaxial (HFC). En un canal de difusión hay muchos hosts conectados a un mismo canal de comunicaciones, por lo que se hace necesario utilizar lo que se denomina un protocolo de acceso al medio para coordinar la transmisión de las tramas. En algunos casos, se puede emplear un controlador central para coordinar las transmisiones; en otros casos, son los propios hosts los que coordinan estas transmisiones. El segundo tipo de canal de la capa de enlace es el canal de comunicaciones punto a punto, como el que existe entre dos routers conectados mediante un enlace de larga distancia, o entre la computadora de sobremesa de un usuario y el switch Ethernet al que está conectada. La coordinación del acceso a un enlace punto a punto es más simple; el material de referencia disponible en el sitio web del libro proporciona una exposición detallada acerca del Protocolo punto a punto (PPP, *Point-to-Point Protocol*), el cual se

utiliza en configuraciones tan variadas como el servicio de acceso a través de una línea telefónica o en el transporte de tramas punto a punto a alta velocidad a través de enlaces de fibra óptica.

En este capítulo exploraremos diversos conceptos y tecnologías importantes de la capa de enlace. Examinaremos en profundidad la detección y corrección de errores, un tema que hemos visto por encima en el Capítulo 3. Consideraremos diversas redes de acceso y las redes LAN conmutadas, incluyendo Ethernet, que es, con diferencia, la tecnología LAN cableada predominante. También abordaremos las redes LAN virtuales y las redes para centros de datos. Aunque WiFi, y más en general las redes LAN inalámbricas, es un tema relacionado con la capa de enlace, pospondremos nuestro estudio de este importante tema hasta el Capítulo 7.

6.1 Introducción a la capa de enlace

Comencemos proporcionando alguna terminología útil. En este capítulo nos resultará conveniente referirnos a cualquier dispositivo que ejecute un protocolo de la capa de enlace (esto es, de la capa 2) como **nodo**. Los nodos incluyen a los hosts, los routers, los switches y los puntos de acceso WiFi (que veremos en el Capítulo 7). También nos referimos a los canales de comunicación que conectan nodos adyacentes a lo largo de la ruta de comunicaciones con el nombre de **enlaces**. Para que un datagrama pueda ser transferido desde el host de origen al de destino, debe moverse a través de cada uno de los *enlaces individuales* que forman la ruta extremo a extremo. Por ejemplo, en la red empresarial mostrada en la parte inferior de la Figura 6.1, considere el envío de un datagrama desde uno de los hosts inalámbricos a uno de los servidores. Este datagrama realmente atraviesa seis enlaces: un enlace WiFi entre el host emisor y el punto de acceso WiFi, un enlace Ethernet entre el punto de acceso y un switch, un enlace entre ese switch y el router, un enlace entre los dos routers, un enlace Ethernet entre el router y un switch y, finalmente un enlace Ethernet entre el switch y el servidor. En un determinado enlace, un nodo transmisor encapsula el datagrama en una **trama de la capa de enlace** y transmite la trama a través del enlace.

Para poder comprender el funcionamiento de la capa de enlace y cómo esta se relaciona con la capa de red, vamos a utilizar una analogía del sector del transporte. Piense en una agencia de viajes que planifica un viaje para un turista que quiere ir desde Princeton, Nueva Jersey, a Lausana, Suiza. La agencia de viajes decide que lo más conveniente para el turista es tomar una limusina en Princeton hasta el aeropuerto JFK, luego un avión desde dicho aeropuerto hasta el aeropuerto de Ginebra y, finalmente, un tren desde el aeropuerto de Ginebra hasta la estación de tren de Lausana. Una vez que la agencia de viajes hace las tres reservas, es responsabilidad de la empresa de limusinas de Princeton llevar al turista hasta el aeropuerto JFK; de la misma forma, será responsabilidad de la compañía aérea transportar al turista desde el aeropuerto JFK al de Ginebra y será responsabilidad de la compañía ferroviaria suiza llevar al turista desde Ginebra hasta Lausana. Cada uno de estos tres segmentos del viaje es un segmento “directo” entre dos ubicaciones “adyacentes”. Observe que los tres segmentos de transporte son gestionados por diferentes empresas y utilizan modos de transporte completamente distintos (limusina, avión y tren). Aunque los modos de transporte son diferentes, todos ellos proporcionan el servicio básico de transportar pasajeros desde una ubicación a otra adyacente. En esta analogía del sector del transporte, el turista sería un datagrama, cada uno de los segmentos de transporte sería un enlace de comunicaciones, el modo de transporte sería un protocolo de la capa de enlace y la agencia de viajes sería un protocolo de enrutamiento.

6.1.1 Servicios proporcionados por la capa de enlace

Aunque el servicio básico de cualquier capa de enlace es mover un datagrama desde un nodo hasta otro adyacente a través de un único enlace de comunicaciones, los detalles del servicio proporcionado pueden variar de un protocolo de la capa de enlace a otro. Entre los posibles servicios que un protocolo de la capa de enlace puede ofrecer se incluyen:

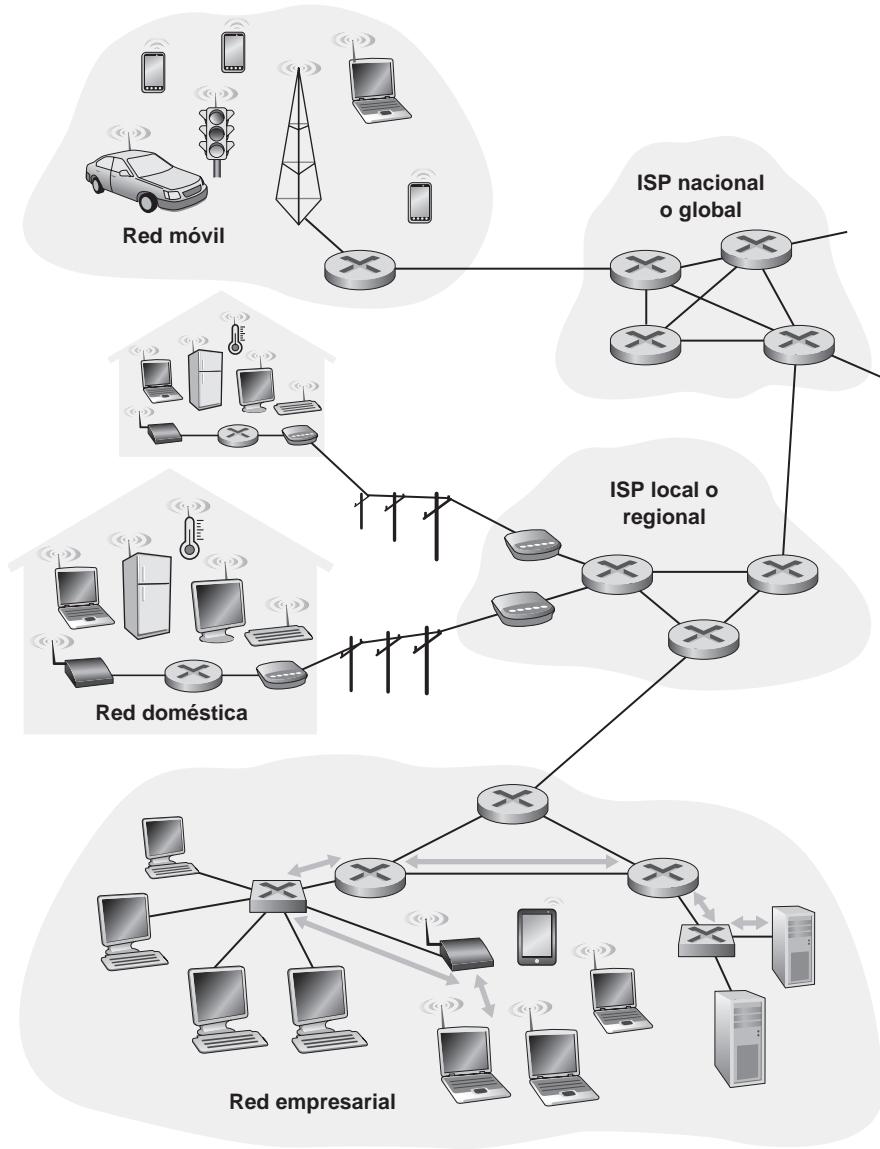


Figura 6.1 ♦ Seis saltos de la capa de enlace entre el host inalámbrico y el servidor.

- **Entramado**. Casi todos los protocolos de la capa de enlace encapsulan cada datagrama de la capa de red dentro de una trama de la capa de enlace antes de transmitirla a través del enlace. Una trama consta de un campo de datos, en el que se inserta el datagrama de la capa de red, y de una serie de campos de cabecera. La estructura de la trama está especificada por el protocolo de la capa de enlace. Veremos diferentes formatos de trama cuando examinemos una serie de protocolos específicos de la capa de enlace en la segunda mitad de este capítulo.
- **Acceso al enlace**. Un protocolo de control de acceso al medio (MAC, *Medium Access Control*) especifica las reglas que se utilizan para transmitir una trama a través del enlace. Para los enlaces punto a punto que tengan un único emisor en un extremo del enlace y un único receptor en el otro extremo, el protocolo MAC es muy simple (o no existe): el emisor puede enviar una trama siempre que el enlace esté inactivo. El caso más interesante es cuando hay varios nodos compartiendo un mismo enlace de difusión, en cuyo caso se presenta el denominado problema

del acceso múltiple. En ese caso, el protocolo MAC sirve para coordinar la transmisión de las tramas de los múltiples nodos.

- *Entrega fiable.* Cuando un protocolo de la capa de enlace proporciona un servicio de entrega fiable, garantiza que va a transportar cada datagrama de la capa de red a través del enlace sin errores. Recuerde que ciertos protocolos de la capa de transporte (como TCP) también proporcionan un servicio de entrega fiable. De forma similar a los servicios de entrega fiable de la capa de transporte, el servicio de entrega fiable de la capa de enlace suele implementarse mediante confirmaciones y retransmisiones (véase la Sección 3.4). A menudo se utiliza un servicio de entrega fiable de la capa de enlace en aquellos enlaces que suelen presentar altas tasas de error, como por ejemplo en los enlaces inalámbricos, con el objetivo de corregir los errores localmente (en el enlace en el que se producen los errores), en lugar de obligar a que un protocolo de la capa de transporte o de la de aplicación realice una retransmisión de datos extremo a extremo. Sin embargo, la entrega fiable en la capa de enlace puede considerarse una sobrecarga innecesaria en aquellos enlaces que tengan una baja tasa de errores de bit, incluyendo los enlaces de fibra, los coaxiales y muchos enlaces de cobre de par trenzado. Por esta razón, muchos protocolos de la capa de enlace para enlaces cableados no proporcionan un servicio de entrega fiable.
- *Detección y corrección de errores.* El hardware de la capa de enlace en un nodo receptor pudiera llegar a decidir, incorrectamente, que un bit contenido en una trama es cero cuando fue transmitido como un uno, y viceversa. Dichos errores de bit se introducen debido a la atenuación de las señales y al ruido electromagnético. Puesto que no hay necesidad de reenviar un datagrama que contenga un error, muchos protocolos de la capa de enlace proporcionan un mecanismo para detectar dichos errores de bit. Esto se lleva a cabo haciendo que el nodo transmisor incluya bits de detección de errores en la trama y que el nodo receptor realice una comprobación de errores. Recuerde, de los Capítulos 3 y 4, que las capas de transporte y de red de Internet también ofrecen una forma limitada de detección de errores: la suma de comprobación de Internet. La detección de errores en la capa de enlace normalmente es más sofisticada y se implementa en hardware. La corrección de errores es similar a la detección de errores, salvo porque el receptor no solo detecta si hay bits erróneos en la trama, sino que también determina exactamente en qué puntos de la trama se han producido los errores (luego corrige esos errores).

6.1.2 ¿Dónde se implementa la capa de enlace?

Antes de profundizar en los detalles de la capa de enlace, consideremos la cuestión de dónde se implementa esta capa. Nos centraremos aquí en un sistema terminal, puesto que ya hemos visto en el Capítulo 4 cómo se implementa la capa de enlace en una tarjeta de línea de un router. ¿Cómo se implementa la capa de enlace de un host, por hardware o por software? ¿Se implementa en una tarjeta o chip separados? ¿Cómo se realiza la interfaz con el resto del hardware del host y con el resto de los componentes del sistema operativo?

La Figura 6.2 muestra la arquitectura típica de un host. En su mayor parte, la capa de enlace se implementa en un **adaptador de red**, también denominado a veces **tarjeta de interfaz de red (NIC, Network Interface Card)**. El corazón de la tarjeta adaptadora de red es el controlador de la capa de enlace, que normalmente es un único chip de propósito especial que implementa muchos de los servicios de la capa de enlace (entrnamado, acceso al enlace, detección de errores, etc.). Por tanto, buena parte de la funcionalidad del controlador de la capa de enlace se implementa por hardware. Por ejemplo, el adaptador 710 de Intel [Intel 2016] implementa los protocolos Ethernet que estudiaremos en la Sección 6.5; el controlador Atheros AR5006 [Atheros 2016] implementa los protocolos WiFi 802.11 que estudiaremos en el Capítulo 7. Hasta finales de la década de 1990, la mayoría de los adaptadores de red eran tarjetas físicamente separadas, como por ejemplo tarjetas PCMCIA o una tarjeta insertable que podía introducirse en una ranura de tarjeta PCI del PC, pero los adaptadores de red se suelen integrar cada vez más en la placa base del host, utilizando una configuración que se denomina “LAN sobre placa base”.

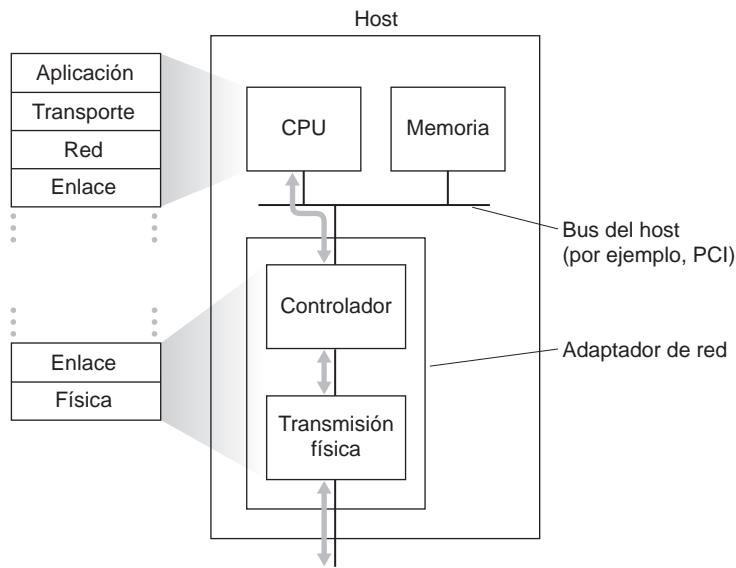


Figura 6.2 ♦ Adaptador de red: su relación con otros componentes del host y con la funcionalidad de la pila de protocolos.

En el lado emisor, el controlador toma un datagrama que haya sido creado y almacenado en la memoria del host por las capas superiores de la pila de protocolos, encapsula ese datagrama en una trama de la capa de enlace (rellenando los diversos campos de la trama) y luego transmite la trama al enlace de comunicaciones, de acuerdo con el protocolo de acceso al enlace. En el lado receptor, un controlador recibe la trama completa y extrae el datagrama de la capa de red. Si la capa de enlace realiza detección de errores, entonces será el controlador del emisor quien se encargue de configurar los bits de detección de errores en la cabecera de la trama, mientras que el controlador del receptor llevará a cabo la detección de errores.

La Figura 6.2 muestra un adaptador de red conectado a un bus del host (por ejemplo, un bus PCI o PCI-X), de modo que a ojos de los restantes componentes del host se parece bastante a cualquier otro dispositivo de E/S. La Figura 6.2 también muestra que, mientras que la mayor parte de la capa de enlace está implementada por hardware, una parte de esa capa se implementa en un software que se ejecuta en la CPU del host. Los componentes software de la capa de enlace normalmente implementan la función de más alto nivel de esa capa como, por ejemplo, el ensamblado de la información de direccionamiento de la capa de enlace y la activación del hardware del controlador. En el lado receptor, el software de la capa de enlace responde a las interrupciones procedentes del controlador (por ejemplo, debidas a la recepción de una o más tramas), se encarga de gestionar las condiciones de error y pasa el datagrama hacia la capa de red. Por tanto, la capa de enlace es una combinación de hardware y software; es el lugar de la pila de protocolos en donde el software se encuentra con el hardware. En [Intel 2016] puede encontrar una introducción comprensible (así como una descripción detallada) del controlador XL710, desde el punto de vista del programador de software.

6.2 Técnicas de detección y corrección de errores

En la sección anterior hemos indicado que la **detección y corrección de errores de nivel de bit** (la detección y corrección de los bits corruptos en una trama de la capa de enlace enviada desde un nodo a otro nodo vecino físicamente conectado a él) son dos servicios ofrecidos a menudo por

la capa de enlace. Hemos visto en el Capítulo 3 que en la capa de transporte se ofrecen también a menudo los servicios de detección y corrección de errores. En esta sección examinaremos algunas de las técnicas más simples que pueden utilizarse para detectar y, en algunos casos, corregir dichos errores de bit. Un tratamiento completo de la teoría e implementación de estas técnicas constituye por sí mismo el tema de muchos libros de texto (por ejemplo, [Schwartz 1980] o [Bertsekas 1991]), pero el tratamiento que realizaremos aquí es necesariamente breve. Nuestro objetivo es desarrollar simplemente en el lector una percepción intuitiva de las capacidades proporcionadas por las técnicas de detección y corrección de errores, y ver cómo funcionan unas cuantas técnicas simples y cómo se emplean en la práctica dentro de la capa de enlace.

La Figura 6.3 ilustra la configuración para nuestro análisis. En el nodo emisor, los datos D que hay que proteger frente a los errores de bit se complementan con una serie de bits de detección y corrección de errores (EDC, *Error Detection and Correction*). Normalmente, los datos que hay que proteger incluyen no solo el datagrama recibido de la capa de red para su transmisión a través del enlace, sino también la información de direccionamiento de la capa de enlace, los números de secuencia y otros campos de la cabecera de la trama de enlace. Tanto D como EDC se envían hacia el nodo receptor en una trama de la capa de enlace. En el nodo receptor, se recibe una secuencia de bits formada por D' y EDC' . Observe que D' y EDC' pueden diferir de las secuencias D y EDC originales, como resultado de alteraciones de los bits sufridas durante el tránsito.

El desafío para el receptor consiste en determinar si D' coincide con la secuencia D original, teniendo en cuenta que lo único que ha recibido son las secuencias D' y EDC' . Es importante observar las palabras exactas utilizadas en el símbolo de decisión mostrado en la Figura 6.3 (lo que preguntamos es si se ha detectado un error, no si se ha producido un error). Las técnicas de detección y corrección de errores permiten al receptor detectar en ocasiones, *pero no siempre*, que se han producido errores en los bits. Incluso utilizando bits de detección de errores pueden seguir existiendo **errores de bit no detectados**; es decir, el receptor podría perfectamente ser inconsciente de que la información recibida contiene errores en los bits. Como consecuencia, el receptor podría entregar un datagrama corrupto a la capa de red, o no ser consciente de que el contenido de un campo de la cabecera de la trama se ha corrompido. Por tanto, lo que intentaremos será elegir un esquema de detección de errores que haga que la probabilidad de que se produzcan estos casos sea pequeña. Por regla general, cuanto más sofisticadas son las técnicas de detección y corrección de errores (es

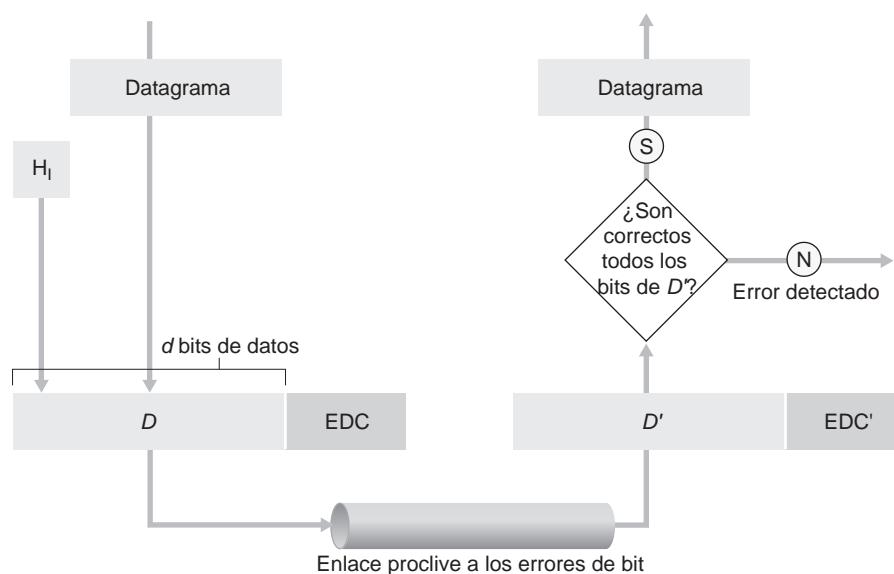


Figura 6.3 ♦ Escenario para la detección y corrección de errores.

decir, cuanto menor sea la probabilidad de que se produzcan errores de bit que no sean detectados), mayores son los recursos adicionales necesarios: harán falta más cálculos para obtener y transmitir un mayor número de bits de detección y corrección de errores.

Examinemos ahora tres técnicas para detectar los errores en los datos transmitidos: comprobaciones de paridad (para ilustrar las ideas básicas que subyacen a las técnicas de detección y corrección de errores), métodos basados en suma de comprobación (que suelen utilizarse más en la capa de transporte) y códigos de redundancia cíclica (que normalmente se emplean más en la capa de enlace en un adaptador).

6.2.1 Comprobaciones de paridad

Quizá la forma más simple de detección de errores sea el uso de un único **bit de paridad**. Suponga que la información que hay que enviar, D en la Figura 6.4, tiene d bits. En un esquema de paridad par, el emisor simplemente incluye un bit adicional y selecciona su valor de modo que el número total de 1s en los $d + 1$ bits (la información original más un bit de paridad) sea par. En los esquemas de paridad impar, el valor del bit de paridad se selecciona de modo que exista un número impar de 1s. La Figura 6.4 ilustra un esquema de paridad par, en el que el único bit de paridad se almacena en un campo separado.

La operación del receptor también es muy simple cuando se utiliza un único bit de paridad. El receptor solo necesita contar el número de 1s dentro de los $d + 1$ bits recibidos. Si se está utilizando un esquema de paridad par y se encuentra un número impar de bits con valor 1, el receptor sabrá que se ha producido al menos un error de bit. De forma más precisa, sabrá que se ha producido un número *ímpar* de errores de bit.

Pero, ¿qué sucede si se produce un número par de errores de bit? Puede comprobar fácilmente que esto haría que el error no fuera detectado. Si la probabilidad de que se produzcan errores en los bits es pequeña y si podemos asumir que los errores que se producen en dos bits sucesivos tienen lugar de forma independiente, la probabilidad de que se produzcan múltiples errores de bit en un único paquete será extremadamente pequeña. En estos casos, podría bastar con utilizar un único bit de paridad. Sin embargo, las medidas realizadas muestran que los errores, más que tener lugar independientemente, suelen agruparse formando “ráfagas”. En condiciones de ráfagas de error, la probabilidad de que se produzcan errores no detectados en una trama protegida mediante un único bit de paridad puede aproximarse al 50 por ciento [Spragins 1991]. Evidentemente, necesitamos un esquema de detección de errores más robusto (y, afortunadamente, en la práctica suelen emplearse esos otros esquemas). Pero antes de examinar los esquemas de detección de errores utilizados en la práctica, vamos a analizar una generalización simple del mecanismo de un único bit de paridad; esta generalización nos permitirá comprender mejor las técnicas de corrección de errores.

La Figura 6.5 muestra una generalización bidimensional del esquema basado en un único bit de paridad. Aquí, los d bits de D se dividen en i filas y j columnas. Para cada una de esas filas y columnas calculamos un valor de paridad. Los $i + j + 1$ bits de paridad resultantes serán los bits de detección de errores utilizados en la trama de la capa de enlace.

Suponga ahora que se produce un único error de bit en los d bits de información originales. Con este esquema de **paridad bidimensional**, detectaremos el error en la paridad tanto de la columna como de la fila que contienen el bit erróneo. De este modo, el receptor no solo podrá *detectar* el hecho de que se ha producido un único error de bit, sino que puede utilizar los índices de la

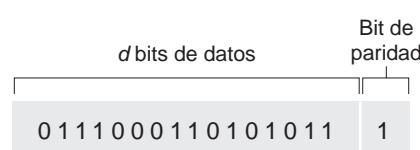


Figura 6.4 ♦ Paridad par de un bit.

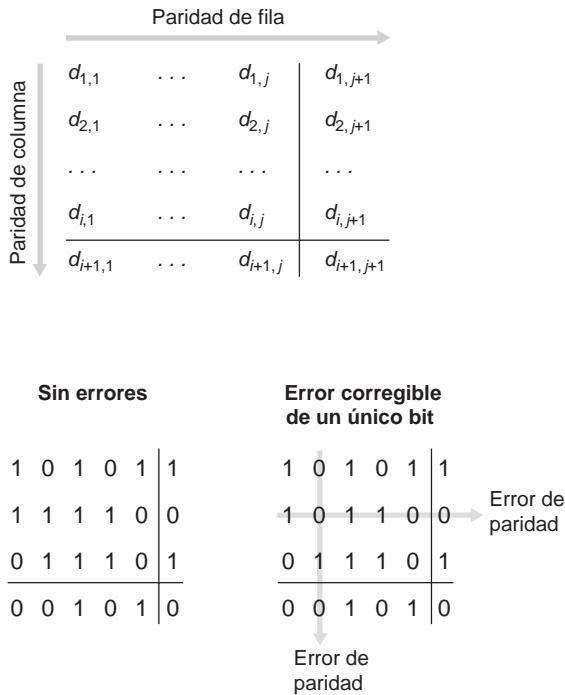


Figura 6.5 ♦ Paridad par bidimensional.

columna y de la fila que presentan errores de paridad para identificar realmente el bit corrompido y *corregir* dicho error. La Figura 6.5 muestra un ejemplo en el que el bit de valor 1 en la posición (2, 2) está corrompido, habiéndose cambiado por un 0; se trata de un error que es tanto detectable como corregible en el receptor. Aunque nuestra explicación se ha centrado en los d bits originales de información, un único error dentro de los propios bits de paridad también es detectable y corregible. Los esquemas de paridad bidimensional también pueden detectar (¡pero no corregir!) cualquier combinación de dos errores dentro de un paquete. En los problemas incluidos al final del capítulo se analizan otras propiedades del esquema de paridad bidimensional.

La capacidad del receptor para detectar y corregir errores a la vez se conoce con el nombre de **Corrección de errores hacia adelante (FEC, Forward Error Correction)**. Estas técnicas se suelen utilizar comúnmente en los dispositivos de almacenamiento y reproducción de audio, como por ejemplo los CD de audio. En un entorno de red, las técnicas FEC también pueden emplearse por sí mismas o en conjunción con técnicas ARQ de la capa de enlace, similares a las que hemos examinado en el Capítulo 3. Las técnicas FEC son valiosas porque pueden reducir el número de retransmisiones realizadas por el emisor pero, lo que quizás sea más importante, también permiten la corrección inmediata de errores en el receptor, lo que evita tener que esperar el retardo de propagación de ida y vuelta necesario para que el emisor reciba un paquete NAK y para que el paquete retransmitido se propague de nuevo hacia el receptor; esta es una característica potencialmente muy importante para las aplicaciones de red en tiempo real [Rubenstein 1998] o para los enlaces (como los enlaces en el espacio profundo) que presenten grandes retardos de propagación. Entre las referencias en las que se examina el uso de las técnicas FEC en los protocolos de control de errores podemos citar [Biersack 1992; Nonnenmacher 1998; Byers 1998; Shacham 1990].

6.2.2 Métodos basados en la suma de comprobación

En las técnicas de suma de comprobación, los d bits de datos de la Figura 6.4 se tratan como una secuencia de enteros de k bits. Un método simple de suma de comprobación consiste en sumar estos

enteros de k bits y utilizar la suma resultante como bits de detección de errores. La **suma de comprobación de Internet** está basada en este enfoque: los bytes de datos se tratan como enteros de 16 bits y se suman. Después, se utiliza el complemento a 1 de esta suma para formar la suma de comprobación de Internet que se incluye en la cabecera del segmento. Como se ha visto en la Sección 3.3, el receptor verifica la suma de comprobación calculando el complemento a 1 de la suma de los datos recibidos (incluyendo la suma de comprobación) y comprobando si el resultado tiene todos los bits a 1. Si alguno de los bits es un 0, eso indicará que se ha producido un error. El documento RFC 1071 explica en detalle el algoritmo de suma de comprobación de Internet y su implementación. En los protocolos TCP y UDP la suma de comprobación de Internet se calcula sobre todos los campos (incluyendo los campos de cabecera y de datos). En IP, la suma de comprobación se calcula sobre la cabecera IP (dado que el segmento UDP o TCP ya tiene su propia suma de comprobación). En otros protocolos, como por ejemplo XTP [Strayer 1992], se calcula una suma de comprobación sobre la cabecera y otra sobre todo el paquete.

Los métodos de suma de comprobación requieren relativamente poca sobrecarga de paquete. Por ejemplo, la suma de comprobación en TCP y UDP solo utiliza 16 bits. Sin embargo, estas sumas proporcionan una protección relativamente débil frente a los errores si las comparamos con las comprobaciones de redundancia cíclica, de las que hablaremos a continuación y que se utilizan a menudo en la capa de enlace. Una pregunta bastante natural llegados a este punto sería: ¿por qué se utilizan sumas de comprobación en la capa de transporte y comprobaciones de redundancia cíclica en la capa de enlace? Recuerde que normalmente la capa de transporte se implementa por software en un host, como parte del sistema operativo del mismo. Puesto que el mecanismo de detección de errores de la capa de transporte se implementa por software, es importante utilizar un esquema de detección de errores simple y rápido, como por ejemplo el de las sumas de comprobación. Por el contrario, la detección de errores en la capa de enlace se implementa en un hardware dedicado dentro de las tarjetas adaptadoras, pudiendo dicho hardware realizar rápidamente las operaciones CRC más complejas. Feldmeier [Feldmeier 1995] presenta una serie de técnicas de implementación rápida en software no solo para códigos de suma de comprobación ponderados, sino también para códigos CRC (véase más adelante) y otros códigos.

6.2.3 Comprobación de redundancia cíclica (CRC)

Una técnica de detección de errores utilizada ampliamente en las redes de computadoras de hoy día está basada en los **códigos de comprobación de redundancia cíclica (CRC, Cyclic Redundancy Check)**. Los códigos CRC también se conocen con el nombre de **códigos polinómicos**, dado que se puede ver la cadena de bits que hay que enviar como si fuera un polinomio cuyos coeficientes son los valores 0 y 1 de la cadena de bits, interpretándose las operaciones realizadas con la cadena de bits según la aritmética de polinomios.

Los códigos CRC operan de la forma siguiente. Considere la secuencia de datos de d bits, D , que el nodo emisor quiere transmitir al nodo receptor. El emisor y el receptor tienen que acordar primero un patrón de $r + 1$ bits, conocido como **generador**, que denominaremos con la letra G . Impondremos la condición de que el bit más significativo (el bit situado más a la izquierda) de G sea 1. La idea clave subyacente a los códigos CRC se muestra en la Figura 6.6. Para una determinada secuencia de datos, D , el emisor seleccionará r bits adicionales, R , y se los añadirá a D , de modo que el patrón de $d + r$ bits resultante (interpretado como un número binario) sea exactamente divisible por G (es decir, no tenga ningún resto) utilizando aritmética módulo 2. El proceso de comprobación de errores con los códigos CRC es, por tanto, muy simple: el receptor divide los $d + r$ bits recibidos entre G . Si el resto es distinto de cero, el receptor sabrá que se ha producido algún error; en caso contrario, se aceptarán los datos como correctos.

Todos los cálculos de los códigos CRC se realizan en aritmética módulo 2, sin ningún tipo de acarreo ni en las sumas ni en las restas. Esto quiere decir que la suma y la resta son idénticas, y que ambas son equivalentes a la operación OR-exclusiva (XOR) bit a bit de los operandos. Así, por ejemplo,

$$\begin{aligned} 1011 \text{ XOR } 0101 &= 1110 \\ 1001 \text{ XOR } 1101 &= 0100 \end{aligned}$$

De forma similar, también tendremos que

$$\begin{aligned} 1011 - 0101 &= 1110 \\ 1001 - 1101 &= 0100 \end{aligned}$$

La multiplicación y la división son iguales que en aritmética en base 2, excepto porque las sumas y restas necesarias se llevan a cabo sin acarreos. Como en la aritmética binaria ordinaria, la multiplicación por 2^k hace que un patrón de bits se desplace k posiciones hacia la izquierda. Por tanto, dados D y R , el valor $D \cdot 2^r$ XOR R nos dará el patrón de $d + r$ bits mostrado en la Figura 6.6. Utilizaremos esta caracterización algebraica del patrón de $d + r$ bits de la Figura 6.6 en nuestras explicaciones.

Volvamos ahora a la pregunta crucial de cómo puede el emisor calcular R . Recuerde que queremos encontrar una secuencia R tal que exista n que cumpla

$$D \cdot 2^r \text{ XOR } R = nG$$

Es decir, queremos seleccionar R tal que G divida a $D \cdot 2^r$ XOR R sin que quede resto. Si aplicamos la operación XOR (es decir, si sumamos en módulo 2 sin acarreo) con R a ambos lados de la ecuación anterior, obtenemos

$$D \cdot 2^r = nG \text{ XOR } R$$

Esta ecuación nos dice que si dividimos $D \cdot 2^r$ entre G , el valor del resto será precisamente R . En otras palabras, podemos calcular R como

$$R = \text{resto} \frac{D \cdot 2^r}{G}$$

La Figura 6.7 ilustra este cálculo para el caso de $D = 101110$, $d = 6$, $G = 1001$ y $r = 3$. Los 9 bits transmitidos en este caso serán 101110 011. Puede comprobar estos cálculos por sí mismo y comprobar también que, efectivamente, $D \cdot 2^r = 101011 \cdot G$ XOR R .

Se han definido estándares internacionales para generadores G de 8, 12, 16 y 32 bits. El estándar CRC-32 para 32 bits, que se ha adoptado en una serie de protocolos del IEEE para la capa de enlace, utiliza el generador

$$G_{\text{CRC-32}} = 100000100110000010001110110110111$$

Cada uno de los estándares de CRC puede detectar ráfagas de errores inferiores a $r + 1$ bits (esto significa que todos los errores de r bits consecutivos o menos serán detectados). Además, en las condiciones adecuadas, una ráfaga de longitud superior a $r + 1$ bits será detectada con una probabilidad de $1 - 0,5^r$. Asimismo, cada uno de los estándares CRC puede detectar cualquier número impar de errores de bit. En [Williams 1993] puede encontrar un análisis de la implementación de códigos CRC. La teoría subyacente a los códigos CRC y a algunos códigos incluso más potentes cae fuera del alcance de este libro. El texto de [Schwartz 1980] proporciona una excelente introducción a este tema.

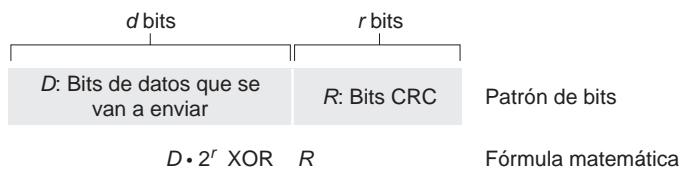


Figura 6.6 ♦ CRC.

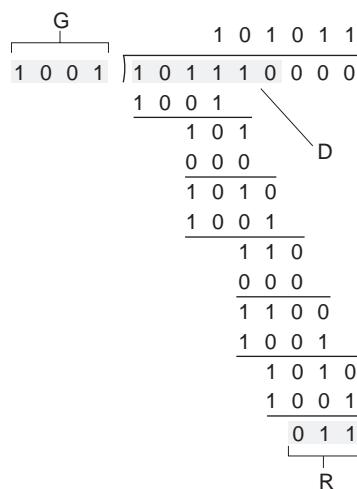


Figura 6.7 ♦ Un ejemplo de cálculo de CRC.

6.3 Protocolos y enlaces de acceso múltiple

En la introducción de este capítulo hemos indicado que existen dos tipos de enlaces de red: enlaces punto a punto y enlaces de difusión (*broadcast*). Un **enlace punto a punto** está compuesto por un único emisor en un extremo del enlace y un único receptor en el otro extremo. Se han diseñado muchos protocolos de la capa de enlace para enlaces punto a punto; dos de esos protocolos son el Protocolo punto a punto (PPP, *Point-to-Point Protocol*) y el protocolo de Control del enlace de datos de alto nivel (HDLC, *High-level Data Link Control*). El segundo tipo de enlace, un **enlace de difusión**, puede tener múltiples nodos emisores y receptores, todos conectados al mismo y único canal de difusión compartido. Utilizamos aquí el término *difusión* porque cuando un nodo transmite una trama, el canal se encarga de difundir esa trama y cada uno de los demás nodos recibe una copia. Ethernet y las redes LAN inalámbricas son ejemplos de tecnologías de difusión de la capa de enlace. En esta sección vamos a abstraernos momentáneamente de los protocolos específicos de la capa de enlace y vamos a examinar en primer lugar un problema de crucial importancia para esa capa: cómo coordinar el acceso de múltiples nodos emisores y receptores a un canal de difusión compartido, lo que se conoce con el nombre de **problema de acceso múltiple**. Los canales de difusión se suelen utilizar en las redes LAN, que son redes geográficamente concentradas en un único edificio (o un campus corporativo o universitario). Por ello, también examinaremos al final de esta sección cómo se utilizan los canales de acceso múltiple en las redes LAN.

Todos estamos familiarizados con la noción de transmisiones de difusión, ya que la televisión ha estado empleando este tipo de mecanismo desde que fuera inventada. Pero la televisión tradicional es una difusión en un único sentido (es decir, hay un nodo fijo que transmite a muchos nodos receptores), mientras que los nodos de un canal de difusión de una red de computadoras pueden tanto enviar como recibir. Quizá una analogía más adecuada para un canal de difusión, extraída del campo de las relaciones sociales, sería un cóctel en el que muchas personas se reúnen en una gran sala (el aire proporciona el medio de difusión) para hablar y escuchar. Una segunda analogía sería algo con lo que muchos lectores estarán familiarizados: una clase, en la que uno o más profesores y estudiantes comparten de forma similar el mismo y único medio de difusión. Un problema crucial en ambos escenarios es el de determinar quién es el que tiene derecho a la palabra (es decir, derecho a transmitir hacia el canal) y cuándo lo tiene. Las personas hemos llegado a desarrollar un conjunto elaborado de protocolos con el fin de compartir el canal de difusión:

- “Dar a todo el mundo una oportunidad de hablar.”
- “No hablar hasta que te hablen.”
- “No monopolizar la conversación.”
- “Levantar la mano si se tiene una pregunta que plantear.”
- “No interrumpir cuando alguien está hablando.”
- “No quedarse dormido cuando alguien está hablando.”

De forma similar, las redes de computadoras tienen protocolos (denominados **protocolos de acceso múltiple**) mediante los cuales los nodos se encargan de regular sus transmisiones al canal de difusión compartido. Como se muestra en la Figura 6.8, los protocolos de acceso múltiple son necesarios en una amplia variedad de escenarios de red, incluyendo las redes de acceso tanto cableadas como inalámbricas y las redes de satélite. Aunque técnicamente cada nodo accede al canal de difusión a través de su adaptador, en esta sección nos referiremos con el término *nodo* a los dispositivos emisor y receptor. En la práctica, puede haber cientos o incluso miles de nodos comunicándose directamente a través de un canal de difusión.

Puesto que todos los nodos son capaces de transmitir tramas, podría darse el caso de que más de dos nodos transmitieran tramas al mismo tiempo. Cuando esto sucede, todos los nodos reciben varias tramas simultáneamente; es decir, las tramas transmitidas **colisionan** en todos los receptores. Normalmente, cuando se produce una colisión ninguno de los nodos receptores puede interpretar ninguna de las tramas transmitidas; en un cierto sentido, las señales de las tramas que han colisionado se entremezclan y no pueden separarse. Por tanto, todas las tramas implicadas en la colisión se pierden y el canal de difusión está desaprovechado durante el intervalo de colisión. Obviamente, si hay muchos nodos que quieren transmitir tramas de manera frecuente, muchas de las transmisiones provocarán colisiones y buena parte del ancho de banda del canal de difusión se desperdiciará.

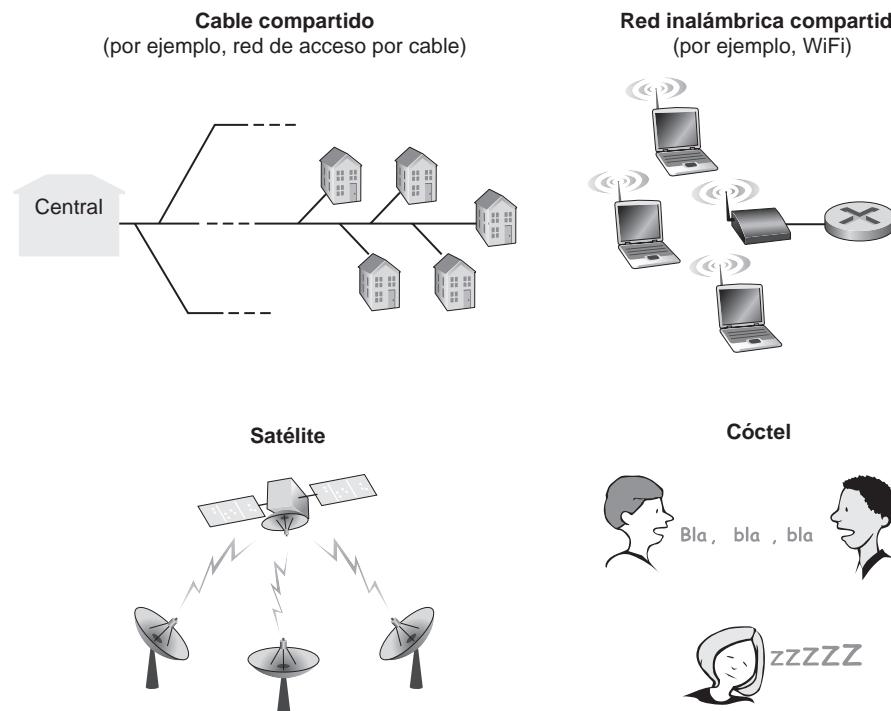


Figura 6.8 ♦ Varios canales de acceso múltiple.

Para poder garantizar que el canal de difusión realice un trabajo útil aun cuando haya múltiples nodos activos, es necesario coordinar de alguna manera las transmisiones de esos nodos activos. Este trabajo de coordinación es responsabilidad del protocolo de acceso múltiple. En los últimos 40 años se han escrito miles de artículos y cientos de tesis doctorales acerca de los protocolos de acceso múltiple; puede encontrar en [Rom 1990] una panorámica bastante completa de los primeros 20 años de estas investigaciones. Hoy día, la investigación acerca de los protocolos de acceso múltiple continúa de manera activa debido a la continua aparición de nuevos tipos de enlaces y, en particular, de nuevos enlaces inalámbricos.

A lo largo de los años se han implementado docenas de protocolos de acceso múltiple utilizando diversas tecnologías de la capa de enlace. No obstante, podemos clasificar casi todos los protocolos de acceso múltiple en una de estas tres categorías: **protocolos de particionamiento del canal**, **protocolos de acceso aleatorio** y **protocolos de toma de turnos**. Hablaremos de estas tres categorías de protocolos de acceso múltiple en las tres siguientes subsecciones.

Vamos a concluir esta introducción observando que, idealmente, un protocolo de acceso múltiple para un canal de difusión con una velocidad de R bits por segundo debería tener las siguientes características deseables:

1. Cuando solo haya un nodo que tenga datos para enviar, a dicho nodo se le asignará una tasa de transferencia de R bps.
2. Cuando haya M nodos con datos para enviar, cada uno de esos nodos tendrá una tasa de transferencia de R/M bps. Esto no implica necesariamente que cada uno de los M nodos tenga siempre una tasa instantánea igual a R/M , sino más bien que cada nodo tendrá una tasa media de transmisión igual a R/M a lo largo de un intervalo de tiempo definido adecuadamente.
3. El protocolo será descentralizado; es decir, no habrá ningún nodo maestro que pueda actuar como punto único de fallo para la red.
4. El protocolo será simple, de modo que no sea costoso de implementar.

6.3.1 Protocolos de particionamiento del canal

Recuerde de nuestras explicaciones de la Sección 1.3 que la multiplexación por división en el tiempo (TDM) y la multiplexación por división de frecuencia (FDM) son dos técnicas que se pueden utilizar para particionar el ancho de banda de un canal de difusión entre todos los nodos que comparten el canal. Por ejemplo, suponga que el canal da soporte a N nodos y que la tasa de transmisión del canal es igual a R bps. TDM divide el tiempo en **marcos temporales** y luego subdivide cada marco temporal en N **particiones de tiempo**. (En inglés, el término *time frame*, marco de tiempo, de TDM no debe confundirse con la unidad de datos de la capa de enlace intercambiada entre las tarjetas adaptadoras de red del emisor y del receptor, que se denomina también *frame*, y que nosotros hemos denominado trama. En cualquier caso, para evitar confusiones en esta subsección nos referiremos a la unidad de datos intercambiada en la capa de enlace como paquete.) Cada partición de tiempo se asigna entonces a uno de los N nodos. Cada vez que un nodo tenga un paquete para enviar, transmite los bits del paquete durante su partición de tiempo asignada, dentro del marco TDM que se repite de forma cíclica. Normalmente, los tamaños de partición se eligen de modo que pueda transmitirse un único paquete durante la partición de tiempo asignada. La Figura 6.9 muestra un ejemplo simple de TDM con cuatro nodos. Si volvemos a nuestra analogía del cóctel, un cóctel regulado mediante TDM permitiría a uno de los participantes hablar durante un periodo fijo de tiempo, luego permitiría a otro participante hablar durante la misma cantidad de tiempo, y así sucesivamente. Una vez que todos hubieran tenido la oportunidad de hablar, el patrón se repetiría.

TDM resulta muy atractivo porque elimina las colisiones y es perfectamente equitativo: cada nodo obtiene una tasa de transmisión dedicada igual a R/N bps durante cada marco temporal. Sin embargo, presenta dos importantes inconvenientes. En primer lugar, cada nodo está limitado a una

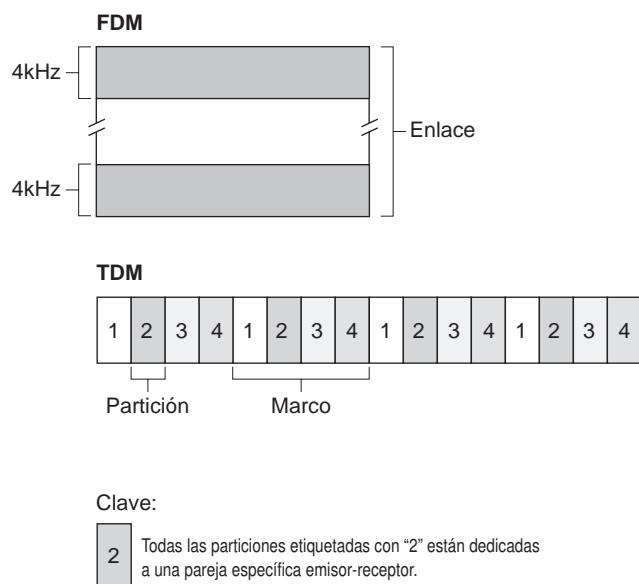


Figura 6.9 ♦ Un ejemplo de TDM y FDM para cuatro nodos.

tasa promedio de R/N bps aunque sea el único nodo que tiene paquetes para transmitir. El segundo inconveniente es que un nodo siempre tiene que esperar a que le llegue el turno dentro de la secuencia de transmisión; de nuevo, esa espera será obligatoria aún cuando sea el único nodo que tenga una trama que enviar. Imagine que esa situación se diera con un asistente al cóctel que es el único que tiene algo que decir (imagine que nos encontráramos con la circunstancia, todavía más rara, de que todo el mundo quiere escuchar lo que esa persona tiene que decir). Evidentemente, TDM sería un protocolo de acceso múltiple bastante inadecuado para ese cóctel en concreto.

Mientras que TDM hace que los nodos comparten el canal de difusión a lo largo del tiempo, FDM divide el canal de R bps en diferentes frecuencias (cada una con un ancho de banda de R/N) y asigna cada frecuencia a cada uno de los N nodos. Así, FDM crea N canales más pequeños de R/N bps a partir de un único canal disponible mayor, de R bps. FDM comparte con TDM tanto las ventajas como los inconvenientes. Evita las colisiones y divide el ancho de banda equitativamente entre los N nodos. Sin embargo, FDM también comparte una desventaja fundamental con TDM: cada nodo está limitado a un ancho de banda de R/N , incluso cuando sea el único nodo que tiene paquetes para enviar.

Un tercer protocolo de particionamiento del canal es el protocolo **CDMA (Code Division Multiple Access, Acceso múltiple por división de código)**. Mientras que TDM y FDM asignan particiones de tiempo y frecuencia, respectivamente, a los nodos, CDMA asigna un *código* diferente a cada nodo. Cada nodo utiliza entonces su código exclusivo para codificar los bits de datos a enviar. Si se seleccionan los códigos cuidadosamente, las redes CDMA presentan la maravillosa característica de que los distintos nodos pueden transmitir *simultáneamente* y conseguir que sus respectivos receptores decodifiquen correctamente los bits de datos codificados por el emisor (suponiendo que el receptor conozca el código utilizado por el emisor) aunque haya interferencias provocadas por las transmisiones realizadas por los otros nodos. CDMA se ha utilizado en sistemas militares durante algún tiempo (debido a su resistencia a las interferencias) y ahora se usa ampliamente en el mundo civil, en particular en la telefonía celular. Puesto que el uso de CDMA está tan estrechamente ligado a los canales inalámbricos, dejaremos las explicaciones acerca de los detalles técnicos de CDMA para el Capítulo 7. Por el momento, nos basta con saber que se pueden asignar códigos CDMA, al igual que particiones de tiempo en TDM y frecuencias en FDM, a los usuarios del canal de acceso múltiple.

6.3.2 Protocolos de acceso aleatorio

La segunda clase general de protocolos de acceso múltiple son los protocolos de acceso aleatorio. En un protocolo de acceso aleatorio, cada nodo transmisor transmite siempre a la máxima velocidad del canal, que es R bps. Cuando se produce una colisión, cada uno de los nodos implicados en la colisión retransmite repetidamente su trama (es decir, su paquete) hasta que la trama consiga pasar sin sufrir colisiones. Pero cuando un nodo experimenta una colisión no retransmite necesariamente la trama de forma inmediata. *En lugar de ello, espera durante un tiempo aleatorio antes de retransmitir la trama.* Cada nodo implicado en una colisión selecciona un retardo aleatorio independientemente. Puesto que los retardos aleatorios son elegidos de forma independiente, es posible que uno de los nodos seleccione un retardo que sea suficientemente menor que los retardos de los otros nodos que han intervenido en la colisión, pudiendo así ser capaz de conseguir que su trama pase a través del canal sin experimentar una nueva colisión.

Existen docenas, si no centenares, de protocolos de acceso aleatorio descritos en la literatura científica [Rom 1990; Bertsekas 1991]. En esta sección describiremos unos pocos de los protocolos de acceso aleatorio más comúnmente utilizados: los protocolos ALOHA [Abramson 1970; Abramson 1985; Abramson 2009] y los protocolos de Acceso múltiple con sondeo de portadora (CSMA, *Carrier Sense Multiple Access*) [Kleinrock 1975b]. Ethernet [Metcalfe 1976] es un protocolo CSMA muy popular y ampliamente difundido.

ALOHA con particiones

Comencemos nuestro estudio de los protocolos de acceso aleatorio con uno de los protocolos de este tipo más simples: el protocolo ALOHA con particiones. En nuestra descripción de ALOHA con particiones haremos las siguientes suposiciones:

- Todas las tramas constan de exactamente L bits.
- El tiempo está dividido en particiones de L/R segundos (es decir, cada partición equivale al tiempo que se tarda en transmitir una trama).
- Los nodos comienzan a transmitir las tramas solo al principio de las particiones.
- Los nodos están sincronizados, de modo que cada nodo sabe cuándo comienzan las particiones.
- Si dos o más tramas colisionan en una partición, entonces todos los nodos detectan la colisión incluso antes de que la partición termine.

Sea p una probabilidad, es decir, un número comprendido entre 0 y 1. El funcionamiento del protocolo ALOHA con particiones en cada nodo es simple:

- Cuando el nodo tiene una nueva trama que enviar, espera hasta el comienzo de la siguiente partición y transmite la trama completa dentro de la partición.
- Si no se produce una colisión, el nodo habrá transmitido correctamente su trama y por tanto no considerará la posibilidad de retransmitirla (el nodo puede preparar una nueva trama para su transmisión, si tiene una disponible).
- Si se produce una colisión, el nodo detecta la colisión antes de que la partición termine. El nodo retransmitirá su trama en cada partición posterior con una probabilidad p , hasta conseguir que la trama sea transmitida sin experimentar colisiones.

Al decir que se retransmite con probabilidad p , queremos decir que el nodo lleva a cabo en la práctica una especie de lanzamiento de una moneda trucada: si sale cara, la trama se retransmite, lo cual sucede con probabilidad p ; si sale cruz, se deja pasar la partición y se vuelve a lanzar la moneda para la partición siguiente, lo que ocurre con probabilidad $(1 - p)$. Todos los nodos implicados en la colisión “arrojan sus monedas” de forma independiente.

Puede parecer que el protocolo ALOHA con particiones tiene muchas ventajas. A diferencia de los mecanismos de particionamiento del canal, ALOHA con particiones permite a un nodo transmitir continuamente a la velocidad máxima, R , cuando dicho nodo sea el único activo (decimos que un nodo está activo si tiene tramas que transmitir). ALOHA con particiones también es un protocolo altamente descentralizado, porque cada nodo detecta las colisiones y decide de forma independiente cuándo debe retransmitir (sin embargo, ALOHA con particiones requiere que las particiones estén sincronizadas en los nodos; en breve analizaremos una versión no particionada del protocolo ALOHA, así como los protocolos CSMA, ninguno de los cuales requiere dicho tipo de sincronización). ALOHA con particiones es también un protocolo extremadamente simple.

ALOHA con particiones funciona bien cuando solo hay un nodo activo, pero ¿qué eficiencia tiene cuando existen múltiples nodos activos? Son dos las posibles preocupaciones en lo que respecta a la eficiencia. En primer lugar, como se muestra en la Figura 6.10, cuando hay múltiples nodos activos una cierta fracción de las particiones experimentará colisiones y por tanto se “desperdiciará”. La segunda preocupación es que otra fracción de las particiones estará *vacía* en aquellos casos en que todos los nodos activos se abstengan de transmitir, como resultado de la política probabilística de retransmisión. Las únicas particiones “no desperdiciadas” serán aquellas para las que haya exactamente un nodo transmitiendo. A las particiones en las que hay exactamente un nodo transmitiendo se las denomina **particiones con éxito**. La **eficiencia** de un protocolo de acceso múltiple con particiones se define como la fracción, calculada a largo plazo, de particiones con éxito cuando existe un gran número de nodos activos, cada uno de los cuales tiene siempre una gran cantidad de tramas que enviar. Observe que si no se utilizara ningún tipo de control de acceso y cada nodo intentara retransmitir inmediatamente después de cada colisión, la eficiencia sería cero. El protocolo ALOHA con particiones hace obviamente que la eficiencia aumente por encima de cero pero, ¿cuánto aumenta?

Vamos a esbozar el modo de determinar la eficiencia máxima del protocolo ALOHA con particiones. Para simplificar las cosas, vamos a modificar el protocolo ligeramente y a suponer que cada nodo trata de transmitir una trama en cada partición con probabilidad p . (Es decir, suponemos que todos los nodos tienen siempre una trama que enviar y que el nodo transmite con probabilidad p para las nuevas tramas, y no solo para las tramas que ya hayan sufrido una colisión.) Suponga que existen N nodos. Entonces, la probabilidad de que una partición dada sea una partición con éxito es la probabilidad de que uno de los nodos transmita y de que los restantes $N - 1$ nodos no

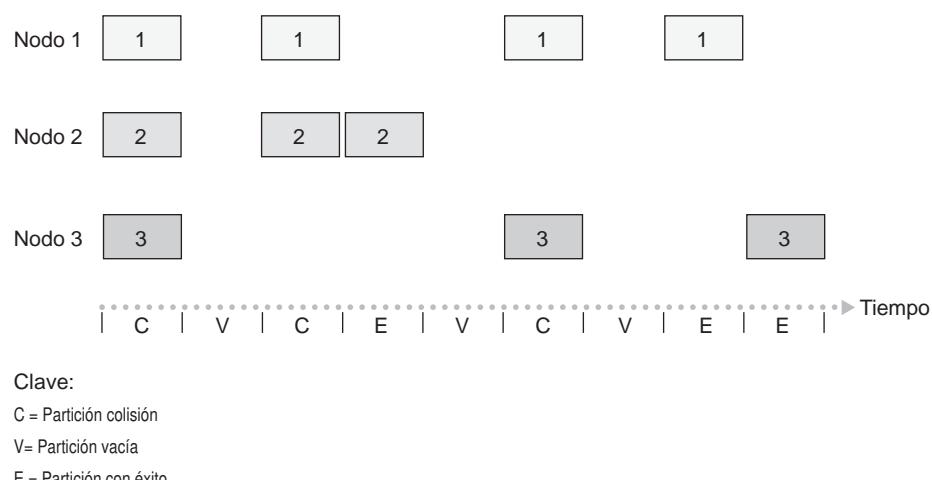


Figura 6.10 ♦ Los nodos 1, 2 y 3 colisionan en la primera partición. El nodo 2 consigue tener éxito finalmente en la cuarta partición, el nodo 1 en la octava partición y el nodo 3 en la novena.

transmitan. La probabilidad de que un cierto nodo transmita es p ; la probabilidad de que los demás nodos no transmitan es $(1 - p)^{N-1}$. Por tanto, la probabilidad de que un cierto nodo tenga éxito al transmitir será $p(1 - p)^{N-1}$. Puesto que hay N nodos, la probabilidad de que exactamente uno de los N nodos tenga éxito es $Np(1 - p)^{N-1}$.

Por tanto, cuando hay N nodos activos, la eficiencia del protocolo ALOHA con particiones es $Np(1 - p)^{N-1}$. Para obtener la eficiencia *máxima* para N nodos activos, tenemos que determinar el valor p^* que maximice esta expresión. (Consulte los problemas de repaso para ver un esbozo de estos cálculos.) Y para obtener la máxima eficiencia para un gran número de nodos activos, tomaremos el límite de $Np^*(1 - p^*)^{N-1}$ cuando N tiende a infinito. (Consulte de nuevo los problemas de repaso.) Despues de realizar estos cálculos, se puede comprobar que la eficiencia máxima del protocolo está dada por $1/e = 0,37$. Es decir, cuando un gran número de nodos tienen muchas tramas que transmitir, entonces (como máximo) solo el 37 por ciento de las particiones conseguirán transmitir la información con éxito. Por tanto, ¡la velocidad de transmisión efectiva del canal no es R bps sino solo $0,37 R$ bps! Un análisis similar muestra también que el 37 por ciento de las particiones quedarán vacías y el 26 por ciento sufrirán colisiones. ¡Imagíñese al pobre administrador de la red que acaba de adquirir un sistema ALOHA con particiones a 100 Mbps y que esperaba poder utilizar la red para transmitir datos entre una gran cantidad de usuarios, con una velocidad agregada de, digamos, en torno a 80 Mbps! Aunque el canal es capaz de transmitir una determinada trama a la velocidad máxima del canal de 100 Mbps, a largo plazo, la tasa de transferencia efectiva de este canal será inferior a 37 Mbps.

ALOHA

El protocolo ALOHA con particiones requiere que todos los nodos sincronicen sus transmisiones para que estas comiencen al principio de una partición. El primer protocolo ALOHA [Abramson 1970] era en realidad un protocolo no particionado y completamente descentralizado. En el protocolo ALOHA puro, cuando llega una trama (es decir, cuando se pasa un datagrama desde la capa de red en el nodo emisor) el nodo transmite inmediatamente la trama en su totalidad hacia el canal de difusión. Si una trama transmitida experimenta una colisión con una o más transmisiones de otros nodos, el nodo (después de transmitir completamente la trama que ha sufrido la colisión) retransmitirá la trama de forma inmediata con una probabilidad p . En caso contrario, el nodo esperará durante un tiempo equivalente al tiempo total de transmisión de una trama. Después de esta espera, transmitirá la trama con probabilidad p , o esperará (permaneciendo inactivo) durante otro periodo de tiempo igual al tiempo de transmisión de una trama con una probabilidad $1 - p$.

Para determinar la eficiencia máxima del protocolo ALOHA puro vamos a centrarnos en un nodo individual. Haremos las mismas suposiciones que en nuestro análisis del protocolo ALOHA con particiones y tomaremos como unidad de tiempo el tiempo de transmisión de una trama. En cualquier instante, la probabilidad de que un nodo esté transmitiendo una trama será p . Suponga que esta trama comienza su transmisión en el instante t_0 . Como se muestra en la Figura 6.11, para que esta trama pueda transmitirse con éxito ningún otro nodo puede iniciar su transmisión en el intervalo de tiempo $[t_0 - 1, t_0]$, ya que dicha transmisión se solaparía con el inicio de la transmisión de la trama del nodo i . La probabilidad de que todos los demás nodos no comiencen una transmisión en ese intervalo es $(1 - p)^{N-1}$. De forma similar, ningún otro nodo puede comenzar una transmisión mientras el nodo i está transmitiendo, ya que dicha transmisión se solaparía con la última parte de la transmisión del nodo i . La probabilidad de que todos los demás nodos no inicien una transmisión en este intervalo será también $(1 - p)^{N-1}$. Por tanto, la probabilidad de que un cierto nodo pueda transmitir con éxito será $p(1 - p)^{2(N-1)}$. Tomando límites como en el caso del protocolo ALOHA con particiones, encontramos que la máxima eficiencia del protocolo ALOHA puro es solo de $1/(2e)$, que es exactamente la mitad que la del ALOHA con particiones. Por tanto, ese será el precio que habrá que pagar por disponer de un protocolo ALOHA completamente descentralizado.

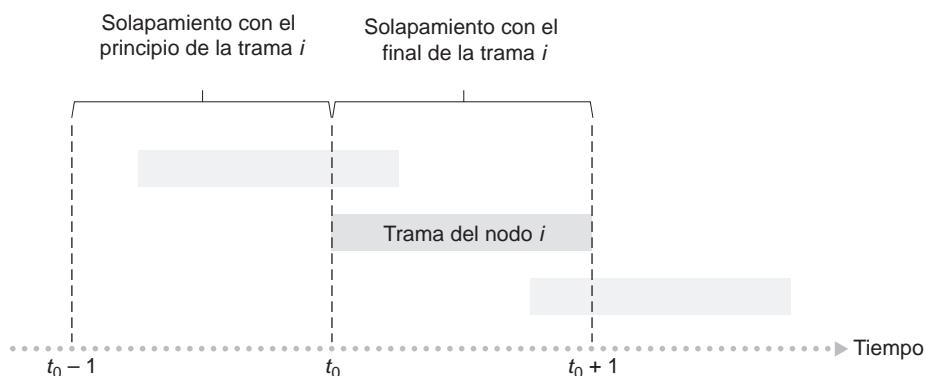


Figura 6.11 ♦ Transmisiones que interfieren en el protocolo ALOHA puro.

Acceso múltiple con sondeo de portadora (CSMA)

Tanto en el protocolo ALOHA puro como con particiones, la decisión de transmitir por parte de un nodo se toma independientemente de la actividad de los otros nodos conectados al canal de difusión. En particular, los nodos nunca prestan atención, en el momento de comenzar a transmitir, a si hay otros nodos transmitiendo, ni tampoco dejan de transmitir si otro nodo comienza a interferir con su transmisión. En nuestra analogía del cóctel, los protocolos ALOHA se parecen bastante a uno de esos invitados maleducados que continúan charlando independientemente de si hay otras personas haciendo uso de la palabra. Las personas disponemos de protocolos que nos permiten no solo comportarnos de manera civilizada, sino también reducir la cantidad de tiempo desperdiciado “colisionando” unos con otros durante las conversaciones e incrementar así, como consecuencia, la cantidad de información que en nuestras conversaciones podemos intercambiar. Específicamente, hay dos reglas importantes de buena educación en las conversaciones que mantenemos los seres humanos:

HISTORIA

NORM ABRAMSON Y ALOHANET

Norm Abramson, doctor ingeniero, era un apasionado del surf y estaba también interesado en el tema de la conmutación de paquetes. Esa combinación de intereses le llevó a la universidad de Hawaii en 1969. Hawaii está compuesto por muchas islas montañosas, lo que hace difícil instalar y operar redes terrestres. Cuando no estaba practicando surf, Abramson se dedicaba a pensar en cómo diseñar una red que realizará la conmutación de paquetes vía radio. La red que diseñó disponía de un host central y de varios nodos secundarios dispersos por las islas del archipiélago de Hawaii. La red tenía dos canales, cada uno de los cuales utilizaba una banda de frecuencia distinta. El canal de bajada difundía los paquetes desde el host central hacia los hosts secundarios, mientras que el de subida permitía enviar paquetes desde los hosts secundarios al host central. Además de enviar paquetes de información, el host central también enviaba a través del canal de bajada un mensaje de reconocimiento para cada uno de los paquetes recibidos con éxito desde los hosts secundarios.

Puesto que los hosts secundarios transmitían los paquetes de forma descentralizada, las colisiones en el canal de subida eran inevitables. Esta observación condujo a Abramson a desarrollar el protocolo ALOHA puro, descrito en este capítulo. En 1970, gracias a las aportaciones económicas de ARPA, Abramson conectó su red ALOHAnet con ARPAnet. El trabajo de Abramson es importante no solo porque fue el primer ejemplo de red de paquetes vía radio, sino también porque sirvió de inspiración a Bob Metcalfe. Unos pocos años más tarde, Metcalfe modificó el protocolo ALOHA para crear el protocolo CSMA/CD y las redes LAN Ethernet.

- *Escuchar antes de hablar.* Si hay otra persona hablando, esperaremos hasta que haya terminado. En el mundo de las redes esto se denomina **sondeo de portadora**: cada nodo escucha el canal antes de transmitir. Si actualmente se está transmitiendo una trama de otro nodo por el canal, el nodo esperará un intervalo de tiempo corto hasta detectar que no hay ninguna transmisión y luego volverá a iniciar la transmisión.
- *Si alguien comienza a hablar al mismo tiempo, hay que dejar de hablar.* En el mundo de las redes esto se denomina **detección de colisiones**: un nodo que esté transmitiendo escuchará qué es lo que hay en el canal mientras dure la transmisión. Si detecta que otro nodo está transmitiendo una trama que interfiere con la suya, dejará de transmitir y esperará una cantidad aleatoria de tiempo antes de repetir el ciclo de detectar y transmitir si no hay actividad.

Estas dos reglas están integradas en la familia de protocolos de **Acceso múltiple con sondeo de portadora (CSMA, Carrier Sense Multiple Access)** y **CSMA con detección de colisiones (CSMA/CD)** [Kleinrock 1975b; Metcalfe 1976; Lam 1980; Rom 1990]. Se han propuesto muchas variantes de CSMA y CSMA/CD. Aquí consideraremos algunas de las características más importantes y fundamentales de CSMA y CSMA/CD.

La primera cuestión que podríamos plantearnos acerca de CSMA es por qué, si todos los nodos llevan a cabo un sondeo de portadora, se producen colisiones. Después de todo, los nodos se guardarán de transmitir cada vez que detecten que otro nodo está transmitiendo. La mejor forma de responder a esta pregunta es ilustrarla mediante diagramas espacio-tiempo [Molle 1987]. La Figura 6.12 muestra un diagrama espacio-tiempo de cuatro nodos (A, B, C, D) conectados a un bus lineal de difusión. El eje horizontal muestra la posición de cada nodo en el espacio y el eje vertical representa el tiempo.

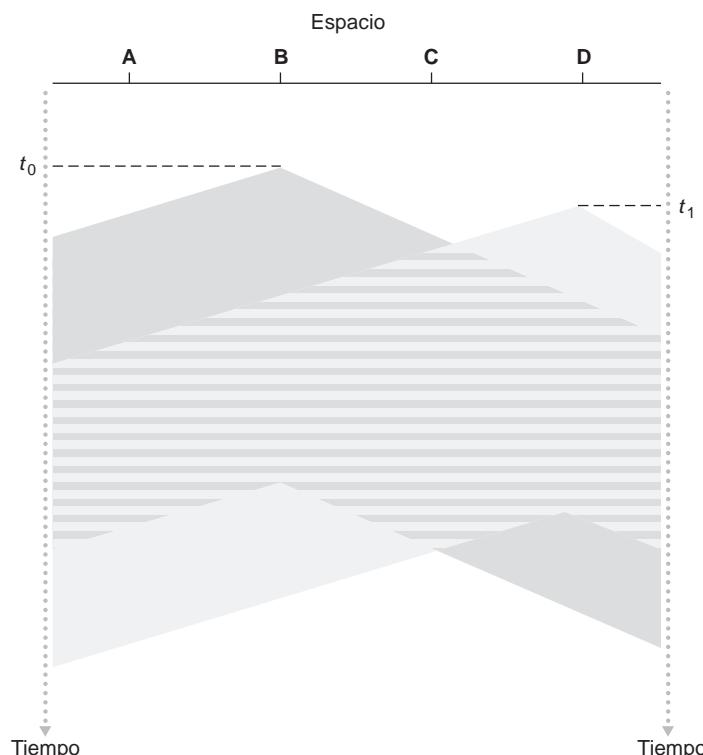


Figura 6.12 ♦ Diagrama espacio-tiempo para dos nodos CSMA con transmisiones que entran en colisión.

En el instante t_0 el nodo B comprueba que el canal está inactivo, ya que no hay ningún nodo transmitiendo actualmente. Por tanto, el nodo B comenzará a transmitir propagándose sus bits en ambas direcciones a lo largo del medio de difusión. La propagación hacia abajo de los bits de B en la Figura 6.12 a lo largo del tiempo indica que hace falta un intervalo de tiempo distinto de cero para que los bits de B consigan propagarse (aunque lo hagan a una velocidad próxima a la de la luz) a lo largo de medio de difusión. En el instante t_1 ($t_1 > t_0$) el nodo D tiene una trama que enviar. Aunque el nodo B actualmente está transmitiendo en el instante t_1 , los bits que B está transmitiendo todavía no han alcanzado a D, por lo que D detectará que el canal está inactivo en t_1 . De acuerdo con el protocolo CSMA, D por tanto comienza a transmitir su trama. Un corto intervalo de tiempo después, la transmisión de B comienza a interferir en D con la propia transmisión de D. A partir de la Figura 6.12 resulta evidente que el **retardo de propagación de canal** extremo a extremo de un canal de difusión (el tiempo que una señal tarda en propagarse de uno de los nodos a otro) desempeñará un papel fundamental a la hora de determinar el rendimiento del canal. Cuanto mayor sea este retardo de propagación, mayor será la probabilidad de que un nodo que efectúa el sondeo de portadora no sea capaz de detectar una transmisión que ya ha comenzado en otro nodo de la red.

CSMA con detección de colisiones (CSMA/CD)

En la Figura 6.12 los nodos no realizan una detección de colisiones; tanto B como D continúan transmitiendo sus tramas en su totalidad, aún cuando se haya producido una colisión. Cuando un nodo realiza la detección de colisiones, deja de transmitir en cuanto detecta que se ha producido una colisión. La Figura 6.13 muestra el mismo escenario que la Figura 6.12, salvo porque los dos nodos abortan ahora su transmisión poco después de detectar que se ha producido una colisión. Evidentemente, añadir el mecanismo de detección de colisiones a un protocolo de acceso múltiple ayudará a incrementar el rendimiento del protocolo al no transmitirse una trama inútil, dañada (por interferencia con una trama de otro nodo) en su totalidad.

Antes de analizar el protocolo CSMA/CD, vamos a resumir su funcionamiento desde la perspectiva de un adaptador (en un nodo) conectado a un canal de difusión:

1. El adaptador obtiene un datagrama de la capa de red, prepara una trama de la capa de enlace y la coloca en un buffer del adaptador.
2. Si el adaptador detecta que el canal está inactivo (es decir, el adaptador no recibe intensidad de señal procedente del canal), comienza a transmitir la trama. Si, por el contrario, el adaptador detecta que el canal está ocupado, espera hasta comprobar que no hay intensidad de señal y luego comienza a transmitir la trama.
3. Mientras está transmitiendo, el adaptador monitoriza la presencia de señales procedentes de otros adaptadores que empleen el canal de difusión.
4. Si el adaptador transmite la trama completa sin detectar ninguna señal procedente de otros adaptadores, concluye que ha terminado su trabajo con esa trama. Si, por el contrario, el adaptador detecta intensidad de señal procedente de otros adaptadores mientras está transmitiendo, cancela la transmisión (es decir, deja de transmitir su trama).
5. Despues de abortar la transmisión de la trama, el adaptador espera una cantidad de tiempo aleatoria y vuelve al paso 2.

Suponemos que la necesidad de esperar un tiempo aleatorio (en lugar de uno fijo) estará clara para el lector: si dos nodos transmitieran tramas al mismo tiempo y luego ambos esperaran un mismo tiempo fijo, continuarían colisionando eternamente. ¿Pero cuál sería un intervalo de tiempo adecuado para elegir el tiempo de *backoff* (espera) aleatorio? Si el intervalo es grande y el número de nodos que colisionan es pequeño, es probable que los nodos tengan que esperar una gran cantidad de tiempo (durante el cual el canal permanecerá inactivo) antes de repetir el paso de “detección y transmisión cuando el canal esté inactivo”. Por otro lado, si el intervalo es pequeño y el número de nodos que colisionan es grande, es probable que los valores aleatorios elegidos sean casi iguales, por lo que los nodos que transmitan volverán a colisionar. Lo que queríamos es elegir un intervalo

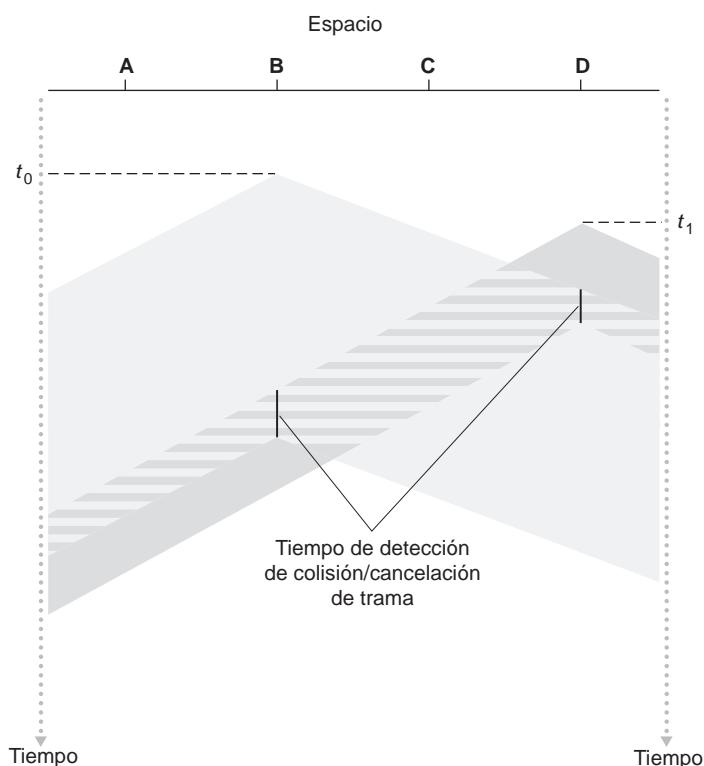


Figura 6.13 ♦ CSMA con detección de colisiones.

que sea corto cuando el número de nodos que colisionan es pequeño, y largo cuando el número de nodos que colisionan sea grande.

El algoritmo de **backoff exponencial binario**, utilizado en Ethernet y en los protocolos de acceso múltiple de red por cable DOCSIS [DOCSIS 2011], resuelve este problema de manera elegante. Específicamente, al transmitir una trama que ya ha experimentado n colisiones un nodo elige el valor de K de forma aleatoria del conjunto $\{0, 1, 2, \dots, 2^{n-1}\}$. Así, cuantas más colisiones experimente una trama, mayor será el intervalo del que se seleccione K . En Ethernet, el periodo de tiempo real que un nodo espera es $K \cdot 512$ períodos de bit (es decir, K veces la cantidad de tiempo necesaria para enviar 512 bits en Ethernet) y el valor máximo que n puede tomar se establece en 10.

Veamos un ejemplo. Supongamos que un nodo trata de transmitir una trama por primera vez y durante la transmisión detecta una colisión. El nodo elige entonces $K = 0$ con una probabilidad de 0,5 o $K = 1$ con una probabilidad de 0,5. Si el nodo elige $K = 0$, entonces comienza de forma inmediata a detectar el canal. Si elige $K = 1$, espera 512 períodos de bit (es decir, 5,12 microsegundos en el caso de Ethernet a 100 Mbps) antes de iniciar el ciclo de “detección y transmisión cuando el canal esté inactivo”. Después de una segunda colisión, K se elige con la misma probabilidad del conjunto $\{0, 1, 2, 3\}$. Después de tres colisiones, K se elige con la misma probabilidad del conjunto $\{0, 1, 2, 3, 4, 5, 6, 7\}$. Tras 10 o más colisiones, K se selecciona con la misma probabilidad del conjunto $\{0, 1, 2, \dots, 1023\}$. Por tanto, el tamaño de los conjuntos de los que se selecciona K crece exponencialmente con el número de colisiones; por esta razón este algoritmo se denomina algoritmo de *backoff exponencial binario*.

Observe también que, cada vez que un nodo prepara una nueva trama para su transmisión, ejecuta el algoritmo CSMA/CD, no teniendo en cuenta las colisiones que hayan tenido lugar en el pasado reciente. Por tanto, es posible que un nodo con una nueva trama sea capaz de llevar a cabo una transmisión con éxito de forma inmediata mientras que otros nodos se encuentran en el estado de *backoff exponencial*.

Eficiencia de CSMA/CD

Cuando solo un nodo tiene una trama para enviar, el nodo puede transmitir a la velocidad máxima del canal (por ejemplo, velocidades típicas de Ethernet son 10 Mbps, 100 Mbps o 1 Gbps). Sin embargo, si muchos nodos tienen tramas que transmitir, la velocidad efectiva de transmisión del canal puede ser mucho menor. Definimos **eficiencia de CSMA/CD** como la fracción (a largo plazo) de tiempo durante el que las tramas están siendo transmitidas al canal sin colisiones, cuando existe un gran número de nodos activos, teniendo cada uno de ellos una gran cantidad de tramas para enviar. Para obtener una buena aproximación de la eficiencia de Ethernet, definimos d_{prop} como el tiempo máximo que tarda la intensidad de la señal en propagarse entre dos adaptadores cualesquiera. Sea d_{trans} el tiempo necesario para transmitir una trama de tamaño máximo (aproximadamente 1,2 milisegundos para Ethernet a 10 Mbps). Una demostración de la expresión de la eficiencia de CSMA/CD queda fuera del alcance de este libro (consulte [Lam 1980] y [Bertsekas 1991]). A continuación proporcionamos simplemente la siguiente aproximación:

$$\text{Eficiencia} = \frac{1}{1 + 5d_{\text{prop}} / d_{\text{trans}}}$$

A partir de esta fórmula vemos que cuando d_{prop} se aproxima a 0 la eficiencia tiende a 1. Esto confirma la idea intuitiva de que si el retardo de propagación es cero, los nodos que han colisionado abortarán de forma inmediata sus transmisiones, evitando así que el canal se desperdicie. También, cuando d_{trans} se hace muy grande, la eficiencia tiende a 1. Esto es igualmente intuitivo, ya que cuando una trama se apropiá del canal se mantendrá en él durante bastante tiempo y, por tanto, el canal estará realizando un trabajo productivo la mayor parte del tiempo.

6.3.3 Protocolos de toma de turnos

Recuerde que dos propiedades deseables de un protocolo de acceso múltiple son: (1) cuando solo haya un nodo activo, este tendrá una tasa de transferencia de R bps y (2) cuando haya M nodos activos, entonces cada nodo activo dispondrá de una tasa de transferencia de aproximadamente R/M bps. Los protocolos ALOHA y CSMA presentan la primera de las propiedades, pero no la segunda. Esto ha servido de motivación para que una serie de investigadores creen otra clase de protocolos: los **protocolos de toma de turnos**. Al igual que con los protocolos de acceso aleatorio, existen docenas de protocolos de toma de turnos distintos y cada uno de estos protocolos tiene múltiples variantes. Aquí vamos hablar de dos de los protocolos más importantes de este tipo. El primero es el **protocolo de sondeo** (*polling*). Este protocolo requiere que se designe a uno de los nodos como nodo maestro. El nodo maestro **sondea** a cada uno de los otros nodos a la manera de turno rotatorio (*round robin*). En particular, el nodo maestro envía primero un mensaje al nodo 1, diciéndole que puede transmitir hasta un cierto número máximo de tramas. Después de que el nodo 1 transmite una serie de tramas, el nodo maestro le dirá al nodo 2 que puede transmitir hasta el máximo número de tramas. (El nodo maestro puede determinar cuándo un nodo ha terminado de enviar sus tramas observando la falta de señal en el canal.) El procedimiento continúa de esta forma ininterrumpidamente, encargándose el nodo maestro de sondear a cada uno de los otros nodos de forma cíclica.

El protocolo de sondeo elimina las colisiones y las particiones vacías que infectan los protocolos de acceso aleatorio. Esto permite que el mecanismo de sondeo consiga una eficiencia mucho mayor, aunque también presenta algunas desventajas. La primera es que el protocolo introduce un retardo de sondeo: el intervalo de tiempo requerido para indicarle a un nodo que puede transmitir. Por ejemplo, si solo hay un nodo activo, entonces el nodo transmitirá a una velocidad menor que R bps, ya que el nodo maestro deberá sondear a cada uno de los nodos inactivos por turno cada vez que el nodo activo haya terminado de enviar su número máximo de tramas. El segundo inconveniente, que puede ser más grave, es que si el nodo maestro falla entonces todo el canal dejará de estar operativo. El protocolo 802.15 y el protocolo Bluetooth que estudiaremos en la Sección 7.3 son ejemplos de protocolos de sondeo.

El segundo protocolo de toma de turnos es el **protocolo de paso de testigo**. En este protocolo no existe ningún nodo maestro; en su lugar hay una trama de pequeño tamaño y de propósito especial conocida con el nombre de **testigo** (*token*) que va siendo intercambiada entre los nodos en un determinado orden fijo. Por ejemplo, puede que el nodo 1 envíe el testigo siempre al nodo 2, el nodo 2 puede mandarlo al nodo 3 y el nodo N debe enviarlo siempre al nodo 1. Cuando un nodo recibe el testigo, lo retiene solo en el caso de que tenga alguna trama para transmitir; en caso contrario, reenvía inmediatamente el testigo al siguiente nodo. Si un nodo tiene tramas que transmitir cuando recibe el testigo, envía una trama detrás de otra, hasta el número máximo de tramas permitido y luego reenvía el testigo al siguiente nodo. El mecanismo de paso de testigo es descentralizado y extremadamente eficiente, aunque también tiene sus propios problemas. Por ejemplo, el fallo de un nodo puede hacer que todo el canal quede inutilizable, o si un nodo se olvidara accidentalmente de liberar el testigo, entonces sería necesario invocar algún procedimiento de recuperación para hacer que el testigo vuelva a circular. A lo largo de los años se han desarrollado muchos protocolos de paso de testigo entre los que se incluyen el protocolo de interfaz de datos distribuida para fibra (FDDI, *Fiber Distributed Data Interface*) [Jain 1994] y el protocolo de paso de testigo en anillo IEEE 802.5 [IEEE 802.5 2012], y cada uno de ellos tuvo que preocuparse de resolver estos problemas, así como algunos otros problemas complicados.

6.3.4 DOCSIS: el protocolo de la capa de enlace para acceso a Internet por cable

En las tres subsecciones anteriores hemos estudiado tres clases amplias de protocolos de acceso múltiple: protocolos de particionamiento del canal, protocolos de acceso aleatorio y protocolos de toma de turnos. Una red de acceso por cable constituye aquí un excelente caso de estudio, ya que podremos encontrar aspectos de *cada una* de estas tres clases de protocolos de acceso múltiple dentro de dicha red.

Recuerde, de la Sección 1.2.1, que una red de acceso por cable suele conectar varios miles de módems domésticos de acceso por cable a un sistema de terminación de módem por cable (CMTS, *Cable Modem Termination System*) en la cabecera de la red de cable. La Especificación de interfaz de servicio de datos por cable (DOCSIS, *Data-Over-Cable Service Interface Specification*) [DOCSIS 2011] especifica la arquitectura de la red de datos por cable y sus protocolos. DOCSIS utiliza FDM para dividir los segmentos de red de bajada (del CMTS al módem) y de subida (del módem al CMTS) en múltiples canales de frecuencia. Cada canal de bajada tiene una anchura de 6 MHz, con una tasa de transferencia máxima de aproximadamente 40 Mbps por canal (aunque, en la práctica, raras veces se suele ver esta velocidad de datos en los modems por cable); cada canal de subida tiene un ancho de banda máximo de 6,4 MHz y una tasa de transferencia máxima de subida de aproximadamente 30 Mbps. Cada canal de subida y de bajada es un canal de difusión. Las tramas transmitidas por el CMTS a través del canal de bajada son recibidas por todos los modems por cable que reciben dicho canal; sin embargo, como hay un único CMTS transmitiendo a través del canal de bajada, no existen problemas de acceso múltiple. La dirección de subida, por su parte, es más interesante y presenta mayores desafíos técnicos, ya que múltiples modems por cable comparten el mismo canal (frecuencia) de subida hacia el CMTS, por lo que podrían producirse colisiones.

Como se ilustra en la Figura 6.14, cada canal de subida está dividido en intervalos de tiempo (al estilo de TDM), cada uno de los cuales contiene una secuencia de mini-particiones durante las cuales los modems por cable pueden transmitir hacia el CMTS. El CMTS concede explícitamente permiso a los modems por cable individuales para transmitir durante ciertas mini-particiones específicas. El CMTS lleva esto a cabo enviando un mensaje de control, conocido como mensaje MAP, a través de un canal de bajada para especificar qué módem por cable (que tenga datos para enviar) puede transmitir durante qué mini-partición, para el intervalo de tiempo especificado en el mensaje de control. Puesto que las mini-particiones se asignan explícitamente a los modems por cable, el CMTS puede garantizar que no se producirá colisión de transmisiones durante una mini-partición.

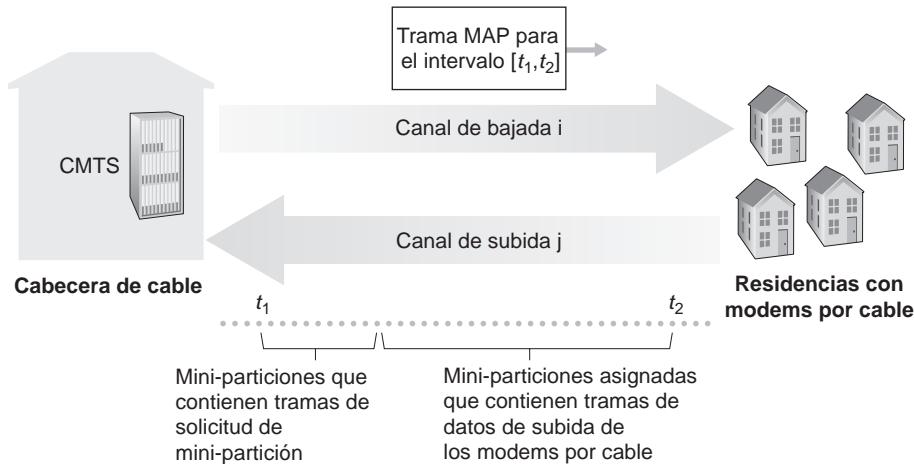


Figura 6.14 ♦ Canales de subida y de bajada entre el CMTS y los modems por cable.

¿Pero cómo sabe el CMTS qué módems tienen datos para enviar? Esto se consigue haciendo que los modems envíen al CMTS tramas de solicitud de mini-partición, durante un intervalo especial de mini-particiones que están dedicadas a este propósito, como se muestra en la Figura 6.14. Estas tramas de solicitud de mini-partición se transmiten por el sistema de acceso aleatorio, por lo que pueden colisionar. Un módem no puede detectar si el canal de subida está ocupado, ni tampoco puede detectar las colisiones. En lugar de ello, el módem deduce que su trama de solicitud de mini-partición sufrió una colisión si no recibe una respuesta a la asignación solicitada en el siguiente mensaje de control recibido a través del canal de bajada. Cuando deduce que se ha producido una colisión, el módem utiliza la técnica de *backoff* exponencial binario para diferir hasta un instante posterior la retransmisión de su trama de solicitud de mini-partición. Cuando hay poco tráfico en el canal de subida, el módem puede, en la práctica, enviar tramas de datos durante las ranuras nominalmente asignadas a las tramas de solicitud de mini-partición (evitando así tener que esperar a la asignación de una mini-partición).

Una red de acceso por cable es, por tanto, un excelente ejemplo de los protocolos de acceso múltiple en acción: ¡FDM, TDM, acceso aleatorio y ranuras temporales asignadas de forma centralizada, todo en la misma red!

6.4 Redes de área local conmutadas

Una vez estudiados en la sección anterior los protocolos de acceso múltiple y las redes de difusión, vamos a ocuparnos ahora de las redes de área local conmutadas. La Figura 6.15 muestra una red local conmutada que conecta tres departamentos, dos servidores y un router con cuatro switches. Puesto que estos switches operan en la capa de enlace, intercambian tramas de la capa de enlace (en lugar de datagramas de la capa red), no reconocen las direcciones de la capa de red y no emplean algoritmos de enrutamiento como RIP u OSPF para determinar las rutas a través de la red de switches de capa 2. En lugar de utilizar direcciones IP, veremos enseguida que emplean direcciones de la capa de enlace para reenviar las tramas de la capa de enlace a través de la red de switches. Vamos a iniciar nuestro estudio sobre las redes LAN conmutadas abordando en primer lugar el direccionamiento de la capa de enlace (Sección 6.4.1). A continuación examinaremos el famoso protocolo Ethernet (Sección 6.5.2). Una vez analizados el direccionamiento de la capa de enlace y Ethernet, veremos cómo operan los switches de la capa de enlace (Sección 6.4.3) y después veremos (en la Sección 6.4.4) cómo suelen emplearse estos switches para construir redes LAN a gran escala.

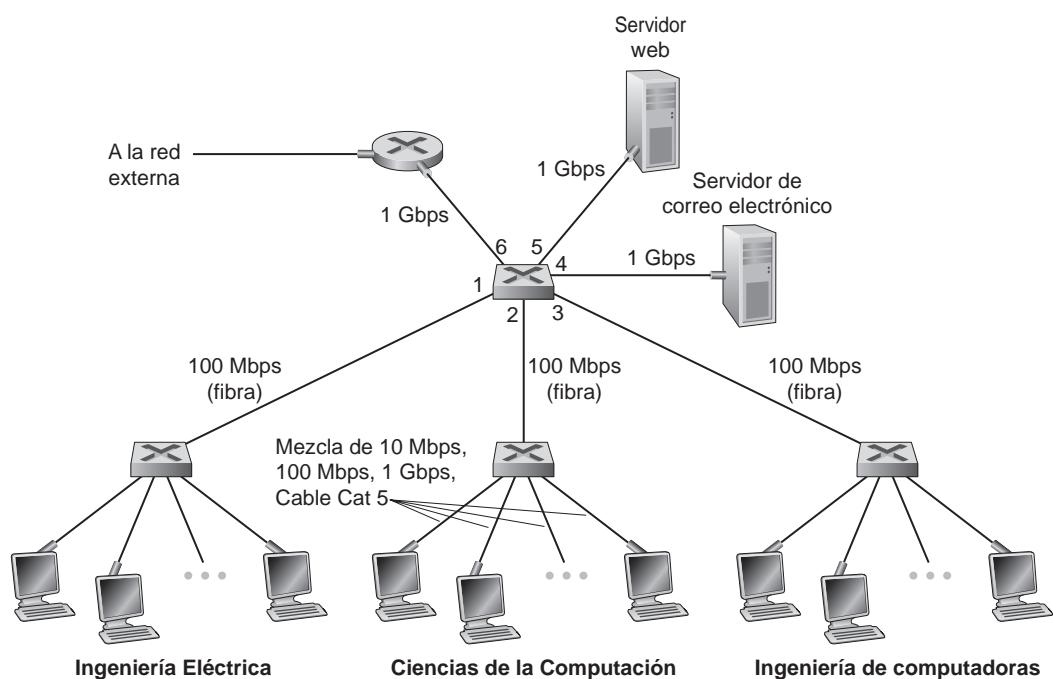


Figura 6.15 ♦ Red institucional conectada mediante cuatro switches.

6.4.1 Direcciónamiento de la capa de enlace y ARP

Los hosts y los routers tienen direcciones de la capa de enlace. Esto podría parecer sorprendente si recordamos del Capítulo 4 que los hosts y los routers tienen también direcciones de la capa de red. Es posible que el lector se esté preguntando para qué necesitamos disponer de direcciones tanto en la capa de red como en la de enlace. Además de describir la sintaxis y la función de las direcciones de la capa de enlace, en esta sección confiamos en arrojar algo de luz sobre las razones por las que resulta útil emplear las dos capas de direcciones y, de hecho, veremos que esto es indispensable. También nos ocuparemos del Protocolo de resolución de direcciones (ARP, *Address Resolution Protocol*), que proporciona un mecanismo para traducir las direcciones IP en direcciones de la capa de enlace.

Direcciones MAC

En realidad, no son los hosts ni los routers los que tienen asignadas direcciones de la capa de enlace, sino que las direcciones de la capa de enlace se asignan a sus adaptadores (es decir, sus interfaces de red). Un host o un router con múltiples interfaces de red tendrá asociadas, por tanto, múltiples direcciones de la capa de enlace, igual que tendrá asociadas múltiples direcciones IP. No obstante, es importante observar que los switches de la capa de enlace no tienen direcciones de la capa de enlace asociadas con sus interfaces que se conectan a los hosts y a los routers. Esto es así porque el trabajo de un switch es transportar datagramas entre hosts y routers; un switch lleva a cabo este trabajo de forma transparente, es decir, sin que el host o el router tengan que dirigir la trama explícitamente hacia el switch intermedio. Esto se ilustra en la Figura 6.16. A las direcciones de la capa de enlace se las denomina de diversas formas, como **dirección LAN**, **dirección física** o **dirección MAC**. Dado que el término dirección MAC parece ser el más popular, en lo sucesivo nos referiremos a las direcciones de la capa de enlace utilizando dicho término. En la mayoría de las redes LAN (incluyendo las redes Ethernet y las LAN inalámbricas 802.11) la dirección MAC tiene 6 bytes de longitud, lo que nos da 2^{48} posibles direcciones MAC. Como se muestra en la Figura 6.16, estas direcciones de

6 bytes suelen expresarse en notación hexadecimal, indicándose cada byte de la dirección mediante una pareja de números hexadecimales. Aunque las direcciones MAC se diseñaron para ser permanentes, hoy día es posible modificar la dirección MAC de un adaptador mediante un software apropiado. Sin embargo, en el resto de esta sección supondremos que la dirección MAC de un adaptador es fija.

Una propiedad interesante de las direcciones MAC es que nunca puede haber dos adaptadores con la misma dirección. Esto puede parecer sorprendente, dado que los adaptadores son fabricados por muchas compañías distintas en muchos países diferentes. ¿Cómo puede una empresa fabricante de adaptadores de Taiwan estar segura de que está utilizando un conjunto diferente de direcciones del que emplea otra empresa que fabrica adaptadores en Bélgica? La respuesta es que el IEEE se encarga de gestionar el espacio de direcciones MAC. En particular, cuando una empresa quiere fabricar adaptadores, compra por un precio fijado una parte del espacio de direcciones compuesta por 2^{24} direcciones. IEEE asigna el fragmento de 2^{24} direcciones fijando los primeros 24 bits de una dirección MAC y dejando que la empresa diseñe combinaciones únicas de los últimos 24 bits para cada adaptador.

La dirección MAC de un adaptador tiene una estructura plana (en oposición a una estructura jerárquica) y nunca varía, independientemente de a dónde se lleve el adaptador. Una computadora portátil con una tarjeta Ethernet siempre tendrá la misma dirección MAC, independientemente de dónde se utilice esa computadora. Un smartphone con una interfaz 802.11 tendrá siempre también la misma dirección MAC, independientemente de dónde lo llevemos. Recuerde que, por contraste, las direcciones IP tienen una estructura jerárquica (es decir, una parte de red y una parte de host) y que es necesario modificar la dirección IP de un nodo cuando el host se mueve, es decir, cuando cambia la red a la que está conectado. La dirección MAC de un adaptador es análoga al número del carnet de identidad o de la seguridad social de una persona, que también tiene una estructura de direccionamiento plana que no cambia, independientemente de a dónde se vaya a vivir esa persona. Una dirección IP, por su parte, sería análoga a la dirección postal de una persona, que es jerárquica y que debe modificarse cada vez que la persona cambia de domicilio. Al igual que para las personas resulta útil disponer tanto de una dirección postal como de un número de la seguridad social, también para las interfaces de un host o un router resulta útil disponer de una dirección de la capa de red y de una dirección MAC.

Cuando un adaptador quiere enviar una trama a otro adaptador de destino, inserta la dirección MAC del adaptador de destino en la trama y luego la envía a través de la red LAN. Como veremos enseguida, en ocasiones los switches difunden una trama entrante a través de todas sus interfaces. En el Capítulo 7 veremos que las redes 802.11 también difunden tramas. Así, un adaptador puede recibir

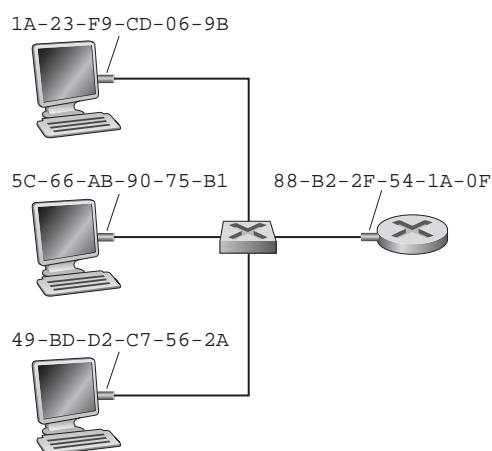


Figura 6.16 ♦ Cada interfaz conectada a una LAN dispone de una dirección MAC única.

una trama que no va dirigida a él. Por tanto, cuando un adaptador recibe una trama, la comprobará para ver si la dirección MAC de destino contenida en esa trama se corresponde con su propia dirección MAC. Si existe una correspondencia, el adaptador extraerá el datagrama incluido en la trama y lo pasará hacia arriba por la pila de protocolos. Si no existe una correspondencia, el adaptador descarta la trama sin pasar el datagrama de la capa de red hacia arriba por la pila de protocolos. De este modo, el nodo de destino solo será interrumpido cuando se reciba la trama.

Sin embargo, en ocasiones un adaptador de un emisor *sí* que quiere que todos los demás adaptadores de la LAN reciban y procesen la trama que va a enviar. En este caso, el adaptador emisor inserta una **dirección de difusión** MAC especial en el campo de la dirección de destino de la trama. Para las redes LAN que utilizan direcciones de 6 bytes (como las LAN Ethernet y 802.11), la dirección de difusión es una cadena compuesta por 48 unos (1) consecutivos (es decir, FF-FF-FF-FF-FF-FF en notación hexadecimal).

Protocolo de resolución de direcciones (ARP)

Dado que existen tanto direcciones de la capa de red (por ejemplo, direcciones IP de Internet) como direcciones de la capa de enlace (es decir, direcciones MAC), surge la necesidad de una traducción entre ellas. En Internet, esta tarea la lleva a cabo el **protocolo ARP (Address Resolution Protocol, Protocolo de resolución de direcciones)** [RFC 826].

Para comprender la necesidad de un protocolo como ARP, considere la red mostrada en la Figura 6.17. En este sencillo ejemplo, cada host y router tiene una dirección IP única y una dirección MAC única. Como siempre, las direcciones IP se muestran en notación decimal con punto y las direcciones MAC en notación hexadecimal. Para seguir esta exposición, vamos a suponer en esta sección que el switch difunde todas las tramas; es decir, cuando un switch recibe una trama a través de una interfaz la reenvía a todas sus restantes interfaces. En la siguiente sección proporcionaremos una explicación más precisa de cómo funcionan los switches.

Supongamos ahora que el host con la dirección IP 222.222.222.220 desea enviar un datagrama IP al host 222.222.222.222. En este ejemplo, tanto el origen como el destino se encuentran en la misma subred, en el sentido de direccionamiento expresado en la Sección 4.3.3. Para enviar un datagrama, el origen tiene que proporcionar a su adaptador no solo el datagrama IP sino también la dirección MAC del destino 222.222.222.222. El adaptador del emisor construirá entonces una trama de la capa de enlace que contendrá la dirección MAC del destino y enviará la trama a la red LAN.

La pregunta fundamental en esta sección es: ¿cómo determina el host emisor la dirección MAC del host de destino con la dirección IP 222.222.222.222? Como posiblemente ya haya

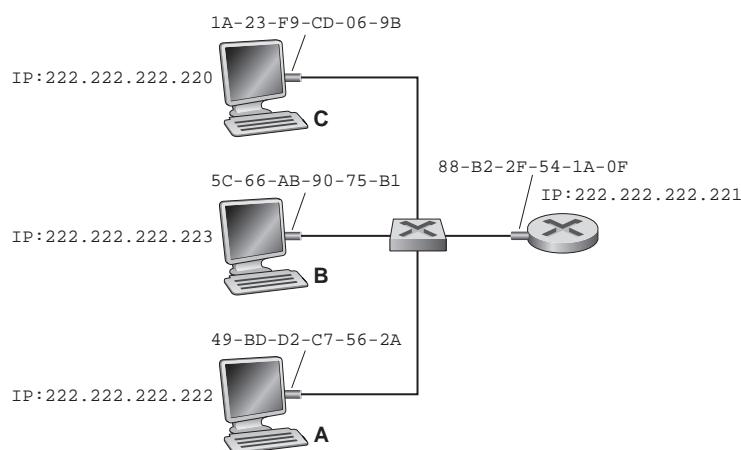


Figura 6.17 ♦ Cada interfaz de una LAN tiene una dirección IP y una dirección MAC.



EN LA PRÁCTICA

MANTENER LAS CAPAS INDEPENDIENTES

Existen varias razones por las que las interfaces de hosts y routers tienen direcciones MAC además de las direcciones de la capa de red. En primer lugar, las redes LAN están diseñadas para protocolos arbitrarios de la capa de red, no solo para IP e Internet. Si los adaptadores de red tuvieran asignadas direcciones IP en lugar de direcciones MAC “neutrales”, entonces no podrían dar soporte fácilmente a otros protocolos de la capa de red (como por ejemplo, IPX o DECnet). En segundo lugar, si los adaptadores utilizaran direcciones de la capa de red en lugar de direcciones MAC, la dirección de la capa de red se tendría que almacenar en la memoria RAM del adaptador y tendría que reconfigurarse cada vez que el adaptador se moviera (o se encendiera). Otra opción sería no utilizar ninguna dirección en los adaptadores y que cada adaptador pasara los datos (normalmente, un datagrama IP) de cada trama recibida hacia arriba por la pila de protocolos. La capa de red podría entonces comprobar si existe una coincidencia con la dirección de la capa de red. Un problema que surge con esta opción es que el host sería interrumpido por cada trama enviada a través de la LAN, incluyendo aquellas tramas que estuvieran destinadas a otros hosts de la misma LAN de difusión. En resumen, con el fin de que las capas sean independientes en gran medida en una arquitectura de red, las distintas capas necesitan disponer de su propio esquema de direccionamiento. Hasta ahora hemos hablado de tres tipos de direcciones: nombres de host para la capa de aplicación, direcciones IP para la capa de red y direcciones MAC para la capa de enlace.

adivinado, utiliza el protocolo ARP. Un módulo ARP en el host emisor toma como entrada cualquier dirección IP de la misma LAN y devuelve la dirección MAC correspondiente. En nuestro ejemplo, el host emisor 222.222.222.220 proporciona a su módulo ARP la dirección IP 222.222.222.222 y el módulo ARP devuelve la correspondiente dirección MAC 49-BD-D2-C7-56-2A.

Vemos por tanto que ARP resuelve una dirección IP en una dirección MAC. En muchos sentidos, esto es análogo a DNS (estudiado en la Sección 2.5), que resuelve nombres de host en direcciones IP. Sin embargo, una diferencia importante entre los dos resolvedores es que DNS resuelve nombres de host para hosts ubicados en cualquier lugar de Internet, mientras que ARP resuelve direcciones IP solo para las interfaces de los hosts y routers de una misma subred. Si un nodo situado en California intentara utilizar ARP para resolver la dirección IP de un nodo en Mississippi, ARP devolvería un error.

Ahora que hemos explicado lo que hace ARP, vamos a ver cómo lo hace. Cada host o router tiene en su memoria una **tabla ARP**, que contiene las correspondencias entre las direcciones IP y las direcciones MAC. La Figura 6.18 muestra el aspecto que puede tener la tabla ARP del host 222.222.222.220. La tabla ARP también contiene un valor de tiempo de vida (TTL), que indica cuándo se eliminará cada correspondencia de la tabla. Observe que la tabla no necesariamente contiene una entrada para cada host y router de la subred; algunos pueden haber tenido entradas que han caducado, mientras que otros puede que nunca hayan tenido una entrada en la tabla. El tiempo típico de caducidad de una entrada es de 20 minutos desde el momento que se incluye la entrada en la tabla ARP.

Suponga ahora que el host 222.222.222.220 quiere enviar un datagrama con direccionamiento IP a otro host o router de dicha subred. El host emisor necesita obtener la dirección MAC del nodo de destino dada la dirección IP. Esta tarea es fácil si la tabla ARP del emisor tiene una entrada para el nodo de destino. Pero, ¿qué ocurre si la tabla ARP no contiene actualmente una entrada para el nodo de destino? En particular, suponga que el host 222.222.222.220 desea enviar un datagrama al host 222.222.222.222. En este caso, el emisor utiliza el protocolo ARP para resolver la dirección. En primer lugar, el emisor construye un paquete especial denominado **paquete ARP**. Un paquete ARP contiene varios campos, incluyendo las direcciones MAC e IP del emisor y el receptor. Los paquetes de consulta y de respuesta ARP tienen el mismo formato.

Dirección IP	Dirección MAC	TTL
222.222.222.221	88-B2-2F-54-1A-0F	13:45:00
222.222.222.223	5C-66-AB-90-75-B1	13:52:00

Figura 6.18 ♦ Una posible tabla ARP en el nodo 222.222.222.220.

El propósito del paquete de consulta ARP es consultar a todos los demás hosts y routers de la subred con el fin de determinar la dirección MAC correspondiente a la dirección IP que está resolviendo.

Volvamos a nuestro ejemplo. El nodo 222.222.222.220 pasa un paquete de consulta ARP al adaptador junto con una indicación de que el adaptador debe enviar el paquete a la dirección de difusión MAC, FF-FF-FF-FF-FF-FF. El adaptador encapsula el paquete ARP en una trama de la capa de enlace, utiliza la dirección de difusión como dirección de destino de la trama y la transmite a la subred. Recuerde la analogía del número de la seguridad social y la dirección postal: una consulta ARP es equivalente a una persona gritando en una sala abarrotada de cubículos de alguna empresa (por ejemplo, CualquierEmpresa): “¿Cuál es el número de la seguridad social de la persona cuya dirección postal es Cubículo 13, Sala 112, CualquierEmpresa, Palo Alto, California?” La trama que contiene la consulta ARP es recibida por todos los demás adaptadores existentes en la subred y (a causa de la dirección de difusión) cada adaptador pasa la consulta ARP contenida en la trama a su módulo ARP. Cada uno de estos módulos ARP realiza una comprobación para ver si su dirección IP se corresponde con la dirección IP de destino del paquete ARP. El único nodo en el que se produzca la coincidencia devolverá al nodo que ha realizado la consulta una respuesta ARP con la correspondencia deseada. El host que ha realizado la consulta 222.222.222.220 podrá entonces actualizar su tabla ARP y enviar su datagrama IP, encapsulado dentro de una trama de la capa de enlace cuya dirección de destino MAC es la del host o router que ha contestado a la anterior consulta ARP.

Hay un par de cosas interesantes que comentar acerca del protocolo ARP. En primer lugar, el mensaje ARP de consulta se envía dentro de una trama de difusión, mientras que el mensaje ARP de respuesta se envía dentro de una trama estándar. Antes de continuar leyendo debería pararse a pensar por qué esto es así. En segundo lugar, ARP es plug-and-play; es decir, la tabla ARP de un nodo se construye automáticamente (no tiene que ser configurada por el administrador del sistema). Y si un host se desconecta de la subred, su entrada terminará por eliminarse de las restantes tablas ARP de la subred.

A menudo, los estudiantes se preguntan si ARP es un protocolo de la capa de enlace o un protocolo de la capa de red. Como hemos visto, un paquete ARP se encapsula dentro de una trama de la capa de enlace y así se sitúa, arquitectónicamente, encima de la capa de enlace. Sin embargo, un paquete ARP dispone de campos que contienen direcciones de la capa de enlace, por lo que se podría decir que es un protocolo de la capa de enlace, pero también contiene direcciones de la capa de red y, por tanto, podría también argumentarse que es un protocolo de la capa de red. En último término, probablemente ARP sea considerado un protocolo que se encuentra a caballo entre las capas de enlace y de red (no se ajusta limpiamente a la pila de protocolos en capas simples que hemos estudiado en el Capítulo 1). ¡Tales son las complejidades de los protocolos del mundo real!

Envío de un datagrama fuera de la subred

Ahora debería tener claro cómo funciona ARP cuando un host desea enviar un datagrama a otro host que se encuentra *en la misma subred*. Pasemos entonces a una situación algo más complicada, en la que un host de una subred desea enviar un datagrama de la capa de red a un host que está *frente a la subred* (es decir, a través de un router a otra subred). Estudiemos esta situación en el contexto de la Figura 6.19, que muestra una red simple formada por dos subredes interconectadas mediante un router.

Hay varias cuestiones interesantes que destacar en la Figura 6.19. Cada host tiene exactamente una dirección IP y un adaptador. Pero, como se ha visto en el Capítulo 4, un router tiene una dirección IP para *cada* una de sus interfaces. Para cada interfaz de router existe también un módulo ARP (en el router) y un adaptador. Dado que el router de la Figura 6.19 tiene dos interfaces, tendrá dos direcciones IP, dos módulos ARP y dos adaptadores. Por supuesto, cada adaptador de la red tiene su propia dirección MAC.

Fíjese también en que la Subred 1 tiene la dirección de red 111.111.111/24 y que la Subred 2 tiene la dirección de red 222.222.222/24. Así, todas las interfaces conectadas a la Subred 1 tienen direcciones de la forma 111.111.111.xxx y todas las interfaces conectadas a la Subred 2 tienen direcciones de la forma 222.222.222.xxx.

Examinemos ahora cómo un host de la Subred 1 enviaría un datagrama a un host de la Subred 2. Específicamente, suponga que el host 111.111.111.111 desea enviar un datagrama IP a un host 222.222.222.222. Como es habitual, el host emisor pasa el datagrama a su adaptador. Pero el host emisor también tiene que indicar a su adaptador una dirección MAC de destino apropiada. ¿Qué dirección MAC debería utilizar el adaptador? Una posibilidad sería probar si la dirección MAC apropiada es la del adaptador para el host 222.222.222.222, es decir, 49-BD-D2-C7-56-2A. Sin embargo, esta suposición resultaría errónea. Si el adaptador del emisor utilizara dicha dirección MAC, entonces ninguno de los adaptadores de la Subred 1 se molestaría en pasar el datagrama IP a su capa de red, ya que la dirección de destino de la trama no coincidiría con la dirección MAC de ningún adaptador de la Subred 1. El datagrama terminaría muriendo e iría al cielo de los datagramas.

Si nos fijamos en la Figura 6.19 vemos que para que un datagrama vaya desde 111.111.111.111 a un host de la Subred 2, el datagrama tiene en primer lugar que ser enviado a la interfaz de router 111.111.111.110, que es la dirección IP del router del primer salto en el camino hacia su destino final. Por tanto, la dirección MAC apropiada para la trama es la dirección del adaptador de la interfaz de router 111.111.111.110, es decir, E6-E9-00-17-BB-4B. ¿Cómo adquiere el host emisor la dirección MAC para la dirección 111.111.111.110? ¡Por supuesto, utilizando ARP! Una vez que el adaptador del emisor tiene esta dirección MAC, crea una trama (que contiene el datagrama direccional a 222.222.222.222) y envía la trama hacia la Subred 1. El adaptador del router de la Subred 1 ve que la trama de la capa de enlace se dirige hacia él y, por tanto, pasa la trama a la capa de red del router. ¡Estupendo! El diagrama IP se ha transmitido con éxito desde el host de origen al router. Pero todavía no hemos terminado. Todavía nos queda llevar el datagrama desde el router al destino. Ahora el router tiene que determinar la interfaz correcta a la que el datagrama será reenviado. Como se ha explicado en el Capítulo 4, esto se hace consultando la tabla de reenvío del router. La tabla de reenvío indica al router que el datagrama es reenviado a través de la interfaz de router 222.222.222.220. Esta interfaz entonces pasa el datagrama a su adaptador, que encapsula el datagrama en una nueva trama y la envía a la Subred 2. Esta vez, la dirección MAC de destino de la trama es la dirección MAC del destino final. Pero, ¿cómo obtiene el router esta dirección MAC de destino? ¡Por supuesto, de ARP!

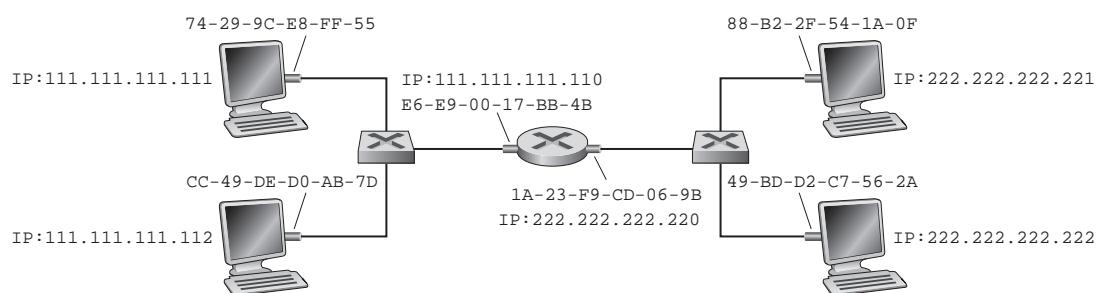


Figura 6.19 ♦ Dos subredes interconectadas mediante un router.

ARP para Ethernet está definido en el documento RFC 826. En el tutorial de TCP/IP, RFC 1180, se proporciona una introducción a ARP. Exploraremos en más detalle ARP en los problemas de repaso.

6.4.2 Ethernet

Ethernet ha avanzado mucho en el mercado de las redes LAN cableadas. En la década de 1980 y a principios de la década de 1990, Ethernet se enfrentó a muchos de los desafíos de otras tecnologías LAN, como Token Ring, FDDI y ATM. Algunas de estas otras tecnologías tuvieron éxito y captaron parte del mercado de las redes LAN durante unos pocos años. Pero, desde su aparición a mediados de la década de 1970, Ethernet ha continuado evolucionando y creciendo y se ha mantenido en una posición dominante. Actualmente, Ethernet es de lejos la tecnología para redes LAN cableadas predominante y, probablemente, se mantendrá ahí en el futuro. Puede decirse que Ethernet ha sido a las redes de área local lo que Internet a las redes globales.

Existen muchas razones por las que Ethernet ha tenido éxito. En primer lugar, Ethernet fue la primera LAN de alta velocidad ampliamente implantada. Puesto que fue implantada muy pronto, los administradores de redes están extremadamente familiarizados con Ethernet (conocen sus grandes y sus rarezas) y fueron reacios a cambiar a otras tecnologías LAN cuando entraron en escena. En segundo lugar, Token Ring, FDDI y ATM eran más complejas y caras que Ethernet, lo que desanimó a los administradores de redes a cambiar. En tercer lugar, la razón más determinante para cambiar a otra tecnología LAN (como FDDI o ATM) era normalmente la más alta velocidad de datos de la nueva tecnología; sin embargo, Ethernet siempre se defendió produciendo versiones que operaban a velocidades iguales o incluso mayores. Ethernet conmutada se introdujo también a principios de la década de 1990, lo que aumentó sus tasas de datos efectivas. Por último, dado que Ethernet ha sido tan popular, el hardware Ethernet (en concreto, los adaptadores y los conmutadores) ha llegado a ser de consumo masivo y notablemente barato.

La LAN Ethernet original fue inventada a mediados de la década de 1970 por Bob Metcalfe y David Boggs. La LAN Ethernet original utilizaba un bus coaxial para interconectar los nodos. Las topologías de bus para Ethernet consiguieron mantenerse durante la década de 1980 y hasta mediados de la década de 1990. Ethernet con una topología de bus es una LAN de difusión (todas las tramas transmitidas viajan hasta *todos* los adaptadores conectados al bus y son procesadas en ellos). Recuerde que ya hemos visto el protocolo de acceso múltiple CSMA/CD de Ethernet con el algoritmo de backoff exponencial en la Sección 6.3.2.

A finales de la década de 1990 la mayor parte de las empresas y universidades habían reemplazado sus redes LAN por instalaciones Ethernet utilizando topologías en estrella basadas en concentradores (hubs). En ese tipo de instalación, los hosts (y los routers) están directamente conectados a un concentrador mediante un cable de cobre de par trenzado. Un **concentrador** es un dispositivo de la capa física que actúa sobre los bits individuales en lugar de sobre las tramas. Cuando un bit, que representa un cero o un uno, llega procedente de una interfaz, el concentrador simplemente vuelve a crear el bit, incrementa su intensidad de energía y lo transmite a todas las demás interfaces. Por tanto, Ethernet con una topología de estrella basada en concentrador es también una red LAN de difusión (cuando un concentrador recibe un bit por una de sus interfaces, envía una copia al resto de sus interfaces). En particular, si un concentrador recibe tramas procedentes de dos interfaces distintas al mismo tiempo, se produce una colisión y los nodos que crean las tramas tendrán que retransmitirlas.

A principios de la década de 2000 Ethernet experimentó una cambio evolutivo aún mayor. Las instalaciones Ethernet continuaron utilizando una topología en estrella, pero el concentrador central fue reemplazado por un **conmutador** (*switch*). Examinaremos en profundidad las redes Ethernet conmutadas más adelante en este capítulo. Por el momento, solo mencionaremos que un switch no es solo un dispositivo “sin colisiones”, sino que también lleva a cabo la conmutación de paquetes mediante un almacenamiento y reenvío de buena fe; pero, a diferencia de los routers, que operan hasta la capa 3, un switch opera solo hasta la capa 2.

Estructura de la trama Ethernet

Podemos aprender mucho acerca de Ethernet examinando la trama que utiliza, la cual se muestra en la Figura 6.20. Con el fin de proporcionar a esta exposición acerca de las tramas de Ethernet un contexto tangible, consideremos el envío de un datagrama IP desde un host a otro host, estando ambos hosts en la misma red LAN Ethernet (por ejemplo, en la LAN Ethernet de la Figura 6.17). (Aunque la carga útil de nuestra trama Ethernet es un datagrama IP, tenga en cuenta que una trama Ethernet puede transportar también otros paquetes de la capa de red.) El adaptador del emisor, adaptador A, tiene la dirección MAC AA-AA-AA-AA-AA-AA y el adaptador del receptor, adaptador B, tiene la dirección MAC BB-BB-BB-BB-BB-BB. El adaptador del emisor encapsula el datagrama IP dentro de una trama Ethernet y pasa dicha trama a la capa física. El adaptador del receptor recibe la trama de la capa física, extrae el datagrama IP y lo pasa a la capa de red. En este contexto, examinemos los seis campos de la trama Ethernet mostrada en la Figura 6.20.

- *Campo de datos (46 a 1.500 bytes)*. Este campo transporta el datagrama IP. La unidad máxima de transmisión (MTU) de Ethernet es 1.500 bytes, lo que quiere decir que si el datagrama IP excede de 1.500 bytes, entonces el host tiene que fragmentar el datagrama, como se ha explicado en la Sección 4.3.2. El tamaño mínimo del campo de datos es 46 bytes, lo que significa que si el datagrama IP tiene menos de 46 bytes, el campo de datos tiene que ser rellenado hasta los 46 bytes. Cuando se utiliza el relleno, los datos pasados a la capa de red contienen tanto el relleno como el datagrama IP. La capa de red utiliza el campo de longitud de la cabecera del datagrama IP para eliminar el relleno.
- *Dirección de destino (6 bytes)*. Este campo contiene la dirección MAC del adaptador de destino, BB-BB-BB-BB-BB-BB. Cuando el adaptador B recibe una trama Ethernet cuya dirección de destino es BB-BB-BB-BB-BB-BB o la dirección MAC de difusión, pasa el contenido del campo de datos de la trama a la capa de red; si recibe una trama con cualquier otra dirección MAC, descarta la trama.
- *Dirección de origen (6 bytes)*. Este campo contiene la dirección MAC del adaptador que transmite la trama hacia la LAN, en este ejemplo AA-AA-AA-AA-AA-AA.
- *Campo de tipo (2 bytes)*. El campo de tipo permite a Ethernet multiplexar los protocolos de la capa de red. Para comprender esto, tenemos que tener en cuenta que los hosts pueden utilizar otros protocolos de la capa de red además de IP. De hecho, un determinado host puede dar soporte a múltiples protocolos de la capa de red utilizando protocolos distintos para las diferentes aplicaciones. Por esta razón, cuando llega la trama Ethernet al adaptador B este necesita saber a qué protocolo de la capa de red debe pasar (es decir, demultiplexar) el contenido del campo de datos. IP y otros protocolos de la capa red (por ejemplo, Novell IPX o AppleTalk) tienen su propio número de tipo estandarizado. Además, el protocolo ARP (estudiado en la sección anterior) tiene su propio número de tipo y si la trama que llega contiene un paquete ARP (es decir, el campo de tipo contiene el hexadecimal 0806), el paquete ARP será demultiplexado y entregado al protocolo ARP. Observe que el campo de tipo es análogo al campo de protocolo del datagrama de la capa de red y a los campos de número de puerto del segmento de la capa de transporte; todos estos campos sirven para enlazar un protocolo de una capa con un protocolo de la capa superior.

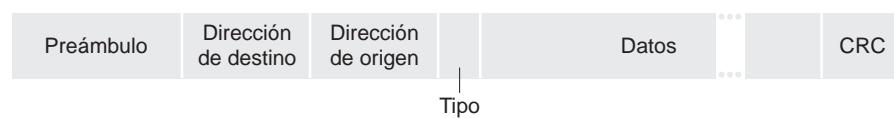


Figura 6.20 ♦ Estructura de la trama Ethernet.

- *Comprobación de redundancia cíclica (CRC) (4 bytes).* Como se ha visto en la Sección 6.2.3, el propósito del campo CRC es permitir que el adaptador del receptor, el adaptador B, detecte los errores de bit de la trama.
- *Preámbulo (8 bytes).* La trama Ethernet comienza con el campo preámbulo de 8 bytes. Cada uno de los siete primeros bytes tiene el valor 10101010 y el último byte tiene el valor 10101011. Los siete primeros bytes sirven para “despertar” a los adaptadores de recepción y sincronizar sus relojes con el reloj del emisor. ¿Por qué podrían estar desincronizados los relojes? Tenga en cuenta que el objetivo del adaptador A es transmitir la trama a 10 Mbps, 100 Mbps o 1 Gbps, dependiendo del tipo de LAN Ethernet. Sin embargo, dado que nada es absolutamente perfecto, el adaptador A no transmitirá la trama a una velocidad exactamente igual a la objetivo; siempre existe cierta *deriva* respecto de dicha velocidad, una deriva que no es conocida *a priori* por los restantes adaptadores de la LAN. Un adaptador de recepción puede sincronizarse con el reloj del adaptador A sincronizándose simplemente con los bits de los siete primeros bytes del preámbulo. Los últimos 2 bits del octavo byte del preámbulo (los dos primeros 1s consecutivos) alertan al adaptador B de que va a llegar “información importante”.

Todas las tecnologías Ethernet proporcionan un servicio sin conexión a la capa de red; es decir, cuando el adaptador A desea enviar un datagrama al adaptador B, el adaptador A encapsula el datagrama en una trama Ethernet y la envía a la red LAN, sin establecer previamente un acuerdo con el adaptador B. Este servicio sin conexión de la capa 2 es análogo al servicio de datagramas de la capa 3 de IP y al servicio sin conexión de la capa 4 de UDP.

Las tecnologías Ethernet proporcionan un servicio no fiable a la capa de red. Específicamente, cuando el adaptador B recibe una trama procedente del adaptador A ejecuta una comprobación CRC de la trama, pero ni envía un mensaje de confirmación cuando la trama pasa la comprobación CRC, ni envía un mensaje de confirmación negativa cuando la comprobación CRC falla. Cuando

HISTORIA

BOB METCALFE Y ETHERNET

Como estudiante de doctorado en la Universidad de Harvard a principios de la década de 1970, Bob Metcalfe trabajaba en la red ARPAnet en el MIT. Durante sus estudios, también conoció el trabajo de Abramson con ALOHA y los protocolos de acceso aleatorio. Después de terminar su doctorado y justo antes de comenzar a trabajar en el Centro de Investigación de Xerox Palo Alto (Xerox PARC), estuvo durante tres meses con Abramson y sus colegas de la universidad de Hawaii y pudo obtener información de primera mano sobre ALOHAnet. En Xerox PARC Metcalfe también conoció las computadoras Alto, que en muchos sentidos fueron las precursoras de las computadoras personales de la década de 1980. Metcalfe vió la necesidad de conectar en red estas computadoras mediante algún sistema económico. Así, armado con sus conocimientos sobre ARPAnet, ALOHAnet y los protocolos de acceso aleatorio, Metcalfe (junto con su colega David Boggs) inventó Ethernet.

La Ethernet original de Metcalfe y Boggs operaba a 2,94 Mbps y podía conectar hasta 256 hosts separados hasta aproximadamente un kilómetro y medio. Metcalfe y Boggs tuvieron éxito al conseguir que la mayoría de los investigadores de Xerox PARC pudieran comunicarse a través de sus computadoras Alto. Metcalfe forjó después una alianza entre Xerox, Digital e Intel para establecer Ethernet a 10 Mbps como estándar del IEEE. Xerox no mostró demasiado interés en la comercialización de Ethernet. En 1979 Metcalfe montó su propia empresa, 3Com, que desarrolló y comercializó tecnología de redes, incluyendo la tecnología Ethernet. En concreto, 3Com desarrolló y puso en el mercado tarjetas Ethernet a principios de la década de 1980 para las tremadamente populares computadoras personales de IBM.

una trama no pasa la comprobación CRC, el adaptador B simplemente la descarta. Por tanto, el adaptador A no sabe si la trama que ha transmitido ha llegado al adaptador B y ha superado la comprobación CRC. Esta ausencia de un transporte fiable (en la capa de enlace) hace que Ethernet sea una tecnología simple y barata. Pero también significa que el flujo de datagramas pasado a la capa de red puede presentar “huecos”.

Si existen huecos porque se han descartado tramas Ethernet, ¿la aplicación del host B ve también esos huecos? Como hemos visto en el Capítulo 3, esto dependerá de si la aplicación está utilizando UDP o TCP. Si la aplicación está empleando UDP, entonces la aplicación del host B verá los huecos en los datos. Por el contrario, si la aplicación está utilizando TCP, entonces TCP en el host B no confirmará los datos contenidos en las tramas descartadas, obligando a TCP en el host A a realizar retransmisiones. Observe que cuando TCP retransmite los datos, estos finalmente volverán al adaptador Ethernet en el que fueron descartados. Por tanto, en este sentido, Ethernet retransmite los datos, aunque no es consciente de si está transmitiendo un nuevo datagrama con nuevos datos, o un datagrama que contiene datos que ya han sido transmitidos al menos una vez.

Tecnologías Ethernet

En nuestra exposición anterior hemos hecho referencia a Ethernet como si fuera un único estándar de protocolo. Pero, en realidad, existen *muchas* versiones diferentes de Ethernet, algunas con acrónimos algo complicados como 10BASE-T, 10BASE-2, 100BASE-T, 1000BASE-LX, 10GBASE-T y 40GBASE-T. Estas y otras muchas tecnologías Ethernet han sido estandarizadas a lo largo de los años por el grupo de trabajo IEEE 802.3 CSMA/CD (Ethernet) [IEEE 802.3 2012]. Aunque estos acrónimos pueden parecer algo complicados, realmente tienen su lógica. La primera parte del acrónimo hace referencia a la velocidad del estándar: 10, 100, 1000 o 10G, para 10 Megabits (por segundo), 100 Megabits, Gigabit, 10 Gigabits y 40 Gigabits Ethernet, respectivamente. “BASE” hace referencia a la tecnología Ethernet banda base, que significa que el medio físico solo transporta tráfico Ethernet; casi todos los estándares 802.3 son para Ethernet banda base. La parte final del acrónimo se refiere al medio físico en sí. Ethernet es una especificación *tanto* de la capa de enlace *como* de la capa física y puede emplear diversos medios físicos, entre los que se incluyen el cable coaxial, el cable de cobre y la fibra. Generalmente la “T” hace referencia al cable de cobre de par trenzado.

Históricamente, Ethernet fue concebida inicialmente como un segmento de cable coaxial. Los primeros estándares 10BASE-2 y 10BASE-5 especificaban Ethernet a 10 Mbps sobre dos tipos de cable coaxial, cada uno de ellos limitado a una longitud de 500 metros. Podían obtenerse recorridos más largos utilizando un **repetidor** (un dispositivo de la capa física que recibe una señal por su entrada y regenera la señal a la salida). Un cable coaxial se corresponde con nuestra visión de Ethernet como un medio de difusión (todas las tramas transmitidas por una interfaz son recibidas en otras interfaces), y el protocolo CDMA/CD de Ethernet resuelve muy bien el problema del acceso múltiple. Los nodos simplemente se conectan al cable, y *voila*, tenemos una red de área local.

La tecnología Ethernet ha experimentado una serie de pasos evolutivos a lo largo de los años y la Ethernet actual es muy diferente de los diseños de las topologías de bus originales que utilizaban cable coaxial. En la mayor parte de las instalaciones actuales, los nodos se conectan a un switch mediante segmentos punto a punto hechos de cable de cobre de par trenzado o de cable de fibra óptica, como se muestra en las Figuras 6.15–6.17.

A mediados de la década de 1990 Ethernet se estandarizó a 100 Mbps, tecnología 10 veces más rápida que la Ethernet a 10 Mbps. El protocolo MAC Ethernet original y el formato de trama se conservaron, pero se definieron capas físicas de mayor velocidad para cable de cobre (100BASE-T) y fibra (100BASE-FX, 100BASE-SX, 100BASE-BX). La Figura 6.21 muestra estos distintos estándares, junto con el protocolo MAC Ethernet y el formato de trama comunes. La tecnología Ethernet a 100 Mbps está limitada a una distancia de 100 metros sobre cable de par trenzado y

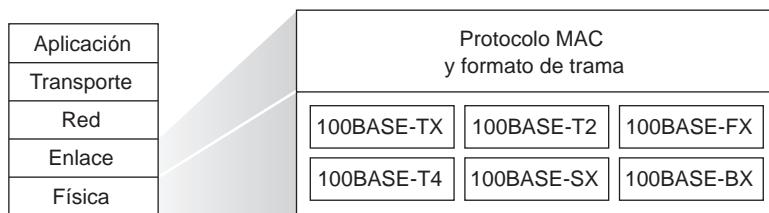


Figura 6.21 ◆ Estándares Ethernet a 100 Mbps: una capa de enlace común y diferentes capas físicas.

a varios kilómetros para cable de fibra, lo que permite conectar switches Ethernet situados en diferentes edificios.

Gigabit Ethernet es una extensión de los exitosos estándares Ethernet a 10 Mbps y 100 Mbps. Ofreciendo una tasa de datos en bruto de 40.000 Mbps, 40 Gigabit Ethernet mantiene una compatibilidad total con la enorme base instalada de equipos Ethernet. El estándar para Gigabit Ethernet, IEEE 802.3z, hace lo siguiente:

- Utiliza el formato de trama Ethernet estándar (Figura 6.20) y es compatible hacia atrás con las tecnologías 10BASE-T y 100BASE-T. Esto permite una fácil integración de Gigabit Ethernet con la base instalada existente de equipos Ethernet.
- Permite el uso de enlaces punto a punto, así como de canales de difusión compartidos. Los enlaces punto a punto utilizan switches mientras que los canales de difusión emplean hubs, como se ha descrito anteriormente. En la jerga de Gigabit Ethernet, los hubs reciben el nombre de *distribuidores con buffer*.
- Utiliza CSMA/CD para los canales de difusión compartidos. Para obtener una eficiencia aceptable, la distancia máxima entre nodos tiene que ser restringida de forma estricta.
- Permite la operación *full-duplex* a 40 Gbps en ambas direcciones en los canales punto a punto.

Operando inicialmente sobre cable de fibra óptica, ahora Gigabit Ethernet es capaz de operar sobre cable UTP de categoría 5.

Vamos a concluir nuestra exposición sobre la tecnología Ethernet planteando una cuestión que puede que le haya estado rondando la cabeza. En la época de las topologías de bus y de las topologías en estrella basadas en hub, Ethernet era claramente un enlace de difusión (como se ha definido en la Sección 6.3) en el que se producían colisiones entre tramas cuando los nodos transmitían al mismo tiempo. Para tratar estas colisiones, el estándar Ethernet incluyó el protocolo CSMA/CD, que es particularmente efectivo en las redes LAN cableadas de difusión con un radio geográfico pequeño. Pero, si el uso prevalente de hoy día de Ethernet es una topología en estrella basada en switches, que utiliza la conmutación de paquetes de almacenamiento y reenvío, ¿existe realmente la necesidad de un protocolo MAC Ethernet? Como veremos enseguida, un switch coordina sus transmisiones y nunca reenvía más de una trama a la misma interfaz en un determinado instante. Además, los switches modernos son full-duplex, por lo que un switch y un nodo pueden enviarse tramas entre sí simultáneamente sin interferir. En otras palabras, en una red LAN Ethernet basada en switches no se producen colisiones y, por tanto, no se necesita un protocolo MAC.

Como hemos visto, las tecnologías Ethernet actuales son *muy* diferentes de la tecnología Ethernet original concebida por Metcalfe y Boggs hace más de 30 años; las velocidades se han incrementado en tres órdenes de magnitud, las tramas Ethernet son transportadas por una amplia variedad de medios, la tecnología Ethernet conmutada se ha convertido en la tecnología dominante y ahora incluso el protocolo MAC con frecuencia no es necesario. ¿Sigue siendo todo esto *realmente* Ethernet? Por supuesto, la respuesta es “sí, por definición”. Sin embargo, es interesante observar que a pesar de todos estos cambios, hay algo que se ha mantenido inalterado a lo largo de estos 30 años:

el formato de la trama Ethernet. Quizá sea entonces esta la única pieza verdaderamente importante del estándar Ethernet.

6.4.3 Switches de la capa de enlace

Hasta el momento no hemos entrado en detalles sobre lo que realmente hace un switch y cómo funciona. La función de un switch es recibir las tramas de la capa de enlace entrantes y reenviarlas a los enlaces de salida; en esta subsección vamos a estudiar la función de reenvío en detalle. El propio switch es **transparente** para los hosts y los routers de la subred; es decir, un host/router dirige una trama a otro host/router (en lugar de dirigirla al switch) y la envía a la red LAN, sin ser consciente de que un switch recibirá la trama y la reenviará. La velocidad a la que llegan las tramas a cualquiera de las interfaces de salida del switch puede ser temporalmente mayor que la capacidad del enlace de dicha interfaz. Para enfrentarse a este problema, las interfaces de salida del switch disponen de buffers, de forma muy parecida a como las interfaces de salida de un router disponen de buffers para los datagramas. Veamos ahora detenidamente cómo funciona un switch.

Reenvío y filtrado

El **filtrado** es la función del switch que determina si una trama debe ser reenviada a alguna interfaz o debe ser descartada. El **reenvío** es la función del switch que determina las interfaces a las que una trama debe dirigirse y luego envía la trama a esas interfaces. Las funciones de filtrado y reenvío del switch se realizan utilizando la **tabla de conmutación**. Esta tabla contiene entradas para algunos de los hosts y routers, no necesariamente todos, de una red LAN. Una entrada de la tabla de conmutación contiene (1) una dirección MAC, (2) la interfaz del switch que lleva hacia dicha dirección MAC y (3) el instante en el que la entrada fue incluida en la tabla. Un ejemplo de tabla de conmutación para el switch superior de la Figura 6.15 se muestra en la Figura 6.22. Esta descripción del reenvío de tramas puede parecer muy similar a lo que hemos visto sobre el reenvío de datagramas en el Capítulo 4. Efectivamente, en nuestra exposición sobre el reenvío generalizado de la Sección 4.4 vimos que es posible configurar muchos conmutadores de paquetes modernos para la función de reenvío basándose en las direcciones MAC de destino de capa 2 (es decir, función como switch de capa 2) o en las direcciones IP de destino de capa 3 (es decir, función como router de la capa 3). No obstante, es importante resaltar que los conmutadores reenvían los paquetes basándose en las direcciones MAC, en lugar de en las direcciones IP. También veremos que la tabla de un switch tradicional (es decir, en un contexto no SDN) se construye de forma muy distinta a como se crea la tabla de reenvío de un router.

Para comprender cómo funciona el filtrado y el reenvío en un switch, suponga que una trama con la dirección de destino DD-DD-DD-DD-DD-DD llega a la interfaz x del switch. Este buscará en su tabla la dirección MAC DD-DD-DD-DD-DD-DD. Se plantean tres posibilidades:

- No hay ninguna entrada en la tabla para DD-DD-DD-DD-DD-DD. En este caso, el switch reenvía copias de la trama a los buffers de salida que preceden a *todas* las interfaces salvo a la interfaz x . En otras palabras, si no hay ninguna entrada para la dirección de destino, el switch difunde la trama.
- Existe una entrada en la tabla que asocia DD-DD-DD-DD-DD-DD con la interfaz x . En este caso, la trama procede de un segmento de la LAN que contiene al adaptador DD-DD-DD-DD-DD-DD. Al no existir la necesidad de reenviar la trama a ninguna de las restantes interfaces, el switch lleva a cabo la función de filtrado descartando la trama.
- Existe una entrada en la tabla que asocia DD-DD-DD-DD-DD-DD con la interfaz $y \neq x$. En este caso, la trama tiene que ser reenviada al segmento de la LAN conectado a la interfaz y . El switch lleva a cabo su función de reenvío colocando la trama en un buffer de salida que precede a la interfaz y .

Aplicemos estas reglas al switch de la parte superior de la Figura 6.15 y a su tabla mostrada en la Figura 6.22. Suponga que una trama con la dirección de destino 62-FE-F7-11-89-A3 llega al switch desde la interfaz 1. El switch examina su tabla y ve que el destino está en el segmento de LAN conectado a la interfaz 1 (es decir, Ingeniería Eléctrica). Esto significa que la trama ya ha sido difundida en el segmento de LAN que contiene el destino. Por tanto, el switch filtra (es decir, descarta) la trama. Suponga ahora que una trama con la misma dirección de destino llega de la interfaz 2. De nuevo, el switch examina su tabla y ve que el destino está en la dirección de interfaz 1; por tanto, reenvía la trama al buffer de salida que precede a la interfaz 1. Con este ejemplo debería quedar claro que, siempre que la tabla de conmutación sea completa y precisa, el switch reenvía las tramas hacia los destinos sin llevar a cabo ninguna difusión.

En este sentido, un switch es “más inteligente” que un hub. Pero, ¿cómo se configura la tabla de conmutación? ¿Existen equivalentes para la capa de enlace de los protocolos de enrutamiento de la capa de red? ¿O tiene un administrador sobrecargado de trabajo que configurar manualmente la tabla de conmutación?

Auto-aprendizaje

Los switches tienen la fantástica propiedad (especialmente para los administradores de redes sobrecargados de trabajo) de que su tabla se construye de forma automática, dinámica y autónoma, sin intervención de un administrador de red ni de ningún protocolo de configuración. En otras palabras, los switches poseen la característica de **auto-aprendizaje**. Esta capacidad se lleva cabo de la forma siguiente:

1. Inicialmente, la tabla de conmutación está vacía
2. Para cada trama entrante recibida en una interfaz, el switch almacena en su tabla (1) la dirección MAC especificada en el *campo dirección de origen* de la trama, (2) la interfaz de la que procede la trama y (3) la hora actual. De esta forma, el switch registra en su tabla el segmento de la LAN en la que reside el emisor. Si todos los hosts de la LAN terminan enviando una trama, entonces todos los host terminarán estando registrados en la tabla.
3. El switch borra una dirección de la tabla si no se recibe ninguna trama con esa dirección como dirección de origen transcurrido un cierto periodo de tiempo (el **tiempo de envejecimiento**). De esta forma, si un PC es sustituido por otro (con un adaptador diferente), la dirección MAC del PC original será eliminada de la tabla de conmutación.

Examinemos la propiedad de auto-aprendizaje del switch superior de la Figura 6.15 y su tabla de conmutación correspondiente, mostrada en la Figura 6.22. Suponga que a las 9:39 una trama con la dirección de origen 01-12-23-34-45-56 llega procedente de la interfaz 2. Suponga también que esa dirección no está incluida en la tabla de conmutación. Entonces el switch añade una nueva entrada, como se muestra en la Figura 6.23.

Continuando con el mismo ejemplo, suponga que el tiempo de envejecimiento para este switch es de 60 minutos, y que entre las 9:32 y las 10:32 no le llega ninguna trama con la

Dirección	Interfaz	Hora
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....

Figura 6.22 ♦ Parte de la tabla de conmutación del switch superior de la Figura 6.15.

Dirección	Interfaz	Hora
01-12-23-34-45-56	2	9:39
62-FE-F7-11-89-A3	1	9:32
7C-BA-B2-B4-91-10	3	9:36
....

Figura 6.23 ♦ El switch aprende la ubicación de un adaptador con la dirección 01-12-23-34-45-56.

dirección de origen 62-FE-F7-11-89-A3. Entonces, a las 10:32 el switch eliminará esta dirección de su tabla.

Los switches son **dispositivos plug-and-play** porque no requieren intervención ni de un administrador de redes ni de los usuarios. Un administrador de redes que desee instalar un switch no tiene que hacer nada más que conectar los segmentos de la LAN a las interfaces del switch. El administrador no tiene que configurar las tablas de conmutación en el momento de la instalación ni cuando se elimina un host de uno de los segmentos de la LAN. Los switches también permiten la comunicación *full-duplex*, lo que significa que cualquier interfaz del switch puede enviar y recibir al mismo tiempo.

Propiedades de la conmutación de la capa de enlace

Una vez que hemos descrito el funcionamiento básico de un switch de la capa de enlace, vamos a pasar a ver sus características y propiedades. Podemos identificar varias ventajas de utilizar switches, en lugar de enlaces de difusión como las topologías de bus o basadas en hubs.

- *Eliminación de las colisiones.* En una red LAN construida con switches (y sin hubs) no se desperdicia ancho de banda a causa de las colisiones. Los switches almacenan las tramas en un buffer y nunca transmiten más de una trama a un segmento simultáneamente. Al igual que con los routers, la tasa máxima de transferencia agregada de un switch es la suma de todas las tasas de las interfaces del switch. Por tanto, los switches proporcionan una mejora significativa de rendimiento respecto al de las redes LAN con enlaces de difusión.
- *Enlaces heterogéneos.* Dado que un switch aísla un enlace de otro, los distintos enlaces de una LAN pueden operar a velocidades diferentes y pueden utilizar diferentes medios físicos. Por ejemplo, el switch superior de la Figura 6.15 puede tener tres enlaces de cobre 1000BASE-T a 1 Gbps, dos enlaces de fibra 100BASE-FX a 100 Mbps y un enlace de cobre 100BASE-T. Por tanto, un switch es ideal para combinar equipos heredados con equipos nuevos.
- *Administración.* Además de proporcionar una seguridad mejorada (véase el recuadro dedicado a la seguridad), un switch también facilita las tareas de gestión de la red. Por ejemplo, si un adaptador de red funciona mal y envía continuamente tramas Ethernet, un switch puede detectar el problema y desconectar internamente el adaptador que está funcionando incorrectamente. Con esta característica, el administrador de la red no tiene que levantarse de la cama y conducir hasta la oficina para corregir el problema. Del mismo modo, un corte en un cable solo desconecta al host que está usando el cable cortado para conectarse al switch. En la época del cable coaxial, los administradores de red pasaban muchas horas “recorriendo las líneas” (o dicho de forma más precisa, “arrastrándose por el suelo”) hasta localizar el cable roto que había hecho que se cayera toda la red. Los switches también recopilan estadísticas acerca del uso del ancho de banda, de las tasas de colisión y de los tipos de tráfico, y ponen esta información a disposición

del administrador de la red. Esta información puede emplearse para depurar y corregir problemas y para planificar cómo deberá evolucionar la red LAN en el futuro. Los investigadores están explorando la adición de todavía más funcionalidades de gestión a las redes LAN Ethernet en implantaciones de prototipos [Casado 2007; Koponen 2011].

Switches frente a routers

Como hemos visto en el Capítulo 4, los routers son dispositivos de conmutación de paquetes de almacenamiento y reenvío que reenvían los paquetes utilizando direcciones de la capa de red. Aunque un switch también es un dispositivo de conmutación de paquetes de almacenamiento y reenvío, es fundamentalmente diferente de un router porque reenvía los paquetes utilizando direcciones MAC. Mientras que un router es un dispositivo de conmutación de paquetes de la capa 3, un switch es un dispositivo de conmutación de paquetes de la capa 2. Sin embargo, recuerde de la Sección 4.4 que los switches modernos que utilizan el paradigma “correspondencia-acción” pueden usarse tanto para reenviar una trama de la capa 2 según la dirección MAC de destino como un datagrama de la capa 3 usando la dirección IP de destino del datagrama. De hecho, vimos que los switches que utilizan el estándar OpenFlow pueden llevar a cabo un reenvío generalizado de los paquetes, basándose en hasta once campos diferentes de las cabeceras de trama, de datagrama y de la capa de transporte.

Aunque los switches y los routers son fundamentalmente diferentes, los administradores de red a menudo tienen que elegir entre ellos a la hora de instalar un dispositivo de interconexión. Por ejemplo, en la red de la Figura 6.15 el administrador de red podría haber utilizado fácilmente un router en lugar de un switch para conectar las redes LAN departamentales, los servidores y el router de pasarela de Internet. De hecho, un router permitiría las comunicaciones entre departamentos sin crear colisiones. Puesto que tanto los switches como los routers son candidatos como dispositivos de interconexión, ¿cuáles son los pros y los contras de cada uno de ellos?

En primer lugar vamos a ocuparnos de los pros y los contras de los switches. Como hemos mencionado anteriormente, los switches son dispositivos plug-and-play, una propiedad muy apreciada por todos los administradores de red del mundo sobrecargados de trabajo. Los switches también ofrecen tasas de filtrado y reenvío relativamente altas (como se muestra en la Figura 6.24,

SEGURIDAD

HUSMEANDO EN UNA LAN CONMUTADA: ENVENENAMIENTO DE UN SWITCH

Cuando un host está conectado a un switch, normalmente solo recibe las tramas que le están siendo enviadas de forma explícita. Por ejemplo, considere la red LAN conmutada de la Figura 6.17. Cuando el host A envía una trama al host B y existe una entrada para el host B en la tabla del conmutador, este reenviará la trama únicamente al host B. Si el nodo C está ejecutando un programa husmeador (*sniffer*) no podrá husmear esta trama de A a B. Por tanto, en una LAN conmutada (en contraste con un entorno de enlaces de difusión como las redes LAN 802.11 o las redes LAN Ethernet basadas en hub) es más difícil para un atacante husmear las tramas. Sin embargo, dado que los switches difunden las tramas que tienen direcciones de destino que no están almacenadas en sus tablas de conmutación, el programa sniffer de C puede todavía husmear algunas tramas que no están explícitamente dirigidas a C. Además, un programa sniffer podrá husmear todas las tramas Ethernet de difusión que tengan la dirección de destino de difusión FF–FF–FF–FF–FF–FF. Un ataque bien conocido contra un switch, que recibe el nombre de **envenenamiento de switch**, consiste en enviar toneladas de paquetes al switch con muchas direcciones MAC de origen falsas diferentes, que llenarán la tabla de conmutación con entradas falsas y no dejarán espacio para las direcciones MAC de los nodos legítimos. Esto hace que el switch difunda la mayor parte de las tramas, las cuales pueden ser seleccionadas por el husmeador [Skoudis 2006]. Dado que este ataque es bastante complejo incluso para un atacante avanzado, los switches son significativamente menos vulnerables a los sniffers que las redes LAN inalámbricas y basadas en hubs.

los switches tienen que procesar las tramas solo hasta la capa 2, mientras que los routers tienen que procesar los datagramas hasta la capa 3). Por otro lado, para impedir los ciclos de las tramas de difusión, la topología activa de una red commutada está restringida a un árbol de recubrimiento. Además, una red commutada grande requerirá tablas ARP grandes en los hosts y routers y generará una cantidad de procesamiento y tráfico ARP sustancial. Los switches tampoco ofrecen ninguna protección frente a las tormentas de difusión (si un host está descontrolado y transmite un flujo de tramas Ethernet de difusión sin fin, los switches reenviarán todas esas tramas, haciendo que toda la red colapse).

Consideremos ahora los pros y los contras de los routers. Puesto que frecuentemente el direccionamiento de red es jerárquico (y no plano, como el direccionamiento MAC), normalmente los paquetes no seguirán ciclos a través de los routers incluso cuando la red tenga rutas redundantes. (Sin embargo, los paquetes pueden seguir ciclos cuando las tablas del router están mal configuradas; pero como hemos estudiado en el Capítulo 4, IP utiliza un campo especial de la cabecera del datagrama para limitar estos ciclos.) Por tanto, los paquetes no están restringidos a un árbol de recubrimiento y pueden utilizar la mejor ruta entre el origen y el destino. Dado que los routers no tienen la restricción del árbol de recubrimiento, han permitido que Internet haya sido creada con una topología rica que incluye, por ejemplo, múltiples enlaces activos entre Europa y América del Norte. Otra funcionalidad de los routers es que proporcionan protección mediante cortafuegos frente a las tormentas de difusión de la capa 2. Quizá el inconveniente más significativo de los routers es que no son dispositivos plug-and-play (ellos, y los hosts conectados a ellos, necesitan que sus direcciones IP sean configuradas). Además, los routers suelen tener un tiempo de procesamiento por paquete mayor que los switches, ya que tienen que procesar campos hasta la capa 3. Por último, en inglés existen dos formas diferentes de pronunciar la palabra router (“rootor” o “rowter”), y la gente pierde mucho tiempo discutiendo acerca de la pronunciación apropiada [Perlman 1999].

Dado que tanto los switches como los routers tienen sus ventajas y sus inconvenientes (como se resume en la Tabla 6.1), entonces ¿cuándo debe utilizar una red institucional (por ejemplo, una red de un campus universitario o una red corporativa) switches y cuándo routers? Normalmente, las redes pequeñas que constan de unos pocos cientos de hosts tienen pocos segmentos de LAN. Los switches son suficientes para estas redes pequeñas, ya que localizan el tráfico y aumentan la tasa de transferencia agregada sin necesidad de configurar direcciones IP. Pero las redes de mayor tamaño que constan de miles de hosts suelen incluir routers dentro de la red (además de switches). Los routers proporcionan un aislamiento más robusto del tráfico, controlan las tormentas de difusión y utilizan rutas más “inteligentes” entre los hosts de la red.

Para ver una exposición sobre los pros y los contras de las redes commutadas y enrutadas, así como un estudio de cómo la tecnología LAN commutada puede ampliarse para acomodar dos órdenes de magnitud más de hosts que las redes Ethernet actuales, consulte [Meyers 2004; Kim 2008].

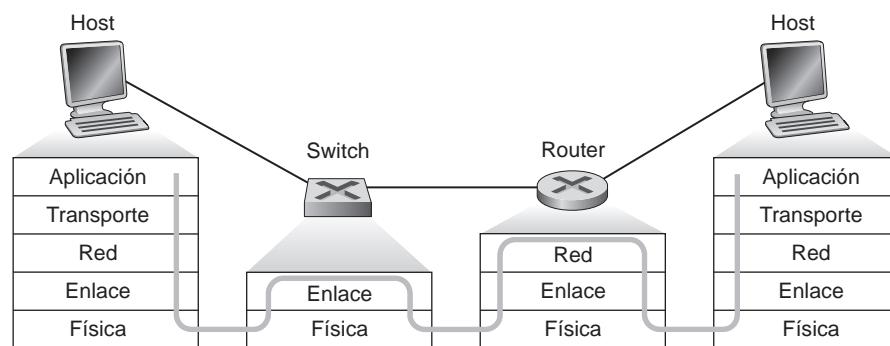


Figura 6.24 ♦ Procesamiento de paquetes en switches, routers y hosts.

	Hubs	Routers	Switches
Aislamiento del tráfico	No	Sí	Sí
Plug and play	Sí	No	Sí
Enrutamiento óptimo	No	Sí	No

Tabla 6.1 ♦ Comparación de funcionalidades típicas de dispositivos de interconexión populares.

6.4.4 Redes de área local virtuales (VLAN)

En nuestra explicación anterior de la Figura 6.15 hemos mencionado que las redes LAN institucionales modernas suelen estar configuradas de forma jerárquica, teniendo cada grupo de trabajo (departamento) su propia red LAN conmutada conectada a las redes LAN conmutadas de los otros grupos a través de una jerarquía de conmutadores. Aunque una configuración de este tipo funciona bien en un mundo ideal, el mundo real está bastante alejado del ideal.

En la configuración de la Figura 6.15 podemos identificar tres desventajas:

- *Falta de aislamiento del tráfico.* Aunque la jerarquía localiza el tráfico del grupo dentro de un mismo switch, el tráfico de difusión (por ejemplo, las tramas que transportan los mensajes ARP y DHCP o las tramas cuyo destino todavía no ha sido aprendido por un switch con auto-aprendizaje) tienen que atravesar toda la red institucional. Limitar el ámbito del tráfico de difusión mejoraría el rendimiento de la LAN. Quizá más importante, sería también deseable limitar el tráfico de difusión de la LAN por razones de seguridad y confidencialidad. Por ejemplo, si un grupo contiene al equipo de dirección de la empresa y otro grupo tiene empleados descontentos que ejecutan *sniffers* de paquetes Wireshark, probablemente el administrador de la red preferirá que el tráfico del equipo de dirección nunca llegue a los hosts de los empleados. Este tipo de aislamiento podría proporcionarse sustituyendo el switch central de la Figura 6.15 por un router. Enseguida veremos que este aislamiento también se puede conseguir a través de una solución conmutada (capa 2).
- *Uso ineficiente de los switches.* Si en lugar de tres grupos la institución tiene 10 grupos, entonces se necesitarían 10 switches de primer nivel. Si cada uno de los grupos es pequeño (por ejemplo, están formados por menos de 10 personas), entonces un único switch de 96 puertos sería lo suficientemente grande como para acomodar a todo el mundo, pero este único switch no proporcionaría la funcionalidad de aislamiento del tráfico.
- *Gestión de los usuarios.* Si un empleado se mueve entre grupos, el cableado físico debe modificarse para conectar al empleado a un switch diferente de la Figura 6.15. Los empleados que pertenecen a dos grupos constituyen incluso un problema mayor.

Afortunadamente, cada una de estas desventajas puede ser abordada por un switch compatible con **redes de área local virtuales (VLAN, Virtual Local Area Network)**. Como su nombre sugiere, un switch compatible con redes VLAN permite definir múltiples redes de área local *virtuales* sobre una única infraestructura de red de área local *física*. Los hosts de una VLAN se comunican entre sí como si solo ellos (y ningún otro host) estuvieran conectados al switch. En una VLAN basada en puertos, el administrador de la red divide los puertos (interfaces) del switch en grupos. Cada grupo constituye una VLAN, con los puertos de cada VLAN formando un dominio de difusión (es decir, el tráfico de difusión de un puerto solo puede llegar a los demás puertos del grupo). La Figura 6.25 muestra un único switch con 16 puertos. Los puertos 2 a 8 pertenecen a la VLAN IE, y los puertos 9 a 15 pertenecen a la VLAN CC (los puertos 1 y 16 no están asignados). Esta VLAN resuelve todas las dificultades mencionadas anteriormente: las tramas de las VLAN IE y CC están aisladas entre

sí, los dos switches de la Figura 6.15 se han sustituido por un único switch y si el usuario del puerto 8 del switch se une al departamento CC, el operador de red simplemente tendrá que reconfigurar el software de la VLAN de modo que el puerto 8 ahora esté asociado con la VLAN CC. Es fácil imaginar cómo se configura y funciona el switch para redes VLAN: el administrador de la red declara que un puerto pertenece a una determinada VLAN (los puertos no declarados pertenecen a una VLAN predeterminada) utilizando un software de gestión de switches; en el switch se mantiene una tabla de correspondencias entre puertos y redes VLAN y el hardware del switch solo entrega tramas entre puertos que pertenecen a la misma VLAN.

Pero a causa del completo aislamiento de las dos redes VLAN hemos introducido una nueva dificultad: ¿cómo puede enviarse el tráfico del departamento IE al departamento CC? Una forma de resolver esto sería conectando un puerto del conmutador VLAN (por ejemplo, el puerto 1 en la Figura 6.25) a un router externo y configurando dicho puerto para que pertenezca tanto a la VLAN IE como a la VLAN CC. En este caso, incluso aunque los departamentos IE y CC comparten el mismo switch físico, la configuración lógica sería como si dichos departamentos tuvieran switches separados conectados a través de un router. Un datagrama IP enviado desde el departamento IE al departamento CC primero atravesaría la VLAN IE para llegar al router y luego sería reenviado por el router por la VLAN CC hasta el host de CC. Afortunadamente, los fabricantes de switches hacen que dicha tarea de configuración resulte sencilla para los administradores de red, incorporando en un único dispositivo un switch VLAN y un router, con lo que no es necesario utilizar un router externo separado. En uno de los problemas de repaso del final del capítulo se examina este escenario más en detalle.

Volviendo de nuevo a la Figura 6.15, suponga ahora que en lugar de tener un departamento de Ingeniería de Computadoras separado algunos de los académicos de IE y CC están alojados en edificios diferentes, donde (¡por supuesto!) necesitan tener acceso a la red y (¡por supuesto también!) desean formar parte de la VLAN de su departamento. La Figura 6.26 muestra un segundo switch de 8 puertos, en el que los puertos se han definido como pertenecientes a la VLAN IE o a la VLAN CC, según sea necesario. Pero, ¿cómo deben interconectarse estos dos switches? Una solución fácil sería definir un puerto que perteneciera a la VLAN CC en cada conmutador (y lo mismo para la VLAN IE) y conectar estos puertos entre sí, como se muestra en la Figura 6.26(a). Sin embargo, esta solución no es escalable, ya que N redes VLAN requerirían N puertos en cada switch simplemente para interconectar los dos switches.

Un método más escalable consiste en interconectar los switches VLAN utilizando la técnica conocida como **troncalización VLAN (VLAN Trunking)**. Con esta técnica, mostrada en la Figura 6.26(b), un puerto especial de cada switch (el puerto 16 en el switch de la izquierda y el puerto 1 en el de la derecha) se configura como un puerto troncal para interconectar los dos switches VLAN. El puerto troncal pertenece a todas las VLAN y las tramas enviadas a cualquier VLAN son reenviadas a

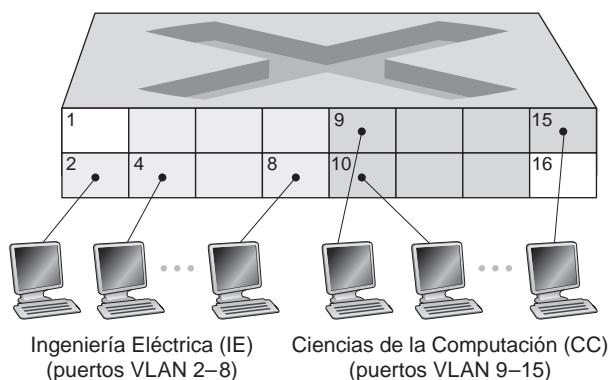


Figura 6.25 ♦ Un mismo conmutador con dos redes VLAN configuradas.

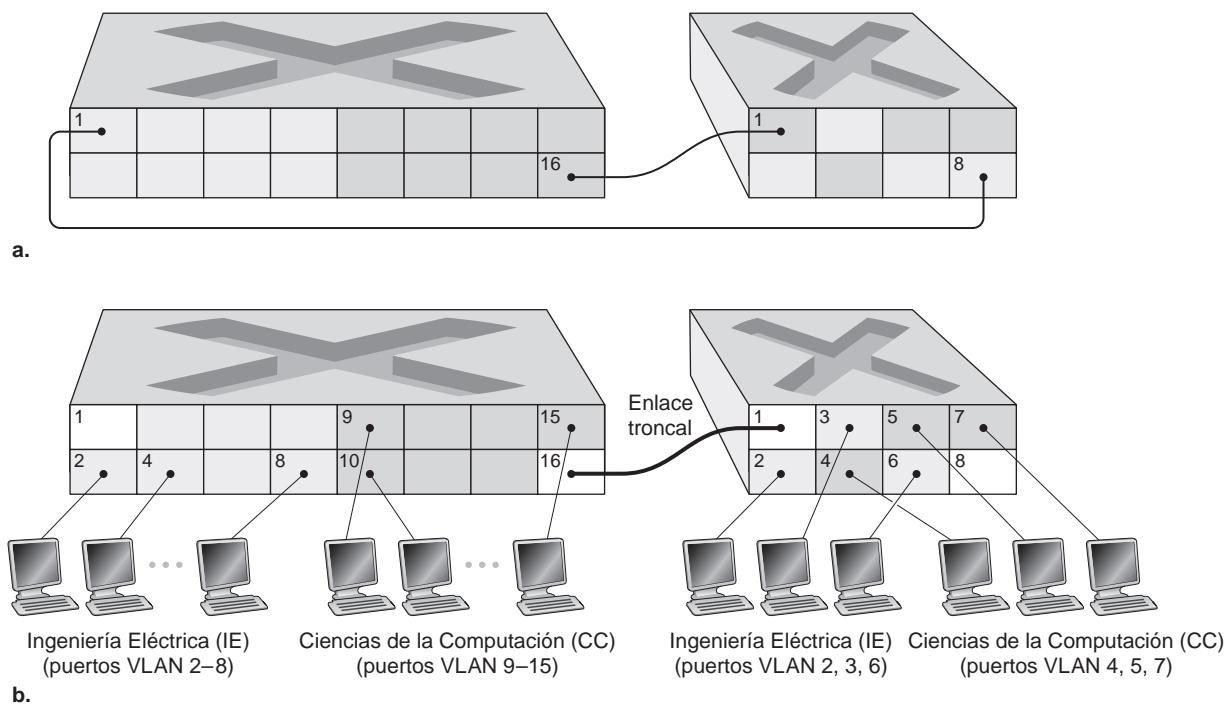


Figura 6.26 ♦ Conexión de dos switches VLAN con dos redes VLAN:
(a) dos cables (b) enlace troncal.

través del enlace troncal hacia el otro switch. Pero esta solución nos conduce a otra pregunta: ¿cómo sabe un switch que una trama que ha llegado a un puerto troncal pertenece a una VLAN concreta? El IEEE ha definido un formato de trama Ethernet ampliado, 802.1Q, para las tramas que atraviesan un enlace troncal VLAN. Como se muestra en la Figura 6.27, la trama 802.1Q está formada por la trama Ethernet estándar más una **etiqueta VLAN** de cuatro bytes añadida a la cabecera, que transporta la identidad de la VLAN a la que pertenece la trama. El switch del lado emisor de un enlace troncal VLAN añade la etiqueta VLAN a la trama, la cual es analizada y eliminada por el switch del lado receptor del enlace troncal. La etiqueta VLAN en sí consta de un campo Identificador de protocolo de etiquetado (TPID, Tag Protocol Identifier) de 2 bytes (que tiene un valor hexadecimal fijo de 81-00) y de un campo Información de control de etiquetado (Tag Control Information) de 2 bytes, que contiene un campo identificador de VLAN de 12 bits y un campo de prioridad de 3 bits, cuya finalidad es similar a la del campo TOS de los datagramas IP.

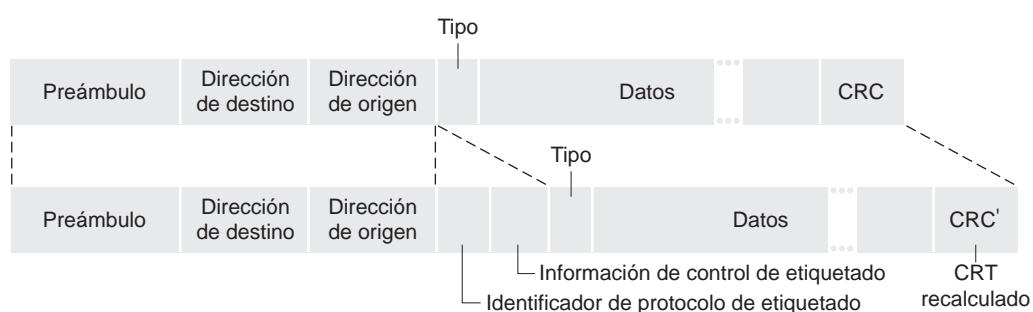


Figura 6.27 ♦ Trama Ethernet original (arriba), trama Ethernet con etiquetado 802.1Q (abajo).

En esta exposición hemos hablado muy brevemente de las redes VLAN y nos hemos centrado en las VLAN basadas en puertos. Debemos decir también que las redes VLAN se pueden definir de otras formas. En las VLAN basadas en direcciones MAC, el administrador de la red especifica el conjunto de direcciones MAC que pertenece a cada VLAN. Cuando un dispositivo se conecta a un puerto, el puerto se conecta a la VLAN apropiada basándose en la dirección MAC del dispositivo. Las redes VLAN también pueden definirse basándose en protocolos de la capa de red (como por ejemplo, IPv4, IPv6 o Appletalk) y en otros criterios. Las redes VLAN también pueden abarcar varios routers IP, lo que permite conectar islas de redes LAN con el fin de formar una única VLAN que podría abarcar todo el globo [Yu 2011]. Consulte el estándar 802.1Q [IEEE 802.1q 2005] para conocer más detalles.

6.5 Virtualización de enlaces: la red como una capa de enlace

Puesto que este capítulo está relacionado con los protocolos de la capa de enlace, y dado que estamos aproximándonos al final del capítulo, vamos a reflexionar acerca del modo en que ha evolucionado nuestra compresión acerca del término *enlace*. Al comenzar el capítulo, contemplábamos los enlaces como cables físicos que conectaban dos hosts que se comunicaban entre sí. Al estudiar los protocolos de acceso múltiple, hemos visto que podían conectarse varios hosts mediante un cable compartido y que el “cable” que conectaba esos hosts podía ser un espectro de radio u otro medio de comunicación. Esto nos llevó a considerar el enlace en forma algo más abstracta, más como un canal que como un cable. En nuestro estudio de las redes LAN Ethernet (Figura 6.15) hemos visto que los medios de interconexión podían ser, de hecho, una infraestructura commutada bastante compleja. Sin embargo, a todo lo largo de esta evolución, los propios hosts mantenían la visión de que el medio de interconexión era simplemente un canal de la capa enlace que conectaba dos o más hosts. Vimos, por ejemplo, que un host Ethernet podía ser afortunadamente inconsciente de si estaba conectado a los otros hosts de la LAN mediante un único segmento LAN de corto alcance (Figura 6.17), o por una LAN commutada geográficamente dispersa (Figura 6.15), o mediante una VLAN (Figura 6.26).

En el caso de una conexión mediante un módem telefónico entre dos hosts, el enlace que conecta los dos hosts es realmente la red telefónica: una red global de telecomunicaciones, lógicamente separada, con sus propios commutadores, enlaces y pilas de protocolos para la transferencia de datos y la señalización. Sin embargo, desde el punto de vista de la capa de enlace de Internet, la conexión de acceso telefónico a través de la red de telefonía es contemplada como un simple “cable”. En este sentido, Internet virtualiza la red telefónica, viéndola como una tecnología de la capa de enlace que proporciona conectividad de dicha capa entre dos hosts de Internet. Recordemos, de nuestro análisis de las redes solapadas en el Capítulo 2, que de forma similar las redes solapadas ven Internet como un medio de proporcionar conectividad entre los nodos solapados, buscando un solapamiento de Internet de la misma manera que Internet solapa la red telefónica.

En esta sección vamos a abordar las redes MPLS (*Multiprotocol Label Switching*, Comutación de etiquetas multiprotocolo). A diferencia de la red telefónica de commutación de circuitos, MPLS es una red de circuitos virtuales de commutación de paquetes de pleno derecho. Tiene sus propios formatos de paquete y comportamientos de reenvío. Por tanto, desde el punto de vista pedagógico, el análisis de MPLS encaja bien en un estudio de la capa de red o de la capa de enlace. Sin embargo, desde el punto de vista de Internet podemos considerar MPLS, al igual que la red telefónica y las redes Ethernet commutadas, como una tecnología de la capa de enlace que sirve para interconectar dispositivos IP. Por tanto, hemos incluido MPLS en nuestro estudio de la capa de enlace. Las redes Frame-Relay y ATM también pueden emplearse para interconectar dispositivos IP, aunque representan una tecnología algo más antigua (aunque todavía con implantación), de modo que no cubriremos esas redes aquí. Puede ver más detalles en el libro de [Goralski 1999]. Nuestro tratamiento de MPLS será necesariamente breve, ya que podrían escribirse (y se han escrito) libros completos sobre estas redes. Le recomendamos que lea [Davie 2000] para conocer más detalles sobre MPLS.

Aquí nos centraremos principalmente en el modo en que los servidores MPLS se interconectan con los dispositivos IP, aunque también profundizaremos algo más en las tecnologías subyacentes.

6.5.1 Conmutación de etiquetas multiprotocolo (MPLS)

La Conmutación de etiquetas multiprotocolo (MPLS) ha evolucionado a partir de una serie de desarrollos industriales que tuvieron lugar a mediados y finales de la década de 1990 y que buscaban mejorar la velocidad de reenvío de los routers IP, adoptando un concepto clave del mundo de las redes de circuitos virtuales: una etiqueta de longitud fija. El objetivo no era abandonar la infraestructura de reenvío de datagramas IP basada en el destino, sustituyéndola por otra basada en etiquetas de longitud fija y circuitos virtuales, sino expandir la infraestructura existente etiquetando selectivamente los datagramas y permitiendo a los routers reenviar esos datagramas basándose en etiquetas de longitud fija (en lugar de en direcciones IP de destino), siempre que fuera posible. Es importante observar que estas técnicas funcionan mano a mano con IP, utilizando el direccionamiento y el enrutamiento IP. El IETF unificó estos esfuerzos mediante el protocolo MPLS [RFC 3031, RFC 3032], consiguiendo así mezclar de forma efectiva las técnicas de circuitos virtuales en una red de datagramas enrutados.

Comencemos nuestro estudio sobre MPLS considerando el formato de una trama de la capa de enlace gestionada por un router compatible con MPLS. La Figura 6.28 muestra que una trama de la capa de enlace transmitida entre dispositivos compatibles con MPLS tiene una cabecera MPLS pequeña, que se añade entre la cabecera de la capa 2 (por ejemplo, Ethernet) y la cabecera de la capa 3 (es decir, IP). El documento RFC 3032 define el formato de la cabecera MPLS para tales enlaces; en otros RFC se definen también las cabeceras para las redes ATM y Frame-Relay. Entre los campos de la cabecera MPLS se encuentran la etiqueta, 3 bits reservados para su uso experimental, un único bit S que se utiliza para indicar el final de una serie de cabeceras MPLS “apiladas” (un tema avanzado del que no hablaremos aquí) y un campo de tiempo de vida.

Es evidente de manera inmediata, a partir de la Figura 6.28, que una trama ampliada MPLS solo se puede intercambiar entre routers compatibles con MPLS (dado que un router no compatible con MPLS se quedaría bastante confundido al encontrar una cabecera MPLS donde esperaba encontrar la cabecera IP). Los routers compatibles con MPLS se suelen denominar **routers de conmutación de etiquetas**, ya que reenvían las tramas MPLS buscando la etiqueta MPLS en su tabla de reenvío y luego pasando inmediatamente el datagrama a la interfaz de salida apropiada. Por tanto, el router compatible con MPLS *no* necesita extraer la dirección IP de destino y realizar una búsqueda del prefijo con la coincidencia más larga dentro de la tabla de reenvío. Pero, ¿cómo sabe un router si su vecino es compatible con MPLS y cómo sabe qué etiqueta asociar con la dirección IP de destino indicada? Para responder a estas preguntas, necesitamos examinar la interacción entre un grupo de routers compatibles con MPLS.

En el ejemplo de la Figura 6.29 los routers R1 a R4 son compatibles con MPLS. Los routers R5 y R6 son routers IP estándar. R1 ha anunciado a R2 y a R3 que él (R1) puede enrutar hacia el destino A y que una trama recibida con una etiqueta MPLS igual a 6 será reenviada al destino A. El router R3 ha anunciado al router R4 que puede realizar el enrutamiento hacia los destinos A y D, y

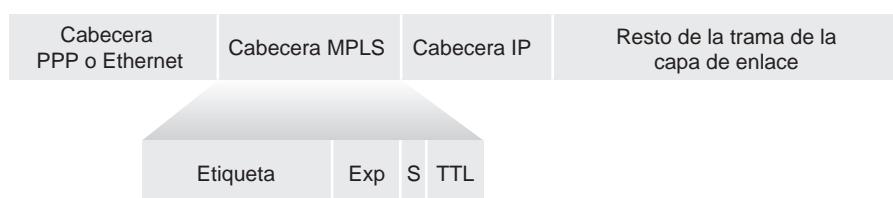


Figura 6.28 ♦ Cabecera MPLS: se localiza entre las cabeceras de la capa de enlace y de la capa de red.

que las tramas entrantes con etiquetas MPLS de valor 10 y 12, respectivamente, serán conmutadas hacia esos destinos. El router R2 también ha anunciado al router R4 que puede alcanzar el destino A y que una trama recibida con la etiqueta MPLS de valor 8 será conmutada hacia A. Observe que ahora el router R4 se encuentra en la interesante situación de tener dos rutas MPLS para llegar a A: a través de la interfaz 0 con etiqueta MPLS saliente igual a 10, y a través de la interfaz 1 con etiqueta MPLS igual a 8. El cuadro general de la estructura de red mostrada en la Figura 6.29 es que los dispositivos IP R5, R6, A y D están interconectados a través de una infraestructura MPLS (los routers compatibles con MPLS R1, R2, R3 y R4) de forma muy similar a como pueden conectarse entre sí diversos dispositivos IP mediante una red ATM o una LAN conmutada. Y al igual que sucede con una red ATM o una LAN conmutada, los routers R1 a R4 compatibles con MPLS se encargan de realizar esa conmutación *sin ni siquiera tocar la cabecera IP de los paquetes*.

En nuestra exposición anterior no hemos especificado el protocolo concreto que se utiliza para distribuir las etiquetas entre los routers compatibles con MPLS, ya que los detalles de este tipo de señalización caen fuera del alcance del libro. Sin embargo, tenemos que dejar constancia de que el grupo de trabajo de IETF dedicado a MPLS ha especificado en [RFC 3468] que el centro de sus esfuerzos dentro del campo de la señalización de MPLS será una extensión del protocolo RSVP, conocida como RSVP-TE [RFC 3209]. Tampoco hemos explicado cómo calcula MPLS en la práctica las rutas para los paquetes entre routers compatibles con MPLS, ni cómo recopila la información del estado de los enlaces (por ejemplo, la cantidad de ancho de banda del enlace no reservada por MPLS) que hay que utilizar en estos cálculos de rutas. Los algoritmos de enrutamiento existentes basados en el estado de los enlaces (por ejemplo, OSPF) han sido ampliados para inundar con estas informaciones los routers compatibles con MPLS. Merece la pena resaltar que los propios algoritmos de cálculo de las rutas no están estandarizados, siendo actualmente específicos del fabricante.

Hasta ahora, el énfasis de nuestro estudio sobre MPLS se ha centrado en el hecho de que MPLS realiza la conmutación basándose en etiquetas, sin necesidad de considerar la dirección IP de un paquete. Las verdaderas ventajas de MPLS y la razón del actual interés en este tipo de tecnología radica, sin embargo, no en los potenciales aumentos de las velocidades de conmutación, sino más bien en las nuevas capacidades de gestión del tráfico que MPLS posibilita. Como ya hemos indicado, R4 dispone de *dos* rutas MPLS hacia A. Si el reenvío se realizara más arriba, en la capa IP, basándose en la dirección IP, los protocolos de enrutamiento IP que hemos estudiado en el Capítulo

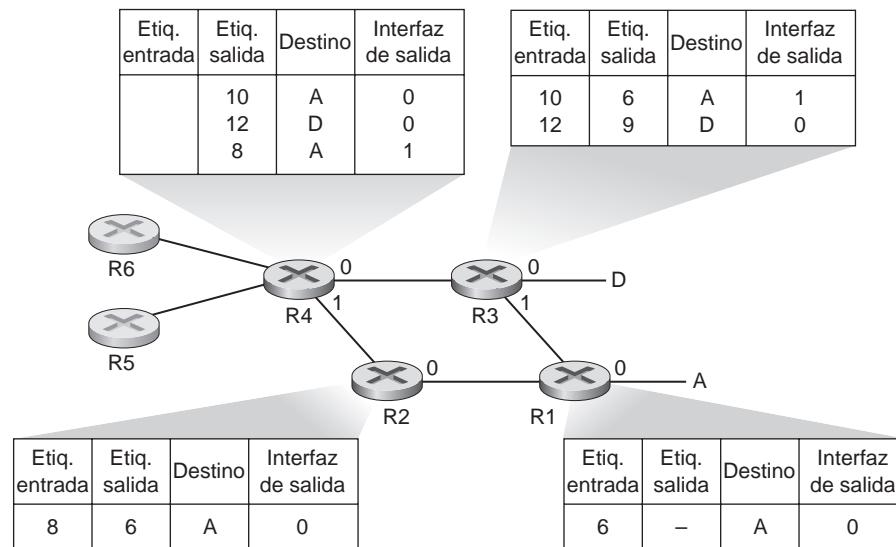


Figura 6.29 ♦ Reenvío mejorado mediante MPLS.

tulo 5 especificarían solamente una única ruta de coste mínimo hacia A. Por tanto, MPLS proporciona la capacidad de reenviar paquetes a través de rutas que no serían posibles utilizando los protocolos de enrutamiento IP estándar. Este es un tipo simple de **ingeniería de tráfico** utilizando MPLS [RFC 3346; RFC 3272; RFC 2702; Xiao 2000], mediante el que un operador de red puede anular el enrutamiento IP normal y forzar a que parte del tráfico dirigido hacia un cierto destino tome una determinada ruta, mientras que el resto del tráfico dirigido a ese mismo destino sigue una ruta distinta (por razones de política, de rendimiento o de algún otro tipo).

También se puede utilizar MPLS para muchos otros propósitos. Puede emplearse para llevar a cabo una restauración rápida de las rutas de reenvío de MPLS; por ejemplo, para volver a enrutar el tráfico a través de una ruta de reserva precalculada como respuesta a un fallo de un enlace [Kar 2000; Huang 2002; RFC 3469]. Por último, digamos que MPLS puede utilizarse (y de hecho se ha utilizado) para implementar las denominadas **redes privadas virtuales (VPN, Virtual Private Network)**. Al implementar una VPN para un cliente, un ISP utiliza su red compatible con MPLS para conectar entre sí las diversas redes del cliente. MPLS puede emplearse para aislar tanto los recursos como el direccionamiento empleados por la VPN del cliente con respecto a los de otros usuarios que también tengan que atravesar la red del ISP; consulte [DeClercq 2002] para ver más detalles.

Nuestra exposición acerca de MPLS ha sido necesariamente breve y animamos al lector a consultar las referencias mencionadas. Hay que destacar que, con tantos posibles usos de MPLS, este protocolo parece estar convirtiéndose rápidamente en la “navaja multiusos” de la ingeniería de tráfico en Internet.

6.6 Redes para centros de datos

En los últimos años, empresas de Internet como Google, Microsoft, Facebook y Amazon (así como sus competidoras en Asia y Europa) han construido inmensos centros de datos, cada uno de los cuales alberga decenas o centenares de miles de hosts, y que soportan simultáneamente muchas aplicaciones diferentes en la nube (por ejemplo, búsquedas, correo electrónico, redes sociales y comercio electrónico). Cada centro de datos tiene su propia **red del centro de datos**, que interconecta entre sí sus hosts y conecta el centro de datos con Internet. En esta sección vamos a realizar una breve introducción a las redes para centros de datos para aplicaciones en la nube.

El coste de un gran centro de datos es enorme, de más de 12 millones de \$ por mes para un centro de datos con 100.000 hosts [Greenberg 2009a]. De estos costes, aproximadamente un 45% puede atribuirse a los propios hosts (que necesitan ser sustituidos cada 3 o 4 años); el 25%, a infraestructura, incluyendo transformadores, sistemas de alimentación ininterrumpida (SAI), generadores para apagones de larga duración y sistemas de refrigeración; 15% son los costes de la factura eléctrica, por la energía consumida y el 15% restante corresponde a las redes, incluyendo los equipos de red (switches, routers y equilibradores de carga), enlaces externos y costes asociados al tráfico de tránsito. (En estos porcentajes, los costes de los equipos se amortizan, de modo que se aplica una métrica de costes común a las adquisiciones y a los gastos corrientes, como la electricidad.) Aunque el de las redes no es el mayor coste, la innovación en este terreno es clave para reducir el coste global y maximizar el rendimiento [Greenberg 2009a].

Las abejas obreras en un centro de datos son los hosts: sirven contenido (por ejemplo, páginas web y vídeos), almacenan correos electrónicos y documentos y realizan, colectivamente, cálculos masivamente distribuidos (por ejemplo, cálculos distribuidos de índices para motores de búsqueda). Los hosts de los centros de datos, a los que se denomina **servidores blade** (cuchilla) y que recuerdan a las cajas de pizza, suele ser hosts de bajo coste, que incorporan la CPU, memoria y almacenamiento en disco. Los hosts se apilan en bastidores (*racks*), soliendo tener cada bastidor entre 20 y 40 servidores blade. En la parte superior de cada bastidor hay un switch, denominado **switch TOR (Top of Rack, parte superior del bastidor)**, que interconecta los hosts entre sí y con otros switches del centro de datos. Para ser concretos, cada host del bastidor tiene una tarjeta de interfaz de red que está conectada con su switch TOR, y cada switch TOR tiene puertos adicionales que pueden conectarse a

otros switches. Los hosts actuales suelen tener conexiones Ethernet a 40 Gbps con sus switches TOR [Greenberg 2015]. A cada host se le asigna también su propia dirección IP, interna al centro de datos.

La red del centro de datos soporta dos tipos de tráfico: el tráfico que fluye entre los clientes externos y los hosts internos, y el tráfico que fluye entre los hosts internos. Para manejar los flujos entre los clientes externos y los hosts internos, la red del centro de datos incluye uno o más **routers de frontera**, que conectan la red del centro de datos con la Internet pública. La red del centro de datos, por tanto, interconecta los bastidores entre sí y con los routers de frontera. La Figura 6.30 muestra un ejemplo de red de centro de datos. El **diseño de redes para centros de datos**, que es el arte de diseñar la red de interconexión y los protocolos que conectan los bastidores entre sí y con los routers de frontera, ha pasado en los últimos años a ser un área de investigación importante dentro del campo de las redes de computadoras [Al-Fares 2008; Greenberg 2009a; Greenberg 2009b; Mysore 2009; Guo 2009; Wang 2010].

Equilibrado de carga

Un centro de datos en la nube, como los de Google o Microsoft, proporciona muchas aplicaciones de manera simultánea, como por ejemplo aplicaciones de búsqueda, de correo electrónico o de vídeo. Para soportar las solicitudes de los clientes externos, cada aplicación está asociada con una dirección IP pública visible, a la que los clientes envían sus solicitudes y de la que los clientes reciben sus respuestas. Dentro del centro de datos, las solicitudes externas se dirigen primero a un **equilibrador de carga** cuya misión consiste en distribuir las solicitudes entre los hosts, equilibrando la carga entre todos ellos en función de su carga actual. Un gran centro de datos tendrá, a menudo, varios equilibradores de carga, cada uno de ellos dedicado a un conjunto de aplicaciones específicas en la nube. A tales equilibradores de carga se les denomina a veces “comunicadores de la capa 4”, ya que toman decisiones basándose tanto en el número de puerto de destino (capa 4) como en la dirección IP de destino del paquete. Al recibir una solicitud para una aplicación concreta, el equilibrador de carga la reenvía a uno de los hosts encargados de gestionar esa aplicación. (El host puede, a su vez, invo-

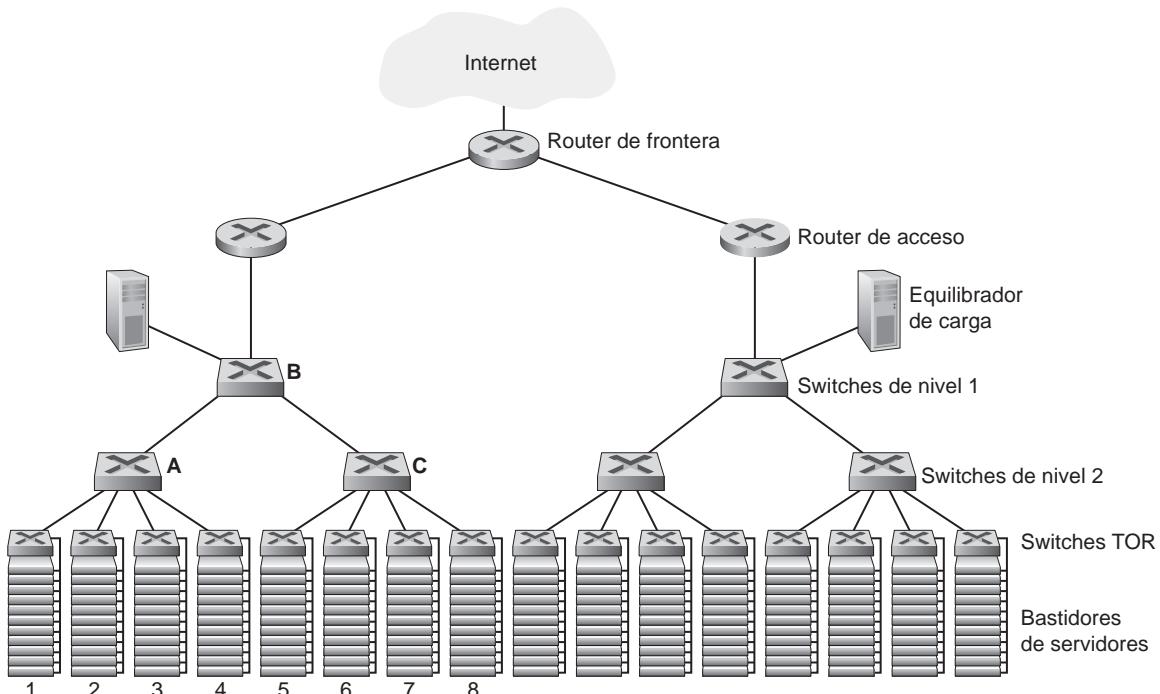


Figura 6.30 ♦ Una red de centro de datos con topología jerárquica.

car los servicios de otros hosts para que le ayuden a procesar la solicitud.) Cuando el host termina de procesar la solicitud, devuelve la respuesta al equilibrador de carga, que a su vez se la reenvía al cliente externo. El equilibrador no solo distribuye equitativamente la carga de trabajo entre los hosts, sino que también proporciona una función similar a NAT, traduciendo la dirección IP pública externa a la dirección IP interna del host apropiado, y luego realizando la traducción inversa para los paquetes que viajan en dirección contraria, hacia los clientes. Esto evita que los clientes contacten directamente con los hosts del centro de datos, lo que proporciona la ventaja de seguridad de ocultar la estructura interna de la red e impedir que los clientes interactúen directamente con los hosts.

Arquitectura jerárquica

Para un pequeño centro de datos que solo albergue unos pocos miles de hosts podría bastar una red simple, compuesta por un router de frontera, un equilibrador de carga y unas pocas decenas de bastidores, todos ellos interconectados mediante un único switch Ethernet. Pero para ampliar el centro a decenas o cientos de miles de hosts, los centros de datos emplean a menudo una jerarquía de routers y switches, como la topología mostrada en la Figura 6.30. En la parte superior de la jerarquía, el router de frontera está conectado con una serie de routers de acceso (en la Figura 6.30 solo se muestran dos, pero puede haber muchos más). Debajo de cada router de acceso hay tres niveles de switches. Cada router de acceso se conecta a un switch de nivel superior y cada switch de nivel superior se conecta a múltiples switches de segundo nivel y a un equilibrador de carga. Cada switch de segundo nivel se conecta, a su vez, con múltiples bastidores, a través de los switches TOR (switches de tercer nivel) de los propios bastidores. Todos los enlaces suelen usar Ethernet para sus protocolos de la capa física y de la capa de enlace, con una mezcla de cables de cobre y de fibra óptica. Con este tipo de diseño jerárquico, resulta posible ampliar un centro de datos hasta los cientos de miles de hosts.

Para un proveedor de aplicaciones en la nube resulta crítico proporcionar de modo continuo aplicaciones con una alta disponibilidad. Es por eso que los diseños de centros de datos también incluyen equipos de red redundantes y enlaces redundantes (estos recursos redundantes no se muestran en la Figura 6.30). Por ejemplo, cada switch TOR puede conectarse a dos switches de nivel 2, y cada router de acceso, switch de nivel 1 y switch de nivel 2 puede duplicarse e integrarse en el diseño [Cisco 2012; Greenberg 2009b]. En el diseño jerárquico de la Figura 6.30, observe que los hosts situados debajo de cada router de acceso forman una única subred. Para confinar el tráfico de difusión ARP, cada una de estas subredes se partitiona, a su vez, en subredes VLAN más pequeñas, cada una de las cuales está compuesta por unos pocos cientos de hosts [Greenberg 2009a].

Aunque la arquitectura jerárquica convencional que acabamos de describir resuelve el problema de la escala, presenta el problema de la *limitada capacidad de host a host* [Greenberg 2009b]. Para entender esta limitación, fíjemonos de nuevo en la Figura 6.30, y supongamos que cada host se conecta a su switch TOR a través de un enlace a 1 Gbps, mientras que los enlaces entre switches son enlaces Ethernet a 10 Gbps. Dos hosts del mismo bastidor siempre podrán comunicarse a la velocidad máxima de 1 Gbps, siendo el único factor limitante la velocidad de las tarjetas de interfaz de red del host. Sin embargo, si hay muchos flujos simultáneos en la red del centro de datos, la velocidad máxima entre dos hosts situados en diferentes bastidores puede ser muy inferior. Para entender el problema, considere un patrón de tráfico compuesto por 40 flujos simultáneos entre 40 pares de hosts situados en bastidores distintos. Específicamente, suponga que cada uno de los 10 hosts del bastidor 1 de la Figura 6.30 envía un flujo al host correspondiente del bastidor 5. Suponga también que, de modo similar, hay diez flujos simultáneos entre pares de hosts de los bastidores 2 y 6, diez flujos simultáneos entre los bastidores 3 y 7 y diez flujos simultáneos entre los bastidores 4 y 8. Si cada flujo comparte la capacidad de un enlace de modo equitativo con los otros flujos que estén atravesando el enlace, entonces a cada uno de los 40 flujos que atraviesan el enlace A-B de 10 Gbps (y también el enlace B-C de 10 Gbps) solo le corresponderán $10 \text{ Gbps} / 40 = 250 \text{ Mbps}$, que es significativamente menor que la velocidad de 1 Gbps de la tarjeta de interfaz de red. El problema se agrava aún más para los flujos entre hosts que necesiten subir más arriba, dentro de la jerarquía de enlaces. Para resolver esta limitación, una posible solución es implantar switches y routers de

velocidad más alta. Pero esto incrementaría significativamente el coste del centro de datos, porque los switches y routers con altas velocidades de puerto son muy caros.

Soportar un gran ancho de banda de comunicación host a host es importante, porque uno de los requisitos clave de los centros de datos es la flexibilidad en la ubicación de la capacidad de cómputo y de los servicios [Greenberg 2009b; Farrington 2010]. Por ejemplo, un motor de búsqueda en Internet a gran escala puede ejecutarse en miles de hosts distribuidos entre múltiples bastidores, con unas necesidades de ancho de banda significativas entre todos los pares de hosts. De forma similar, un servicio de computación en la nube como EC2 podría necesitar albergar las múltiples máquinas virtuales que componen el servicio de un cliente en los hosts físicos que dispongan de mayor capacidad, independientemente de su ubicación en el centro de datos. Si estos hosts físicos están distribuidos entre múltiples bastidores, cuellos de botella en la red como los que hemos descrito anteriormente podrían provocar un bajo rendimiento.

Tendencias en las redes para centros de datos

Para reducir el coste de los centros de datos, y para mejorar al mismo tiempo su retardo de respuesta y su tasa de transferencia, gigantes de Internet en el campo de servicios en la nube, como Google, Facebook, Amazon y Microsoft, están continuamente implantando nuevos diseños de redes para centros de datos. Aunque estos diseños son exclusivos de esas empresas, podemos de todos modos identificar varias tendencias importantes.

Una de esas tendencias es la de implantar nuevas arquitecturas de interconexión y protocolos de red que resuelvan los problemas de los diseños jerárquicos tradicionales. Una de dichas soluciones consiste en sustituir la jerarquía de switches y routers por una **topología completamente conectada** [Facebook 2014; Al-Fares 2008; Greenberg 2009b; Guo 2009], como la mostrada en la Figura 6.31. En este diseño, cada switch de nivel 1 se conecta a todos los switches de nivel 2, de modo que (1) el tráfico de host a host nunca necesita subir por encima de los niveles de switches y (2) con n switches de nivel 1, entre cualesquiera dos switches de nivel 2 existirán n rutas disjuntas. Un diseño de este tipo puede mejorar significativamente la capacidad host a host. Para comprobarlo, considere de nuevo nuestro ejemplo de los 40 flujos. La topología de la Figura 6.31 puede gestionar perfectamente ese patrón de flujo, ya que existen cuatro rutas diferentes entre el primer switch de nivel 2 y el segundo, las cuales proporcionan una capacidad agregada de 40 Gbps entre los dos primeros switches de nivel 2. Este tipo de diseño no solo alivia la limitación de la capacidad host a host, sino que también crea un entorno más flexible de cómputo y de prestación de servicios, en el que la comunicación entre cualesquiera dos bastidores que no estén conectados al mismo switch es lógicamente equivalente, independientemente de dónde estén ubicados en el centro de datos.

Otra tendencia importante es la de emplear centros de datos modulares (MDC, *Modular Data Center*) basados en contenedores [Youtube 2009; Waldrop 2007]. En un MDC, una fábrica construye un “minicentro de datos” dentro de un contenedor estándar de 12 metros y envía el contenedor a las instalaciones del centro de datos. Cada contenedor tiene unos pocos miles de hosts, apilados en decenas de bastidores, que están densamente empaquetados. En las instalaciones del centro de datos se interconectan múltiples contenedores entre sí y con Internet. Una vez implantado un contenedor prefabricado en un centro de datos a menudo resulta difícil realizar su mantenimiento. Por ello, cada contenedor está diseñado para sufrir una degradación suave del rendimiento: a medida que los componentes (servidores y switches) van fallando con el tiempo, el contenedor continúa funcionando, pero con un rendimiento menor. Una vez que ha fallado un gran número de componentes y el rendimiento cae por debajo de un cierto umbral, se elimina todo el contenedor y se lo sustituye por uno nuevo.

La creación de centros de datos a partir de contenedores plantea nuevos desafíos relacionados con la red. En un MDC hay dos tipos de redes: las redes internas a los contenedores, confinadas dentro de los mismos, y la red principal que conecta a unos contenedores con otros [Guo 2009; Farrington 2010]. Dentro de un contenedor, a una escala de hasta unos pocos miles de hosts, resulta posible construir una red completamente conectada (como la que se ha descrito anteriormente)

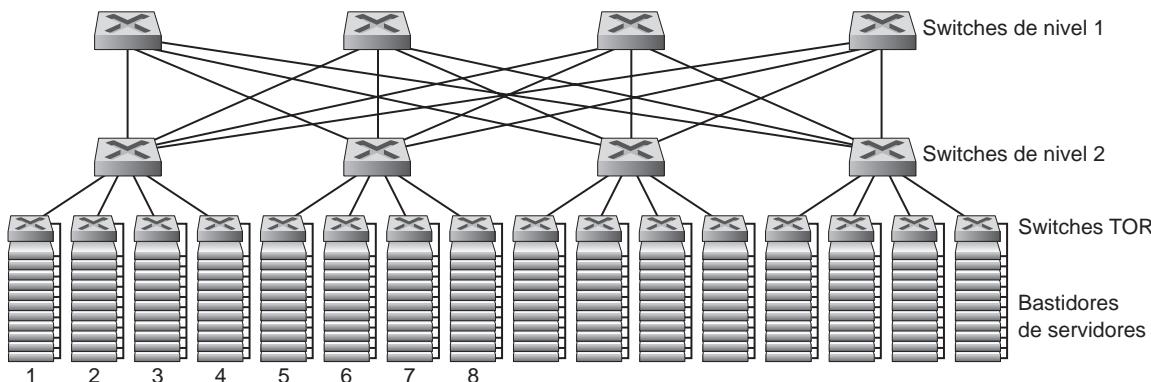


Figura 6.31 ♦ Topología altamente interconectada de red para centro de datos.

utilizando switches Gigabit Ethernet comerciales y baratos. Sin embargo, el diseño de la red principal, que interconecta cientos o miles de contenedores mientras proporciona un gran ancho de banda host a host entre contenedores, para las cargas de trabajo típicas, continúa planteando un desafío. En [Farrington 2010] se propone una arquitectura híbrida eléctrica/óptica de switches para interconectar los contenedores.

Al utilizar topologías altamente interconectadas, uno de los problemas principales es el de diseñar algoritmos de enrutamiento entre los switches. Una posibilidad [Greenberg 2009b] es emplear algún tipo de enrutamiento aleatorio. Otra posibilidad [Guo 2009] es incorporar múltiples tarjetas de interfaz de red en cada host, conectar cada host a múltiples switches comerciales de bajo coste y permitir que los propios hosts enruten el tráfico entre switches de forma inteligente. En los actuales centros de datos se están implantando hoy en día diversas extensiones y variantes de estas soluciones.

Otra tendencia importante es que los grandes proveedores en la nube construyen o personalizan cada vez más casi todos los equipos que incorporan a sus centros de datos, incluyendo los adaptadores de red, los switches, los routers, los TOR, el software y los protocolos de red [Greenberg 2015, Singh 2015]. Otra tendencia, de la que Amazon es pionero, es la de mejorar la fiabilidad mediante “zonas de disponibilidad” que lo que hacen, esencialmente, es duplicar los centros de datos en diferentes edificios cercanos. Al estar próximos los edificios (separados por solo unos kilómetros), los datos de las transacciones pueden sincronizarse entre los centros de datos pertenecientes a la misma zona de disponibilidad, proporcionando al mismo tiempo tolerancia ante fallos [Amazon 2014]. Resulta previsible que continúen produciéndose muchas más innovaciones en el diseño de centros de datos; animamos a los lectores interesados a que consulten los artículos y vídeos recientes sobre el diseño de redes para centros de datos.

6.7 Retrospectiva: un día en la vida de una solicitud de página web

Ahora que ya hemos cubierto el tema de la capa de enlace en este capítulo y las capas de red, de transporte y de aplicación en los anteriores, nuestro viaje de descenso por la pila de protocolos está completo. Al principio del libro (Sección 1.1) decíamos que “buena parte de este libro está relacionada con los protocolos de las redes de computadoras”, y en los primeros cinco capítulos hemos visto que es así. Antes de zambullirnos en los capítulos temáticos de la segunda parte del libro, conviene finalizar nuestro viaje descendente por la pila de protocolos adoptando una vista integrada y holística de los protocolos que hemos estudiado hasta el momento. Una forma, por tanto,

de adoptar esta “vista panorámica” consiste en identificar los muchos (¡muchísimos!) protocolos implicados en satisfacer incluso la más simple de las solicitudes: la descarga de una página web. La Figura 6.32 ilustra el que será nuestro escenario de trabajo: un estudiante, Benito, conecta una computadora portátil al switch Ethernet de su facultad y descarga una página web (por ejemplo, la página principal de www.google.com). Como sabemos ahora, son *muchas* las cosas que suceden “entre bambalinas” para satisfacer esta solicitud aparentemente simple. Una práctica de laboratorio con Wireshark incluida al final del capítulo le permitirá examinar con mayor detalle una serie de archivos de traza que contienen varios de los paquetes implicados en escenarios similares.

6.7.1 Inicio: DHCP, UDP, IP y Ethernet

Supongamos que Benito arranca su computadora portátil y luego la conecta a un cable Ethernet conectado al switch Ethernet de la facultad, que a su vez está conectado al router de la facultad, como se muestra en la Figura 6.32. El router de la facultad está conectado a un ISP, que en este caso se llama comcast.net. En este ejemplo, comcast.net proporciona el servicio DNS para la facultad; por tanto, el servidor DNS reside en la red de Comcast, en lugar de en la red de la facultad. Supondremos que el servidor DHCP está ejecutándose dentro del router, como suele ser el caso.

Cuando Benito conecta por primera vez su portátil a la red no puede hacer nada (por ejemplo, descargar una página web) sin una dirección IP. Por tanto, la primera acción relacionada con la red que lleva a cabo la computadora portátil de Benito es ejecutar el protocolo DHCP para obtener una dirección IP, así como otras informaciones, desde el servidor DHCP local:

- El sistema operativo del portátil de Benito crea un **mensaje de solicitud DHCP** (Sección 4.3.3) y lo incluye en un **segmento UDP** (Sección 3.3) con el puerto de destino 67 (servidor DHCP) y el puerto de origen 68 (cliente DHCP). A continuación, el segmento UDP es insertado dentro de un **datagrama IP** (Sección 4.3.1) con una dirección IP de destino de difusión (255.255.255.255) y una dirección IP de origen igual a 0.0.0.0, dado que la computadora portátil de Benito no tiene todavía una dirección IP.

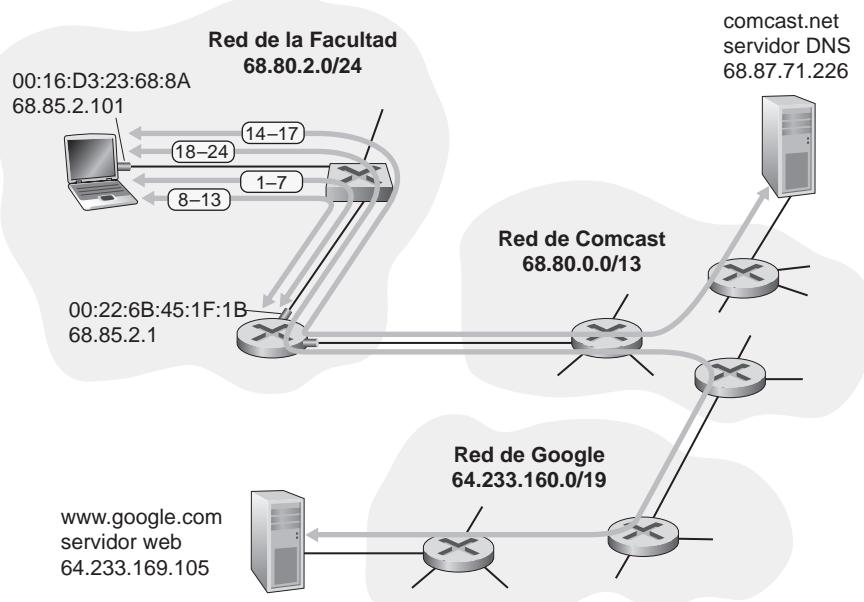


Figura 6.32 ♦ Un día en la vida de una solicitud de una página web: configuración y acciones de red.

2. El datagrama IP que contiene el mensaje de solicitud DHCP se inserta entonces en una **trama Ethernet** (Sección 6.4.2). La trama Ethernet tiene una dirección MAC de destino igual a FF:FF:FF:FF:FF:FF, de modo que la trama será difundida a todos los dispositivos conectados al switch (entre los que cabe esperar que se encuentre un servidor DHCP); la dirección MAC de origen de la trama es la de la computadora portátil de Benito, 00:16:D3:23:68:8A.
3. La trama Ethernet de difusión que contiene la solicitud DHCP es la primera trama enviada por la computadora de Benito al switch Ethernet. El switch difunde la trama entrante a todos los puertos de salida, incluyendo el puerto conectado al router.
4. El router recibe la trama Ethernet de difusión que contiene la solicitud DHCP a través de su propia interfaz con dirección MAC 00:22:6B:45:1F:1B, extrayendo el datagrama IP de la trama Ethernet. La dirección IP de destino de difusión contenida en el datagrama indica que este datagrama IP tiene que ser procesado por los protocolos de la capa superior existentes en este nodo, de modo que se **demultiplexa** (Sección 3.2) la carga útil del datagrama (un segmento UDP) y se entrega esa carga útil a UDP, tras lo cual se extrae del segmento UDP el mensaje de solicitud DHCP. Ahora el servidor DHCP dispone del mensaje de solicitud DHCP.
5. Supongamos que el servidor DHCP que se está ejecutando en el router puede asignar direcciones IP dentro del bloque **CIDR** (Sección 4.3.3) 68.85.2.0/24. En este ejemplo, todas las direcciones IP utilizadas dentro de la Facultad se encuentran dentro del bloque de direcciones de Comcast. Supongamos que el servidor DHCP asigna la dirección 68.85.2.101 al portátil de Benito. El servidor DHCP creará un **mensaje ACK DHCP** (Sección 4.3.3) que contendrá esta dirección IP, así como la dirección IP del servidor DNS (68.87.71.226), la dirección IP del router de pasarela predeterminado (68.85.2.1) y el bloque de subred (68.85.2.0/24) (o, lo que es lo mismo, la “máscara de red”). El mensaje DHCP se inserta dentro de un segmento UDP, que a su vez se incluye dentro de un datagrama IP, que se inserta en una trama Ethernet. La trama Ethernet tiene una dirección MAC de origen que será igual a la de la interfaz entre el router y la red doméstica (00:22:6B:45:1F:1B) y una dirección MAC de destino que será igual a la de la computadora portátil de Benito (00:16:D3:23:68:8A).
6. La trama Ethernet que contiene la respuesta ACK DHCP se envía (unidifusión) desde el router hacia el switch. Puesto que el switch tiene la característica de **auto-aprendizaje** (Sección 6.4.3) y ha recibido anteriormente una trama Ethernet (la que contenía la solicitud DHCP) desde el portátil de Benito, el switch sabe reenviar una trama dirigida a 00:16:D3:23:68:8A únicamente hacia el puerto de salida que conduce al portátil de Benito.
7. La computadora portátil de Benito recibe la trama Ethernet que contiene la respuesta ACK DHCP, extrae el datagrama IP de la trama Ethernet, extrae el segmento UDP del datagrama IP y extrae el mensaje ACK DHCP del segmento UDP. A continuación, el cliente DHCP de Benito anota su dirección IP y la dirección IP de su servidor DNS. También instala la dirección del router de pasarela predeterminado en su **tabla de reenvío IP** (Sección 4.1). El portátil de Benito enviará hacia el router de pasarela predeterminado todos los datagramas cuya dirección de destino caiga fuera de su subred 68.85.2.0/24. Llegados a este punto, la computadora portátil de Benito ha inicializado sus componentes de red y está lista para comenzar a procesar la extracción de la página web. (Observe que solo son necesarios los dos últimos pasos DHCP de los cuatro presentados en el Capítulo 4.)

6.7.2 Seguimos con el inicio: DNS y ARP

Cuando Benito escribe la dirección URL correspondiente a www.google.com en su navegador web, comienza la larga cadena de sucesos que terminará por hacer que se muestre la página de inicio de Google en su navegador web. El navegador de Benito comienza creando un **socket TCP** (Sección 2.7) que se utilizará para enviar la **solicitud HTTP** (Sección 2.2) hacia www.google.com. Para crear el socket, la computadora portátil de Benito necesita conocer la dirección IP de www.google.com. En la Sección 2.5 hemos visto que se utiliza el **protocolo DNS** para proporcionar este servicio de traducción de nombres a direcciones IP.

8. El sistema operativo de la computadora de Benito crea por tanto un **mensaje de consulta DNS** (Sección 2.5.3), incluyendo la cadena “www.google.com” en la sección de consulta del mensaje DNS. Después, este mensaje DNS se inserta dentro de un segmento UDP con un puerto de destino igual a 53 (servidor DNS). Después el segmento UDP se inserta dentro de un datagrama IP con una dirección IP de destino igual a 68.87.71.226 (la dirección del servidor DNS devuelta en el mensaje ACK DHCP en el Paso 5) y una dirección IP de origen igual a 68.85.2.101.
9. La computadora portátil de Benito inserta entonces el datagrama que contiene el mensaje de consulta DNS dentro de una trama Ethernet. Esta trama será enviada (direccionala en la capa de enlace) al router de pasarela de la red de la facultad de Benito. Sin embargo, aún cuando el portátil de Benito conoce la dirección IP del router de pasarela de la facultad (68.85.2.1), gracias al mensaje ACK DHCP del Paso 5 anterior, no sabe la dirección MAC del router de pasarela. Para obtener la dirección MAC de este router de pasarela, el portátil de Benito necesita utilizar el **protocolo ARP** (Sección 6.4.1).
10. La computadora portátil de Benito crea un mensaje de **consulta ARP** con una dirección IP de destino igual a 68.85.2.1 (el router de pasarela predeterminado), incluye el mensaje ARP dentro de una trama Ethernet con una dirección de destino de difusión (FF:FF:FF:FF:FF:FF) y envía la trama Ethernet hacia el switch, que entrega la trama a todos los dispositivos conectados, incluyendo al router de pasarela.
11. El router de pasarela recibe la trama que contiene el mensaje de solicitud ARP a través de la interfaz con la red de la facultad y se encuentra con que la dirección IP de destino, 68.85.2.1, contenida en el mensaje ARP, coincide con la dirección IP de su propia interfaz. El router de pasarela prepara en consecuencia una **respuesta ARP**, indicando que su dirección MAC 00:22:6B:45:1F:1B se corresponde con la dirección IP 68.85.2.1. A continuación, inserta el mensaje de respuesta ARP en una trama Ethernet, con una dirección de destino igual a 00:16:D3:23:68:8A (la de la computadora portátil de Benito) y envía la trama al switch, que la entrega al portátil de Benito.
12. El portátil de Benito recibe la trama que contiene el mensaje de respuesta ARP y extrae la dirección MAC del router de pasarela (00:22:6B:45:1F:1B) de ese mensaje.
13. La computadora portátil de Benito podrá ahora (*finalmente!*) dirigir la trama Ethernet que contiene la consulta DNS hacia la dirección MAC del router de pasarela. Observe que el datagrama IP de esta trama tiene la dirección IP de destino 68.87.71.226 (el servidor DNS), mientras que la trama tiene la dirección de destino 00:22:6B:45:1F:1B (el router de pasarela). El portátil de Benito envía esta trama al switch, que la entrega al router de pasarela.

6.7.3 Seguimos con el inicio: enrutamiento dentro del dominio al servidor DNS

14. El router de pasarela recibe la trama y extrae el datagrama IP que contiene la consulta DNS. El router busca la dirección de destino de este datagrama (68.87.71.226) y determina a partir de su tabla de reenvío que el datagrama debe enviarse al router situado más a la izquierda dentro de la red de Comcast de la Figura 6.32. El datagrama IP se inserta dentro de una trama de la capa de enlace que resulte apropiada para el enlace que conecta el router de la facultad con el router de Comcast situado más a la izquierda, después de lo cual la trama se envía a través de ese enlace.
15. El router situado más a la izquierda dentro de la red de Comcast recibe la trama, extrae el datagrama IP, examina la dirección de destino del datagrama (68.87.71.226) y determina, gracias a su tabla de reenvío, la interfaz de salida a través de la cual debe reenviar el datagrama hacia el servidor DNS. La tabla de reenvío habrá sido previamente rellenada mediante el protocolo interno del dominio de Comcast (por ejemplo **RIP**, **OSPF** o **IS-IS**, Sección 5.3), así como mediante el **protocolo entre dominios de Internet, BGP** (Sección 5.4).
16. Finalmente, el datagrama IP que contiene la consulta DNS terminará por llegar al servidor DNS, el cual extrae el mensaje de consulta DNS, busca el nombre www.google.com en su base de datos DNS (Sección 2.5) y encuentra el **registro de recurso DNS** que contiene la dirección

- IP (64.233.169.105) para www.google.com (suponiendo que esa dirección esté actualmente almacenada en la caché del servidor DNS). Recuerde que estos datos de caché tienen su origen en el **servidor DNS autoritativo** (Sección 2.5.2) correspondiente a google.com. El servidor DNS compondrá un **mensaje de respuesta DNS** con la correspondencia entre el nombre de host y la dirección IP, después de lo cual inserta el mensaje de respuesta DNS en un segmento UDP e inserta este segmento en un datagrama IP dirigido a la computadora portátil de Benito (68.85.2.101). Este datagrama será reenviado de vuelta a través de la red de Comcast hasta el router de la facultad, y desde allí, a través del switch Ethernet, a la computadora de Benito.
17. La computadora portátil de Benito extrae la dirección IP del servidor www.google.com del mensaje DNS. *Finalmente*, después de un *montón* de trabajo, la computadora portátil de Benito estará ya lista para contactar con el servidor www.google.com.

6.7.4 Interacción web cliente-servidor: TCP y HTTP

18. Ahora que el portátil de Benito dispone de la dirección IP de www.google.com puede crear el **socket TCP** (Sección 2.7) que se utilizará para enviar el mensaje **GET HTTP** (Sección 2.2.3) a www.google.com. Cuando Benito crea el socket TCP, el protocolo TCP de su portátil tiene que llevar a cabo primero un **proceso de acuerdo en tres fases** (Sección 3.5.6) con el TCP de www.google.com. El portátil de Benito creará primero, por tanto, un segmento **SYN TCP** con puerto de destino 80 (para HTTP), insertará el segmento TCP dentro de un datagrama IP con una dirección de destino IP igual a 64.233.169.105 (www.google.com), incluirá el datagrama dentro de una trama con una dirección MAC de destino igual a 00:22:6B:45:1F:1B (el router de pasarela) y enviará la trama al switch.
19. Los routers de la red de la facultad, de la red de Comcast y de la red de Google reenvían el datagrama que contiene el segmento SYN TCP hacia www.google.com, utilizando la tabla de reenvío de cada router, como sucedía en los pasos 14–16 anteriores. Recuerde que las entradas de las tablas de reenvío de los routers que gobiernan el reenvío de paquetes a través del enlace entre dominios entre las redes de Comcast y de Google son determinadas mediante el protocolo **BGP** (Capítulo 5).
20. En algún momento, el datagrama que contiene el segmento SYN TCP llegará a www.google.com. El mensaje SYN TCP será extraído del datagrama y demultiplexado para ser entregado al socket de escucha asociado con el puerto 80. Se crea entonces un socket de conexión (Sección 2.7) para la conexión TCP entre el servidor HTTP de Google y la computadora portátil de Benito. Se genera después un segmento **SYNACK TCP** (Sección 3.5.6), se inserta dentro de un datagrama dirigido al portátil de Benito y, finalmente, se inserta dicho datagrama dentro de una trama de la capa de enlace que resulte apropiada para el enlace que conecta www.google.com con su router de primer salto.
21. El datagrama que contiene el segmento SYNACK TCP se reenvía a través de las redes de Google, de Comcast y de la facultad, terminando por llegar hasta la tarjeta Ethernet del portátil de Benito. El datagrama es demultiplexado dentro del sistema operativo y entregado al socket TCP creado en el Paso 18, con lo que entrará en estado conectado.
22. Ahora que el socket del portátil de Benito está (*finalmente!*) listo para enviar bytes a www.google.com, el navegador de Benito crea el mensaje GET HTTP (Sección 2.2.3) que contiene el URL que quiere extraer. Entonces, el mensaje GET HTTP se escribe en el socket con el mensaje GET pasando a constituir la carga útil de un segmento TCP. El segmento TCP se incluye en un datagrama y se envía y se entrega a www.google.com como en los Pasos 18–20.
23. El servidor HTTP en www.google.com lee el mensaje GET HTTP del socket TCP, crea un mensaje de **respuesta HTTP** (Sección 2.2), inserta el contenido de la página web solicitada en el cuerpo del mensaje de respuesta HTTP y envía el mensaje a través del socket TCP.
24. El datagrama que contiene el mensaje de respuesta HTTP se reenvía a través de las redes de Google, de Comcast y de la facultad y llega a la computadora portátil de Benito. El navegador web de Benito lee la respuesta HTTP del socket, extrae el código HTML correspondiente a la

página web del cuerpo de la respuesta HTTP y, finalmente, (*¡finalmente!*) muestra la página web.

El escenario descrito cubre una gran cantidad de aspectos de la comunicación por red. Si ha comprendido la mayor parte del ejemplo anterior o todo él, entonces habrá avanzado mucho desde que leyera por primera vez la Sección 1.1, donde decíamos que “buena parte de este libro trata de los protocolos de redes de computadoras”, momento en el que posiblemente se estaba preguntando qué es un protocolo. Aunque el ejemplo anterior puede parecer bastante detallado, hemos omitido varios posibles protocolos adicionales (como por ejemplo, NAT ejecutándose en el router de pasarela de la facultad, el acceso inalámbrico a dicha red, los protocolos de seguridad para el acceso a red o para cifrar segmentos y datagramas, o los protocolos de gestión de red), así como diversas consideraciones adicionales (el almacenamiento en caché web, la jerarquía DNS) que podemos encontrar en la red Internet pública. Hablaremos de algunos de estos temas y otros en la segunda parte del libro.

Por último, cabe recalcar que el ejemplo anterior era una visión integrada y holística, aunque también refleja los componentes esenciales de muchos de los protocolos que hemos estudiado en esta primera parte del libro. El ejemplo pretendía centrarse más en el “cómo” que en el “por qué”. Si desea una visión más amplia y reflexiva del diseño de los protocolos de red en general, consulte [Clark 1988, RFC 5218].

6.8 Resumen

En este capítulo hemos examinado la capa de enlace: sus servicios, los principios que subyacen a su funcionamiento y una serie de protocolos específicos importantes que utilizan esos principios a la hora de implementar servicios de la capa de enlace.

Hemos visto que el servicio básico de la capa de enlace consiste en mover un datagrama de la capa de red desde un nodo (host, switch, router, punto de acceso WiFi) hasta otro nodo adyacente. También hemos visto que todos los protocolos de la capa de enlace operan encapsulando un datagrama de la capa de red dentro de una trama de la capa de enlace antes de transmitir la trama a través del enlace existente hasta el nodo adyacente. Sin embargo, y yendo más allá de esta función común de entrampado, hemos estudiado que los diferentes protocolos de la capa de enlace proporcionan servicios muy distintos de acceso al enlace, entrega y transmisión. Estas diferencias se deben en parte a la amplia variedad de tipos de enlace sobre los que deben operar los protocolos de la capa de enlace. Un simple enlace punto a punto tiene un único emisor y un único receptor que se comunican a través de único “cable”. Los enlaces de acceso múltiple, por su parte, son compartidos por varios emisores y receptores; en consecuencia, el protocolo de la capa de enlace para un canal de acceso múltiple dispone de un protocolo (su protocolo de acceso múltiple) para la coordinación del acceso al enlace. En el caso de MPLS, el “enlace” que conecta dos nodos adyacentes (por ejemplo, dos routers IP que sean adyacentes en sentido IP, es decir, que ambos son routers IP del siguiente salto hacia un determinado destino) puede ser en realidad una *red* en sí misma. En un cierto sentido, la idea de una *red* considerada como un enlace no debería resultar demasiado extraña. Un enlace telefónico que conecta una computadora/módem doméstico con un módem/router remoto, por ejemplo, es en realidad una ruta que pasa a través de una sofisticada y compleja *red* telefónica.

Entre los principios que subyacen a la comunicación de la capa de enlace, hemos examinado las técnicas de detección y corrección de errores, los protocolos de acceso múltiple, el direccionamiento de la capa de enlace, la virtualización (redes VLAN), la construcción de redes LAN ampliadas y las redes para centros de datos. Gran parte de la atención hoy en día, en la capa de enlace, está puesta en estas redes commutadas. En el caso de la detección y corrección de errores, hemos examinado cómo es posible añadir bits adicionales a la cabecera de una trama con el fin de detectar, y en algunos casos de corregir, errores de inversión de bit que puedan producirse al transmitir la trama a través

de un enlace. Hemos analizado los esquemas simples de paridad y de suma de comprobación, así como los códigos de redundancia cíclica más robustos. Después, hemos pasado a analizar el tema de los protocolos de acceso múltiple. Hemos identificado y estudiado tres enfoques generales para la coordinación del acceso a un canal de difusión: técnicas de particionamiento del canal (TDM, FDM), técnicas de acceso aleatorio (los protocolos ALOHA y CSMA) y técnicas de toma de turnos (sondeo y paso de testigo). Hemos estudiado la red de acceso por cable y determinado que emplea muchos de estos métodos de acceso. Hemos visto que una consecuencia de hacer que múltiples nodos compartan un único canal de difusión era la necesidad de proporcionar direcciones de nodo en la capa de enlace. Hemos observado que las direcciones de la capa de enlace son muy distintas de las direcciones de la capa de red y que, en el caso de Internet, se utiliza un protocolo especial (ARP, Protocolo de resolución de direcciones) para traducir entre estos dos tipos de direccionamiento, y hemos analizado el utilizadísimo protocolo Ethernet en detalle. Después hemos examinado cómo los nodos que comparten un canal de difusión forman una red LAN y cómo pueden conectarse entre sí varias redes LAN para formar otras redes LAN de mayor tamaño, todo ello *sin* la intervención del enrutamiento de la capa red para interconectar esos nodos locales. También hemos visto cómo se pueden crear múltiples redes LAN virtuales sobre una única infraestructura de LAN física.

Hemos terminado nuestro estudio de la capa de enlace centrándonos en cómo las redes MPLS proporcionan servicios de la capa de enlace cuando interconectan routers IP y proporcionando también una introducción a los diseños de red para los centros de datos masivos actuales. Hemos concluido el capítulo (y de hecho los primeros cinco capítulos) identificando los muchos protocolos necesarios para acceder a una simple página web. Habiendo cubierto la capa de enlace, *hemos concluido nuestro viaje descendente por la pila de protocolos*. Verdaderamente, la capa física se encuentra por debajo de la capa de enlace, pero quizás sea mejor dejar los detalles de la capa física para otro curso (por ejemplo, un curso sobre teoría de la comunicación, más que sobre redes de computadoras). No obstante, es cierto que hemos tocado varios aspectos de la capa física en este capítulo y en el Capítulo 1 (nuestra exposición acerca de los medios físicos de la Sección 1.2). Consideraremos de nuevo la capa física cuando estudiemos las características de los enlaces inalámbricos en el siguiente capítulo.

Aunque nuestro viaje por la pila de protocolos ya haya concluido, nuestro estudio de las redes de computadoras no ha terminado en modo alguno. En los siguientes tres capítulos nos ocuparemos de las redes inalámbricas, la seguridad de red y las redes multimedia. Estos tres temas no encajan de manera natural en ninguna de las capas que conocemos. De hecho, cada uno de ellos cruza muchas de esas distintas capas. Comprender estos temas (que se califican de temas avanzados en algunos libros de texto) requiere por tanto un sólido conocimiento de todas las capas de la pila de protocolos, un conocimiento que nuestro estudio de la capa de enlace de datos nos ha permitido terminar de adquirir.

Problemas y cuestiones de repaso

Capítulo 6 Cuestiones de repaso

SECCIONES 6.1–6.2

- R1. Considere la analogía de los transportes de la Sección 6.1.1. Si el pasajero es análogo a un datagrama, ¿qué sería análogo a la trama de la capa de enlace?
- R2. Si todos los enlaces de Internet tuvieran que proporcionar un servicio de entrega fiable, ¿sería el servicio de entrega fiable de TCP redundante? ¿Por qué?
- R3. ¿Cuáles son algunos de los posibles servicios que puede ofrecer un protocolo de la capa de enlace a la capa de red? ¿Cuáles de estos servicios de la capa de enlace tienen servicios correspondientes en IP? ¿Y en TCP?

SECCIÓN 6.3

- R4. Suponga que dos nodos comienzan a transmitir al mismo tiempo un paquete de longitud L a través de un canal de difusión de velocidad R . Sea el retardo de propagación entre los dos nodos d_{prop} . ¿Se producirá una colisión si $d_{\text{prop}} < L/R$? ¿Por qué?
- R5. En la Sección 6.3 hemos enumerado cuatro características deseables de un canal de difusión. ¿Cuáles de estas características presenta el protocolo ALOHA con particiones? ¿Cuáles de estas características presentan los protocolos de paso de testigo?
- R6. En CSMA/CD, después de la quinta colisión, ¿cuál es la probabilidad de que un nodo elija $K = 4$? ¿A cuántos segundos de retardo corresponde el resultado $K = 4$ en una red Ethernet a 10 Mbps?
- R7. Describa los protocolos de sondeo y de paso de testigo utilizando la analogía de las interacciones de las personas que asisten a un cóctel.
- R8. ¿Por qué el protocolo token-ring resulta inefficiente si una red LAN tiene un perímetro muy grande?

SECCIÓN 6.4

- R9. ¿Cuál es el tamaño del espacio de direcciones MAC? ¿Y el del espacio de direcciones de IPv4? ¿Y el del espacio de direcciones de IPv6?
- R10. Suponga que los nodos A, B y C están conectados a la misma red LAN de difusión (a través de sus adaptadores). Si A envía miles de datagramas IP a B, con cada trama que los encapsula dirigida hacia la dirección MAC de B, ¿procesará el adaptador de C estas tramas? En caso afirmativo, ¿pasará el adaptador de C los datagramas IP de dichas tramas a la capa de red de C? ¿Cómo variaría su respuesta si A envía las tramas con la dirección MAC de difusión?
- R11. ¿Por qué las consultas ARP se envían dentro de una trama de difusión? ¿Por qué la respuesta ARP se envía dentro de una trama con una dirección MAC de destino específica?
- R12. En la red de la Figura 6.19 el router tiene dos módulos ARP, cada uno con su propia tabla ARP. ¿Es posible que la misma dirección MAC aparezca en ambas tablas?
- R13. Compare las estructuras de trama de Ethernet 10BASE-T, 100BASE-T y Gigabit. ¿En qué se diferencian?
- R14. Considere la Figura 6.15. En el sentido de direccionamiento explicado en la Sección 4.3, ¿cuántas subredes hay?
- R15. ¿Cuál es el número máximo de redes VLAN que se pueden configurar en un switch que soporta el protocolo 802.1Q? ¿Por qué?
- R16. Suponga que tenemos que conectar N switches que dan soporte a K grupos VLAN mediante un protocolo de enlace troncal (*trunking*)? ¿Cuántos puertos son necesarios para conectar los switches? Justifique su respuesta.

Problemas

- P1. Suponga que el contenido de información de un paquete es el patrón de bits 1110 0110 1001 1101 y que está utilizando un esquema de paridad par. ¿Cuál sería el valor del campo que contiene los bits de paridad para el caso de un esquema de paridad bidimensional? La respuesta debe ser tal que se utilice un campo de suma de comprobación de longitud mínima.
- P2. Demuestre (proporcionando un ejemplo distinto del de la Figura 6.5) que los códigos de paridad bidimensional permiten corregir y detectar un único error de bit. Indique (proporcionando un ejemplo) un error doble de bit que pueda ser detectado pero no corregido.

- P3. Suponga que la parte de información de un paquete (D en la Figura 6.3) contiene 10 bytes compuestos por la representación en código ASCII binario sin signo de 8 bits de la cadena de caracteres “Networking”. Calcule la suma de comprobación de Internet para estos datos.
- P4. Considere el problema anterior, pero suponga que los 10 bytes contienen:
- La representación binaria de los números 1 a 10.
 - La representación ASCII de las letras B hasta K (mayúsculas).
 - La representación ASCII de las letras b hasta k (minúsculas).
- Calcule la suma de comprobación de Internet para estos datos.
- P5. Considere el generador de 5 bits, $G = 10011$, y suponga que D tiene el valor 1010101010. ¿Cuál es el valor de R ?
- P6. Considere el problema anterior, pero ahora suponga que D tiene el valor:
- 1001010101.
 - 0101101010.
 - 1010100000.
- P7. En este problema vamos a explorar algunas de las propiedades del código CRC. Para el generador $G = 1001$ dado en la Sección 6.2.3, responda a las siguientes cuestiones:
- ¿Por qué puede detectar cualquier error simple de bit en los datos D ?
 - ¿Puede el generador G anterior detectar cualquier número impar de errores de bit? ¿Por qué?
- P8. En la Sección 6.3 hemos proporcionado un esbozo del cálculo de la eficiencia del protocolo ALOHA con particiones. En este problema vamos a completar dicho cálculo.
- Recuerde que cuando hay N nodos activos, la eficiencia de ALOHA con particiones es $Np(1 - p)^{N-1}$. Calcule el valor de p que maximiza esta expresión.
 - Utilizando el valor de p determinado en el apartado (a), calcule la eficiencia del protocolo ALOHA con particiones haciendo que N tienda a infinito. *Sugerencia:* $(1 - 1/N)^N$ tiende a $1/e$ cuando N tiende a infinito.
- P9. Demuestre que la eficiencia máxima del protocolo ALOHA puro es $1/(2e)$. *Nota:* este problema es sencillo después de haber completado el problema anterior.
- P10. Considere los nodos A y B que utilizan el protocolo ALOHA con particiones para competir por un canal. Suponga que el nodo A tiene más datos para transmitir que el nodo B, y que la probabilidad de retransmisión del nodo A, p_A , es mayor que la probabilidad de retransmisión del nodo B, p_B .
- Proporcione una fórmula para la tasa media de transferencia del nodo A. ¿Cuál es la eficiencia total del protocolo con estos dos nodos?
 - Si $p_A = 2p_B$, ¿será la tasa media de transferencia de A el doble que la del nodo B? ¿Por qué? Si no es así, ¿cómo podemos seleccionar valores de p_A y p_B para que esto ocurra?
 - En general, suponga que hay N nodos, entre los que el nodo A tiene una probabilidad de retransmisión $2p$ y todos los demás nodos tienen una probabilidad de retransmisión p . Proporcione las expresiones necesarias para calcular las tasas medias de transferencia del nodo A y de los restantes nodos.
- P11. Suponga que cuatro nodos activos (nodos A, B, C y D) están compitiendo por el acceso a un canal utilizando el protocolo ALOHA con particiones. Suponga que cada nodo tiene un número infinito de paquetes que transmitir y que cada nodo intenta transmitir en cada partición con una probabilidad p . La primera partición tiene el número 1, la segunda el número 2, etc.
- ¿Cuál es la probabilidad de que el nodo A tenga éxito por primera vez en la partición 5?
 - ¿Cuál es la probabilidad de que algún nodo (A, B, C o D) tenga éxito en la partición 4?
 - ¿Cuál es la probabilidad de que el primer éxito suceda en la partición 3?
 - ¿Cuál es la eficiencia de este sistema de cuatro nodos?
- P12. Dibuje una gráfica con la eficiencia de los protocolos ALOHA con particiones y ALOHA puro en función de p para los siguientes valores de N :

- a. $N = 15$.
 b. $N = 25$.
 c. $N = 35$.
- P13. Considere un canal de difusión con N nodos y una tasa de transmisión de R bps. Suponga que el canal de difusión utiliza sondeo (con un nodo adicional de sondeo) para regular el acceso múltiple. Suponga que la cantidad de tiempo desde que un nodo completa una transmisión hasta que se le permite transmitir al nodo siguiente (es decir, el retardo de sondeo) es d_{sondeo} . Suponga que dentro de una ronda de sondeo, a cada nodo se le permite transmitir un máximo de Q bits. ¿Cuál es la tasa máxima de transferencia del canal de difusión?
- P14. Considere tres redes LAN interconectadas mediante dos routers, como se muestra en la Figura 6.33.
- Asigne direcciones IP a todas las interfaces. Para la Subred 1 utilice direcciones de la forma 192.168.1.xxx; para la Subred 2 utilice direcciones de la forma 192.168.2.xxx; y para la Subred 3 emplee direcciones de la forma 192.168.3.xxx.
 - Asigne direcciones MAC a todos los adaptadores.
 - Considere el envío de un datagrama IP desde el host E al host B. Suponga que todas las tablas ARP están actualizadas. Enumere todos los pasos, como hemos hecho en el ejemplo para un único router en la Sección 6.4.1.
 - Repita el apartado (c) suponiendo ahora que la tabla ARP del host emisor está vacía (y que todas las demás tablas están actualizadas).
- P15. Considere la Figura 6.33. Ahora vamos sustituir el router situado entre las subredes 1 y 2 por un switch S1, y vamos a etiquetar el router situado entre las subredes 2 y 3 como R1.
- Considere el envío de un datagrama IP desde el host E al host F. ¿Pedirá el host E al router R1 que le ayude a reenviar el datagrama? ¿Por qué? En la trama Ethernet que contiene el datagrama IP, ¿cuáles son las direcciones IP y MAC de origen y de destino?
 - Suponga que E quiere enviar un datagrama IP a B y suponga que la caché ARP de E no contiene la dirección MAC de B. ¿Realizará E una consulta ARP para averiguar la dirección MAC de B? ¿Por qué? En la trama Ethernet (que contiene el datagrama IP destinado a B) que se le entrega al router R1, ¿cuáles son las direcciones IP y MAC de origen y de destino?

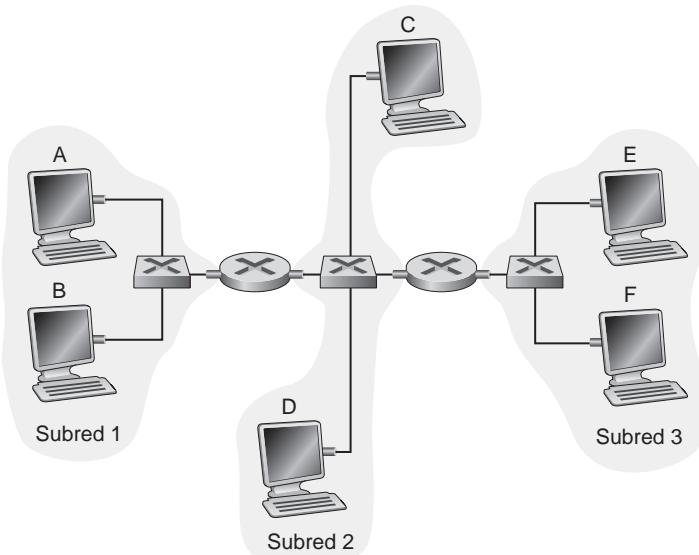


Figura 6.33 ♦ Tres subredes interconectadas mediante routers.

- c. Suponga que el host A quiere enviar un datagrama IP al host B, y que ni la caché ARP de A contiene la dirección MAC de B ni la caché ARP de B contiene la dirección MAC de A. Suponga además que la tabla de reenvío del switch S1 contiene entradas únicamente para el host B y el router R1. Por tanto, A difundirá un mensaje de solicitud ARP. ¿Qué acciones realizará el switch S1 una vez que reciba el mensaje de solicitud ARP? ¿Recibirá también el router R1 esta solicitud ARP? En caso afirmativo, ¿reenviará R1 el mensaje hacia la Subred 3? Una vez que el host B reciba este mensaje de solicitud ARP, devolverá al host A un mensaje de respuesta ARP. Pero, ¿enviará un mensaje de consulta ARP para preguntar por la dirección MAC de A? ¿Por qué? ¿Qué hará el switch S1 una vez que reciba el mensaje de respuesta ARP del host B?
- P16. Considere el problema anterior, pero ahora suponga que el router situado entre las subredes 2 y 3 es sustituido por un switch. Responda a las preguntas de los apartados (a)–(c) del problema anterior en este nuevo contexto.
- P17. Recuerde que con el protocolo CSMA/CD el adaptador espera $K \cdot 512$ períodos de bit después de producirse una colisión, donde K se selecciona aleatoriamente. Para $K = 100$, ¿cuánto tiempo espera el adaptador hasta volver al Paso 2 para un canal de difusión a 10 Mbps? ¿Y para un canal de difusión a 100 Mbps?
- P18. Suponga que los nodos A y B se encuentran en el mismo canal de difusión a 10 Mbps y que el retardo de propagación entre los dos nodos es igual a 325 períodos de bit. Suponga que se emplean CSMA/CD y paquetes Ethernet para este canal de difusión. Suponga que el nodo A comienza a transmitir una trama y que, antes de finalizar, el nodo B comienza a transmitir otra trama. ¿Podría A finalizar la transmisión antes de detectar que B ha transmitido? ¿Por qué? Si la respuesta es afirmativa, entonces A creerá incorrectamente que su trama se ha transmitido con éxito sin que se haya producido una colisión. *Sugerencia:* suponga que en el instante $t = 0$ períodos de bit A comienza a transmitir una trama. En el caso peor, A transmitirá una trama de tamaño mínimo que ocupará $512 + 64$ períodos de bit. Por lo que A terminaría de transmitir la trama en $t = 512 + 64$ períodos de bit. Por tanto, la respuesta es no si la señal de B alcanza a A antes del instante $t = 512 + 64$ períodos de bits. En el caso peor, ¿cuándo alcanzaría a A la señal de B?
- P19. Suponga que los nodos A y B se encuentran en el mismo canal de difusión a 10 Mbps y que el retardo de propagación entre los dos nodos es igual a 245 períodos de bit. Suponga que A y B envían tramas Ethernet al mismo tiempo, que las tramas colisionan y que luego A y B seleccionan diferentes valores de K en el algoritmo CSMA/CD. Suponiendo que no haya ningún otro nodo activo, ¿pueden colisionar las retransmisiones de A y B? Para nuestros propósitos, basta con resolver el siguiente ejemplo. Suponga que A y B comienzan la transmisión en $t = 0$ períodos de bit. Ambos detectan la colisión en el instante $t = 245$ períodos de bits. Suponga que $K_A = 0$ y $K_B = 1$. ¿Para qué instante programará B su retransmisión? ¿En qué momento comenzará A su transmisión? (*Nota:* los nodos tienen que esperar a que el canal esté inactivo después de volver al Paso 2, consulte el protocolo). ¿En qué momento alcanza a B la señal de A? ¿Se abstendrá B de transmitir en el instante programado?
- P20. En este problema tendrá que deducir la eficiencia de un protocolo de acceso múltiple de tipo CSMA/CD. En dicho protocolo, el tiempo está particionado y todos los adaptadores están sincronizados con las particiones. Sin embargo, a diferencia del protocolo ALOHA con particiones, la longitud de una partición (en segundos) es muy inferior al tiempo de trama (el tiempo necesario para transmitir una trama). Sea S la longitud de una partición. Suponga que todas las tramas tienen una longitud constante $L = kRS$, donde R es la velocidad de transmisión del canal y k es un entero de gran magnitud. Suponga que existen N nodos, cada uno con un número infinito de tramas para enviar. Suponga también que $d_{\text{prop}} < S$, de modo que todos los nodos pueden detectar una colisión antes de que finalice una partición de tiempo. El protocolo es como sigue:

- Si, para una partición determinada, ningún nodo está en posesión del canal, todos los nodos competirán por el acceso al canal; en particular, cada uno de los nodos transmite durante esa partición con una probabilidad p . Si exactamente un nodo transmite en la partición, dicho nodo se apropiará del canal durante las subsiguientes $k - 1$ particiones y transmite su trama completa.
- Si algún nodo está en posesión del canal, todos los demás nodos se abstienen de transmitir hasta que el nodo que posee el canal ha terminado de transmitir su trama. Una vez que este nodo ha transmitido su trama, todos los nodos compiten por acceder al canal.

Observe que el canal alterna entre dos estados distintos: el estado productivo, que dura exactamente k particiones, y el estado no productivo, que dura un número aleatorio de particiones. Obviamente, la eficiencia del canal será la relación $k/(k + x)$, donde x es el número esperado de particiones no productivas consecutivas.

- a. Para N y p fijos, determine la eficiencia de este protocolo.
 - b. Para un valor fijo N , determine el valor de p que maximiza la eficiencia.
 - c. Utilizando el valor de p (que es función de N) calculado en el apartado (b), determine la eficiencia cuando N tiende a infinito.
 - d. Demuestre que esta eficiencia tiende a 1 a medida que aumenta el tamaño de la trama.
- P21. Considere la Figura 6.33 del Problema P14. Proporcione direcciones MAC e IP para las interfaces del host A, de ambos routers y del host F. Suponga que el host A envía un datagrama al host F. Indique las direcciones MAC de origen y de destino contenidas en la trama que encapsula este datagrama IP a medida que la trama es transmitida (i) de A al router de la izquierda, (ii) desde el router de la izquierda al router de la derecha y (iii) desde el router de la derecha al host F. Indique también las direcciones IP de origen y de destino del datagrama IP encapsulado dentro de la trama en cada uno de estos instantes de tiempo.
- P22. Suponga ahora que el router de la izquierda de la Figura 6.33 se sustituye por un switch. Los hosts A, B, C y D y el router de la derecha se conectan en estrella a ese switch. Indique las direcciones MAC de origen y de destino contenidas en la trama que encapsula a este datagrama IP a medida que la trama se transmite (i) desde A al switch, (ii) desde el switch al router de la derecha y (iii) desde el router de la derecha a F. Indique también las direcciones IP de origen y de destino contenidas en el datagrama IP encapsulado dentro de la trama en cada uno de estos instantes de tiempo.
- P23. Considere la Figura 6.15. Suponga que todos los enlaces son a 100 Mbps. ¿Cuál es la tasa máxima total agregada de transferencia que puede conseguirse en los nueve hosts y los dos servidores de esta red? Puede suponer que cualquier host o servidor puede enviar a cualquier otro host o servidor. ¿Por qué?
- P24. Suponga que los tres switches departamentales de la Figura 6.15 se sustituyen por hubs. Todos los enlaces son a 100 Mbps. Responda en estas condiciones a las cuestiones planteadas en el Problema P23.
- P25. Suponga que *todos* los switches de la Figura 6.15 son sustituidos por hubs. Todos los enlaces son a 100 Mbps. Responda en estas condiciones a las cuestiones planteadas en el Problema P23.
- P26. Considere la operación de un switch con aprendizaje en el contexto de una red en la que seis nodos etiquetados como A hasta F están conectados en estrella a un switch Ethernet. Suponga que (i) B envía una trama a E, (ii) E responde enviando una trama a B, (iii) A envía una trama a B, (iv) B responde enviando una trama a A. Inicialmente, la tabla del switch está vacía. Muestre el estado de la tabla del switch antes y después de cada uno de estos sucesos. Para cada suceso, identifique el enlace o los enlaces a través de los cuales se reenviará la trama transmitida y justifique brevemente sus respuestas.

- P27. En este problema vamos a explorar el uso de pequeños paquetes para aplicaciones de Voz sobre IP. Uno de los inconvenientes de un tamaño de paquete pequeño es que una gran parte del ancho de banda del enlace es consumido por los bytes de sobrecarga. De cara al análisis, suponga que el paquete consta de P bytes de datos y 5 bytes de cabecera.
- Considere el envío directo de una fuente de voz codificada digitalmente. Suponga que la fuente se codifica a una velocidad constante de 128 kbps. Suponga que cada paquete se rellena completamente antes de que la fuente envíe el paquete a la red. El tiempo requerido para llenar un paquete se denomina **retardo de empaquetado**. En función de L , determine el retardo de empaquetado en milisegundos.
 - Los retardos de empaquetado superiores a 20 ms pueden dar lugar a un eco perceptible y desagradable. Determine el retardo de empaquetado para $L = 1.500$ bytes (lo que se corresponde aproximadamente con un paquete Ethernet de tamaño máximo) y para $L = 50$ (lo que se corresponde con un paquete ATM).
 - Calcule el retardo de almacenamiento y reenvío en un único switch para una velocidad de enlace de $R = 622$ Mbps para $L = 1.500$ bytes y para $L = 50$ bytes.
 - Comente las ventajas de utilizar un tamaño de paquete pequeño.
- P28. Considere el único switch VLAN de la Figura 6.25 y suponga que se conecta un router externo al puerto 1 del switch. Asigne direcciones IP a los hosts de las redes IE y CC y a la interfaz del router. Indique los pasos seguidos tanto en la capa de red como en la capa de enlace para transferir un datagrama IP desde un host de IE a un host de CC (*Sugerencia:* vuelva a leer en el texto los comentarios acerca de la Figura 6.19).
- P29. Considere la red MPLS mostrada en la Figura 6.29 y suponga que ahora los routers R5 y R6 son compatibles con MPLS. Suponga que deseamos realizar la ingeniería de tráfico de modo que los paquetes procedentes de R6 y destinados a A se comutén hacia A a través de R6-R4-R3-R1, y que los paquetes procedentes de R5 destinados a A se comutén a través de R5-R4-R2-R1. Detalle las tablas MPLS de R5 y R6, así como la tabla modificada de R4, que harían esto posible.
- P30. Considere de nuevo el mismo escenario que el problema anterior, pero suponga que los paquetes de R6 destinados a D se comutan a través de R6-R4-R3, mientras que los paquetes procedentes de R5 destinados a D se comutan a través de R4-R2-R1-R3. Determine las tablas MPLS en todos los routers que harían esto posible.
- P31. En este problema vamos a juntar muchas de las cosas que hemos aprendido acerca de los protocolos de Internet. Suponga que entra en una habitación, se conecta a Ethernet y desea descargar una página web. ¿Cuáles son todos los pasos de protocolo que tienen lugar, comenzando desde el instante en que enciende su PC y hasta el momento en que obtiene la página web? Suponga que no hay nada en la caché DNS ni en la caché del navegador cuando enciende su PC. (*Sugerencia:* los pasos incluyen el uso de los protocolos Ethernet, DHCP, ARP, DNS, TCP y HTTP.) Indique explícitamente en sus pasos cómo se obtienen las direcciones IP y MAC de un router de pasarela.
- P32. Considere la red de un centro de datos con la topología jerárquica de la Figura 6.30. Suponga ahora que hay 80 pares de flujos, habiendo diez flujos entre el primer y el noveno bastidores, diez flujos entre el segundo y el décimo, y así sucesivamente. Suponga también que todos los enlaces de la red son de 10 Gbps, excepto los enlaces entre los hosts y los switches TOR, que son de 1 Gbps.
- Cada flujo tiene la misma velocidad de datos; determine la velocidad máxima de un flujo.
 - Para el mismo patrón de tráfico, determine la velocidad máxima de un flujo para la topología altamente interconectada de la Figura 6.31.
 - Suponga ahora que hay un patrón de tráfico similar, pero con 20 hosts en cada bastidor y 160 pares de flujos. Determine las velocidades máximas de flujo para las dos topologías.

- P33. Considere la red jerárquica de la Figura 6.30 y suponga que el centro de datos necesita soportar la distribución de correo electrónico y de vídeo, entre otras aplicaciones. Suponga que cuatro bastidores de servidores están reservados para correo electrónico y que otros cuatro bastidores están reservados para vídeo. Para cada una de las aplicaciones, los cuatro bastidores deben encontrarse por debajo de un único switch de nivel 2, puesto que los enlaces entre el nivel 2 y el nivel 1 no disponen de ancho de banda suficiente para soportar el tráfico intra-aplicación. Para la aplicación de correo electrónico, suponga que durante el 99,9 por ciento del tiempo solo se utilizan tres bastidores, y que la aplicación de vídeo tiene un patrón de uso idéntico.
- ¿Durante qué fracción de tiempo necesitará usar un cuarto bastidor la aplicación de correo electrónico? ¿Y la aplicación de vídeo?
 - Suponiendo que uso del correo electrónico y del vídeo son independientes, ¿durante qué fracción del tiempo (es decir, con qué probabilidad) necesitarán ambas aplicaciones su cuarto bastidor?
 - Suponga que es aceptable que una aplicación presente una carencia de servidores el 0,001 por ciento del tiempo o menos (provocando raros períodos de degradación del rendimiento para los usuarios). Explique cómo puede utilizarse la topología de la Figura 6.31 de manera que solo se asignen colectivamente siete bastidores a las dos aplicaciones (suponiendo que la topología pueda soportar todo el tráfico).

Prácticas de laboratorio con Wireshark

En el sitio web del libro, <http://www.pearsonhighered.com/cs-resources/>, encontrará una práctica de laboratorio con Wireshark que permite examinar el funcionamiento del protocolo IEEE 802.3 y el formato de la trama. Una segunda práctica de laboratorio con Wireshark examina las trazas de paquetes tomadas en un escenario de red doméstica.

UNA ENTREVISTA CON...

Simon S. Lam

Simon S. Lam es profesor y Regent Chair del Departamento de Ciencias de la Computación de la Universidad de Texas en Austin. Entre 1971 y 1974 estuvo trabajando en el Centro de Medidas de Red de ARPA en UCLA, donde trabajó con conmutación de paquetes vía satélite y vía radio. Dirigió un grupo de investigación que inventó los sockets seguros y prototipó, en 1993, la primera capa de sockets seguros (SSL) denominada Secure Network Programming, que ganó el premio al mejor sistema software de ACM en 2004. Sus intereses de investigación se encuentran en el campo del diseño y el análisis de protocolos de red y servicios de seguridad. Se graduó en la universidad del Estado de Washington, obteniendo su máster y su doctorado en UCLA. Fue elegido para la Academia Nacional de Ingeniería en 2007.



¿Por qué decidió especializarse en el campo de las redes?

Cuando llegué a UCLA como estudiante graduado en el otoño de 1969, mi intención era estudiar teoría de control. Entonces asistí a las clases de teoría de colas de Leonard Kleinrock y me quedé muy impresionado. Durante un tiempo estuve trabajando en el control adaptativo de sistemas de colas, como posible tema de mi tesis. A principios de 1972, Larry Roberts inició el proyecto Satellite System de ARPAnet (posteriormente denominado Packet Satellite). El profesor Kleinrock me pidió que me uniera al proyecto. Lo primero que hicimos fue añadir un algoritmo de backoff simple, aunque realista, al protocolo ALOHA con particiones. Poco después, me encontré con numerosos problemas de investigación interesantes, como el problema de inestabilidad de ALOHA y la necesidad de backoff adaptativo, temas que terminarían formando el núcleo de mi tesis.

Usted estuvo muy activo en los primeros días de Internet en la década de 1970, comenzando sus días de estudiante en UCLA. ¿Cómo eran las cosas entonces? ¿Alguien imaginaba lo que Internet llegaría a ser?

La atmósfera no era realmente distinta de la de otros proyectos de construcción de sistemas que yo había conocido en la industria y en las instituciones académicas. El objetivo inicialmente establecido para ARPAnet era bastante modesto, es decir, lo que se quería era proporcionar acceso a una serie de computadoras muy caras desde ubicaciones remotas, de modo que muchos más científicos pudieran utilizarlas. Sin embargo, con el inicio del proyecto Packet Satellite en 1972 y del proyecto Packet Radio en 1973, los objetivos de ARPA se habían ampliado sustancialmente. En 1973, ARPA estaba construyendo tres redes de paquetes distintas al mismo tiempo y Vint Cerf y Bob Kahn se vieron obligados a desarrollar una estrategia de interconexión.

En aquel entonces, todos estos desarrollos progresivos en el campo de las redes se veían (o eso creo) más que como algo lógico que como algo mágico. Nadie podría haber previsto la escala de Internet y la potencia actual de las computadoras personales. Pasó una década antes de que aparecieran los primeros PC. Para poner las cosas en perspectiva, la mayoría de los estudiantes enviaban sus programas de computadora en forma de pilas de fichas perforadas para su procesamiento por lotes. Sólo algunos estudiantes tenían acceso directo a las computadoras, que normalmente estaban ubicadas en áreas restringidas. Los modems eran lentos y constituyían todavía una rareza. Como estudiante graduado lo único que yo tenía era un teléfono sobre mi mesa y utilizaba lápiz y papel para hacer la mayor parte de mi trabajo.

¿Hacia dónde cree que se encaminan el campo de las redes e Internet?

En el pasado, la simplicidad del protocolo IP de Internet era su mayor fortaleza a la hora de vencer otras soluciones competidoras y convertirse en el estándar *de facto* para la comunicación entre redes. A diferencia de otras soluciones competidoras, como X.25 en la década de 1980 y ATM en la década de 1990, IP puede ejecutarse por encima de cualquier tecnología de red de la capa de enlace, porque ofrece únicamente un servicio de datagramas de mejor esfuerzo. Por tanto, cualquier red de paquetes puede conectarse a Internet.

Hoy día, la mayor fortaleza de IP es en realidad una desventaja. IP es como una especie de camisa de fuerza que hace que los desarrollos de Internet queden constreñidos a una serie de direcciones específicas. En los últimos años muchos investigadores han redirigido sus esfuerzos, concentrándose únicamente en la capa de aplicación. También se están desarrollando numerosas investigaciones en el campo de las redes inalámbricas ad hoc, de las redes de sensores y de las redes por satélite. Estas redes pueden verse como sistemas autónomos o como sistemas de la capa de enlace, pudiendo florecer ese tipo de sistemas porque caen fuera de la camisa de fuerza representada por IP.

Muchas personas están entusiasmadas con las posibilidades que los sistemas P2P ofrecen como plataforma para el desarrollo de aplicaciones de Internet novedosas. Sin embargo, los sistemas P2P son muy inefficientes en su uso en los recursos de Internet. Una de las cosas que me preocupa es si la capacidad de transmisión y de commutación del núcleo de Internet continuará incrementándose más rápido que la demanda de tráfico a medida que Internet crezca para interconectar todo tipo de dispositivos y soportar las futuras aplicaciones de tipo P2P. Sin un sustancial sobredimensionamiento de la capacidad, garantizar la estabilidad de la red en presencia de ataques maliciosos y de congestión continuará representando un enorme desafío.

El enorme crecimiento de Internet también requiere asignar nuevas direcciones IP a gran velocidad a los operadores de red y a las empresas de todo el mundo. A la velocidad actual, el conjunto de direcciones IPv4 no asignadas se agotará en unos pocos años. Cuando eso suceda, solo podrán asignarse grandes bloques contiguos del espacio de direcciones a partir del espacio de direcciones de IPv6. Como la adopción de IPv6 está siendo lenta en un principio, debido a la falta de incentivos para los nuevos usuarios, lo más probable es que IPv4 e IPv6 coexistan en Internet durante muchos años todavía. Una migración satisfactoria desde una Internet predominantemente IPv4 a otra predominantemente IPv6 requerirá un sustancial esfuerzo global.

¿Cuál es la parte más atractiva de su trabajo?

La parte más atractiva de mi trabajo como profesor es enseñar y motivar a *todos* los alumnos de mi clase y a *todos* los estudiantes de doctorado a los que superviso, en lugar de concentrarme solo en los más brillantes. Las personas muy brillantes y motivadas pueden requerir alguna guía pero no mucho más. A menudo aprendo más de estos estudiantes de lo que ellos aprenden de mí. Lo más atractivo, el mayor desafío, es educar y motivar a los estudiantes que no son sobresalientes.

¿Qué impacto cree que tendrá la tecnología en el futuro sobre la enseñanza?

Antes o después, casi todos los conocimientos humanos serán accesibles a través de Internet, que será la herramienta más potente para el aprendizaje. Esta enorme base de conocimientos nos dará la posibilidad de nivelar el campo de juego para los estudiantes de todo el mundo. Por ejemplo, los estudiantes motivados en cualquier país podrán acceder a los sitios web de primera fila, a conferencias multimedia y a materiales formativos. Hoy día, se dice que las bibliotecas digitales de IEEE y ACM han acelerado el desarrollo de las investigaciones en ciencias de la computación en China. Con el tiempo, Internet eliminará todas las barreras geográficas en lo que a la enseñanza se refiere.



Redes inalámbricas y móviles

En el mundo de la telefonía, los últimos 20 años han sido indudablemente la edad dorada de la telefonía celular. El número de abonados a los servicios de telefonía celular en todo el mundo ha pasado de 34 millones en 1993 a 7.000 millones en 2014, sobrepasando ahora el número de abonados celulares al de líneas telefónicas cableadas. En la actualidad existe un mayor número de abonos de teléfonos móviles que personas hay en el planeta. Las numerosas ventajas de los teléfonos celulares son evidentes para todo el mundo: disponemos de acceso a la red de telefonía global en cualquier momento, en cualquier lugar y sin ningún tipo de restricción, utilizando un dispositivo ligero y muy portátil. Más recientemente, las computadoras portátiles, los smartphones (teléfonos inteligentes) y las tabletas se conectan de forma inalámbrica a Internet a través de una red celular o WiFi. Y, cada vez más, dispositivos como las consolas de juegos, termostatos, sistemas de alarma, electrodomésticos, relojes, gafas, vehículos, sistemas de control de tráfico y otros, se conectan de forma inalámbrica a Internet.

Desde el punto de vista de la comunicación por red, los desafíos planteados por estos dispositivos inalámbricos y móviles, particularmente en las capas de enlace y de red, son tan diferentes de los de las redes de computadoras cableadas tradicionales, que resulta imperativo dedicar todo un capítulo (es decir, *este* capítulo) al estudio de las redes inalámbricas y móviles.

Comenzaremos el capítulo con un estudio de los usuarios móviles, los enlaces inalámbricos y las redes inalámbricas, así como de su relación con las redes de mayor tamaño (normalmente cableadas) a las que se conectan. Estableceremos la distinción entre los desafíos planteados por la naturaleza *inalámbrica* de los enlaces de comunicaciones en dichas redes y por la *movilidad* que esos enlaces inalámbricos permiten. Al realizar esta importante distinción entre el carácter inalámbrico y la movilidad podremos aislar, identificar y dominar mucho mejor los conceptos clave de cada una de las áreas. Observe que existen, por supuesto, muchos entornos de red en los que los nodos de red son inalámbricos pero no móviles (por ejemplo, redes inalámbricas domésticas o empresariales con estaciones de trabajo estáticas y grandes pantallas de computadora), y que existen formas de movilidad que no requieren de enlaces inalámbricos (por ejemplo, un trabajador que utiliza una computadora portátil con conexión por cable en casa, apaga la computadora, va hasta la oficina

y conecta la computadora a la red cableada de la empresa). Por supuesto, muchos de los entornos de red más atractivos son aquellos en los que los usuarios son a la vez inalámbricos y móviles; por ejemplo, un escenario en el que un usuario móvil (situado por ejemplo en el asiento posterior de un vehículo) mantiene una llamada de voz sobre IP y múltiples conexiones TCP activas, mientras el vehículo circula por la autopista a 160 kilómetros por hora, muy pronto en un vehículo autónomo. Es aquí, en la intersección del carácter inalámbrico y la movilidad, donde encontraremos los desafíos técnicos más interesantes.

Comenzaremos ilustrando el entorno que vamos a emplear para nuestro análisis de la comunicación inalámbrica y la movilidad: una red en la que hay una serie de usuarios inalámbricos (y posiblemente móviles) conectados a otra infraestructura de red de mayor tamaño mediante un enlace inalámbrico situado en la frontera de la red. A continuación, pasaremos a estudiar las características de este enlace inalámbrico en la Sección 7.2. Incluimos una breve introducción al Acceso múltiple por división de código (CDMA, *Code Division Multiple Access*), un protocolo de acceso a un medio compartido que se emplea a menudo en las redes inalámbricas. Dicho estudio lo haremos también en la Sección 7.2. En la Sección 7.3, examinaremos con un cierto grado de profundidad los aspectos de nivel de enlace del estándar de red LAN inalámbrica IEEE 802.11 (WiFi); también dedicaremos unas pocas palabras a Bluetooth y a otras redes inalámbricas de área personal. En la Sección 7.4 veremos una panorámica del acceso celular a Internet, incluyendo las tecnologías celulares 3G y la emergente 4G que proporcionan tanto voz como acceso a Internet a alta velocidad. En la Sección 7.5 fijaremos nuestra atención en la movilidad, centrándonos en los problemas de localizar a un usuario móvil, efectuar el enrutamiento hasta ese usuario móvil e ir “transfiriendo” al usuario móvil, que se está desplazando dinámicamente desde un punto de conexión a la red hasta otro. Examinaremos cómo se implementan estos servicios móviles en el estándar de IP móvil en las redes empresariales 802.11 y en las redes celulares LTE en las Secciones 7.6 y 7.7, respectivamente. Por último, en la Sección 7.8, consideraremos el impacto de los enlaces inalámbricos y la movilidad sobre los protocolos de la capa de transporte y las aplicaciones en red.

7.1 Introducción

La Figura 7.1 muestra el escenario con el que vamos a analizar los temas de la comunicación inalámbrica de datos y la movilidad. Comenzaremos manteniendo nuestro estudio en un nivel lo suficientemente general como para cubrir un amplio rango de redes, incluyendo tanto las redes LAN inalámbricas (como por ejemplo IEEE 802.11) y las redes celulares (como una red 4G); en secciones posteriores profundizaremos en un análisis más detallado de determinadas arquitecturas inalámbricas más específicas. Dentro de una red inalámbrica podemos identificar los siguientes elementos:

- *Hosts inalámbricos.* Como en el caso de las redes cableadas, los hosts son los dispositivos que actúan como sistemas terminales y que ejecutan las aplicaciones. Un **host inalámbrico** puede ser una computadora portátil, una tableta, un teléfono inteligente o una computadora de escritorio. Los hosts en sí pueden ser móviles o no.
- *Enlaces inalámbricos.* Un host se conecta a una estación base (definida más adelante) o a otro host inalámbrico a través de un **enlace de comunicaciones inalámbrico**. Las diferentes tecnologías de enlace inalámbrico tienen distintas velocidades de transmisión y pueden transmitir a diferentes distancias. La Figura 7.2 muestra dos características clave (área de cobertura y velocidad del enlace) de los estándares más populares de redes inalámbricas. (La figura solo pretende proporcionar una idea aproximada de estas características. Por ejemplo, algunos de estos tipos de redes solo ahora se están comenzando a implantar y algunas velocidades de enlace pueden aumentar o disminuir respecto a los valores mostrados, dependiendo de la distancia, de las condiciones del canal y del número de usuarios en la red inalámbrica.) Nos ocuparemos

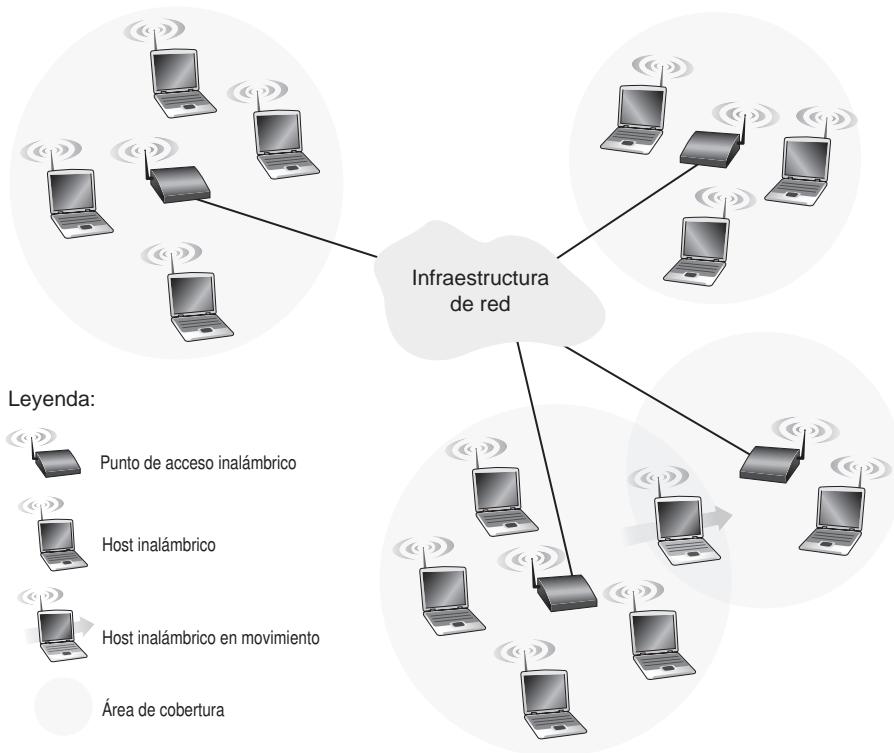


Figura 7.1 ♦ Elementos de una red inalámbrica.

de estos estándares más adelante a lo largo de la primera mitad de este capítulo; también consideraremos otras características de los enlaces inalámbricos, como sus tasas de error de bit y las causas de esos errores, en la Sección 7.2.

En la Figura 7.1 una serie de enlaces inalámbricos conectan a un conjunto de hosts inalámbricos ubicados en la frontera de la red con la infraestructura de esa red de mayor tamaño. Conviene añadir que los enlaces inalámbricos también se utilizan en ocasiones *dentro* de una red para conectar entre sí routers, switches y otros equipos de red. Sin embargo, nuestro enfoque en este capítulo se centrará en el uso de las comunicaciones inalámbricas en la frontera de la red, ya que es ahí donde podemos encontrar los desafíos técnicos más atractivos y donde se está experimentado un auténtico crecimiento.

- **Estación base.** La **estación base** es una parte clave de la infraestructura de la red inalámbrica. A diferencia del host inalámbrico y de los enlaces inalámbricos, una estación base no tiene un equivalente obvio dentro de las redes cableadas. La estación base es responsable de enviar y recibir datos (es decir, paquetes) hacia y desde un host inalámbrico que esté asociado con esa estación base. La estación base será a menudo responsable de coordinar la transmisión de los múltiples hosts inalámbricos que estén asociados con ella. Cuando decimos que un host inalámbrico está “asociado” con una estación base, queremos decir que (1) el host se encuentra dentro de la distancia de comunicación inalámbrica de la estación base y (2) el host utiliza la estación base para reenviar datos hacia y desde la red de mayor tamaño. Las **torres de telefonía** en las redes celulares y los **puntos de acceso** en las redes LAN inalámbricas 802.11 son ejemplos de estaciones base.

En la Figura 7.1, la estación base está conectada a la red de mayor tamaño (por ejemplo, a Internet, a una red doméstica o corporativa o a una red telefónica), funcionando así como retransmisor de la capa de enlace entre el host inalámbrico y el resto del mundo con el que el host se comunica.

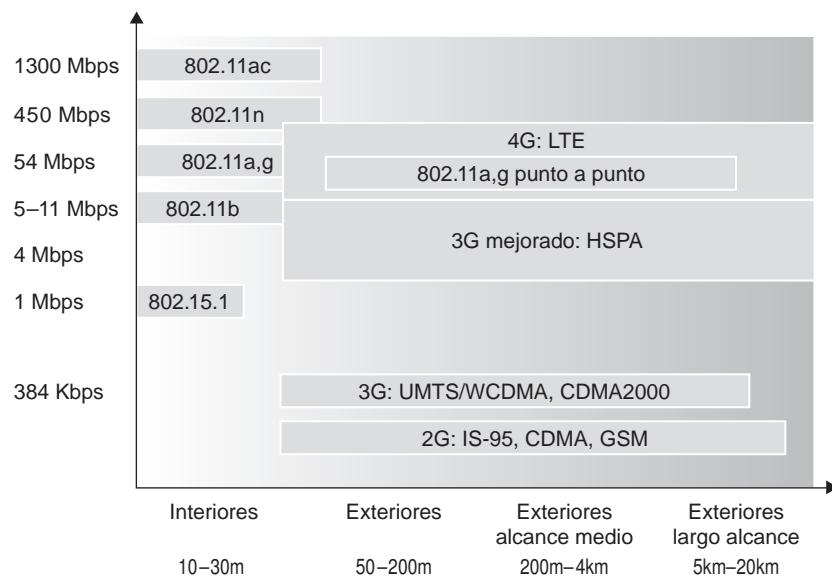


Figura 7.2 ♦ Características del enlace para una serie de estándares seleccionados de redes inalámbricas.

De los hosts asociados con una estación base se suele decir que operan en **modo de infraestructura**, puesto que todos los servicios de red tradicionales (como por ejemplo, la asignación de direcciones y el enrutamiento) son proporcionados por la red con la que un host se conecta a través de la estación base. En las **redes ad hoc**, los hosts inalámbricos no tienen ninguna infraestructura de ese tipo a la que conectarse. En ausencia de dicha infraestructura, los propios hosts tienen que proporcionar servicios tales como el enrutamiento, la asignación de direcciones, la traducción de nombres de tipo DNS, etc.

Cuando un host móvil se desplaza fuera del alcance de una estación base y entra dentro del área de cobertura de otra, cambia su punto de conexión con la red de mayor tamaño (es decir, cambia la estación base con la que está asociado); este proceso se conoce con el nombre de **transferencia (handoff)**. Este tipo de movilidad plantea numerosos y complejos problemas. Si un host puede moverse, ¿cómo podemos averiguar la ubicación actual del host móvil dentro de la red, para poder reenviar datos a ese host móvil? ¿Cómo se lleva a cabo el direccionamiento, sabiendo que un host puede estar en una de muchas posibles ubicaciones? Si el host se mueve *durante* una conexión TCP o llamada telefónica, ¿cómo se pueden enrutar los datos, para que la conexión continúe activa de forma ininterrumpida? Estas y otras muchas (¡muchísimas!) cuestiones hacen de las redes inalámbricas y móviles un área de investigación particularmente atractiva.

- *Infraestructura de red.* Es la red de mayor tamaño con la que un host inalámbrico puede querer comunicarse.

Habiendo examinado los elementos de una red inalámbrica, observemos que estos elementos pueden combinarse de muchas formas distintas para componer distintos tipos de redes inalámbricas. Conocer una taxonomía de estos tipos de redes inalámbricas puede ser útil a la hora de leer este capítulo o a la hora de leer o aprender más acerca de las redes inalámbricas más allá de lo que en este libro se expone. En el nivel más general, podemos clasificar las redes inalámbricas según dos criterios: (i) si un paquete dentro de la red inalámbrica realiza exactamente *un salto inalámbrico o varios saltos inalámbricos* y (ii) si existe una *infraestructura*, como por ejemplo una estación base, dentro de la red:



HISTORIA

ACCESO WIFI PÚBLICO: ¿ESTARÁ PRONTO DISPONIBLE EN LOS SEMÁFOROS?

Los puntos de acceso WiFi (*WiFi hotspots*, ubicaciones públicas en las que los usuarios pueden encontrar acceso inalámbrico 802.11) están siendo cada vez más comunes en los hoteles, aeropuertos y cafés de todo el mundo. La mayoría de los campus universitarios ofrecen un acceso inalámbrico ubicuo y resulta difícil encontrar un hotel en el que no exista acceso inalámbrico a Internet.

A lo largo de la última década diversas ciudades han diseñado, implantado y puesto en funcionamiento redes WiFi municipales. Resulta atractiva la idea de proporcionar a la comunidad un servicio público de acceso WiFi ubicuo (igual que se instalan semáforos), ayudando así a fomentar el crecimiento económico y a reducir la desigualdad digital, al proporcionar a todos los ciudadanos acceso a Internet. Muchas ciudades del mundo, entre las que se incluyen Filadelfia, Toronto, Hong Kong, Minneapolis, Londres y Auckland, han anunciado planes para proporcionar acceso inalámbrico ubicuo dentro de la ciudad, o han llevado a la práctica la idea en mayor o menor grado. El objetivo en Filadelfia era "convertir Filadelfia en el área de acceso WiFi más grande de Estados Unidos y ayudar a mejorar la educación, eliminar las diferencias sociales de carácter digital, mejorar el desarrollo de los barrios y reducir el coste de administración". El ambicioso programa, un acuerdo entre la ciudad, Wireless Philadelphia (una entidad sin ánimo de lucro) y Earthlink (un proveedor de servicios Internet), creó una red operativa de puntos de acceso 802.11b en semáforos y dispositivos de control de tráfico que cubría el 80 por ciento de la ciudad. Pero los problemas financieros y de funcionamiento forzaron a vender la red a un grupo de inversores privados en 2008, que posteriormente volvió a vender la red al ayuntamiento en 2010. Otras ciudades, como Minneapolis, Toronto, Hong Kong y Auckland, han tenido éxito con proyectos a menor escala.

El hecho de que las redes 802.11 operen en un espectro que no requiere licencia (y que, por tanto, puedan implantarse sin necesidad de adquirir carísimos derechos de uso del espectro) podría llevar a pensar que resultan atractivas desde el punto de vista financiero. Sin embargo, los puntos de acceso 802.11 (véase la Sección 7.3) tienen alcances mucho más cortos que las estaciones base celulares 4G (véase la Sección 7.4), lo que obliga a instalar un mayor número de puntos terminales para cubrir una misma región geográfica. Las redes de datos celulares que proporcionan acceso a Internet, por su parte, operan en una parte del espectro sujeta a licencia. Los operadores de redes celulares pagan miles de millones de dólares por los derechos de uso del espectro en sus redes, lo que hace de estas redes un negocio, en vez de un proyecto municipal.

- *Redes basadas en infraestructura y un único salto.* Estas redes tienen una estación base conectada a una red cableada de mayor tamaño (por ejemplo, Internet). Además, toda la comunicación se realiza entre esta estación base y un host inalámbrico con un único salto inalámbrico. Las redes 802.11 que utilizamos en las aulas, en las cafeterías o en las bibliotecas, las redes de datos LTE 4G de las que pronto hablaremos, caen todas ellas dentro de esta categoría. La gran mayoría de nuestras interacciones diarias las llevamos a cabo en redes inalámbricas basadas en infraestructura y un único salto.
- *Redes sin infraestructura y un único salto.* En estas redes no existe una estación base conectada a una red inalámbrica. Sin embargo, como veremos, uno de los nodos de esta red de un único salto puede coordinar las transmisiones de los restantes nodos. Las redes Bluetooth (que conectan pequeños dispositivos inalámbricos como teclados, altavoces y cascos, y que estudiaremos en la Sección 7.3.6) y las redes 802.11 en modo ad hoc son redes sin infraestructura y de un único salto.
- *Redes basadas en infraestructura y múltiples saltos.* En estas redes existe una estación base que está cableada a la red de mayor tamaño. Sin embargo, algunos nodos inalámbricos pueden tener

que retransmitir sus comunicaciones a través de otros nodos inalámbricos con el fin de comunicarse a través de la estación base. Algunas redes de sensores inalámbricos y las denominadas **redes de malla inalámbricas** caen dentro de esta categoría.

- *Redes sin infraestructura y múltiples saltos.* En estas redes no existe una estación base y los nodos pueden tener que retransmitir sus mensajes a través de otros diversos nodos para alcanzar un cierto destino. Los nodos también pueden ser móviles, con lo que la conectividad entre los nodos irá variando, lo que constituye una clase de redes conocidas con el nombre de **redes móviles ad hoc (MANET, Mobile ad hoc network)**. Si los nodos móviles son vehículos, la red se denomina **red vehicular ad hoc (VANET, Vehicular Ad hoc NETwork)**. Como puede imaginar, el desarrollo de protocolos para tales redes es enormemente complicado y es materia de muchas investigaciones que están actualmente en marcha.

En este capítulo vamos a limitarnos fundamentalmente a la redes de un único salto y, dentro de ellas, a las redes basadas en infraestructura.

Pero examinemos ahora con mayor profundidad los desafíos técnicos que surgen dentro de las redes inalámbricas y móviles. Comenzaremos considerando los enlaces inalámbricos individuales y dejando nuestro estudio acerca de las cuestiones de movilidad para más adelante dentro del capítulo.

7.2 Características de las redes y enlaces inalámbricos

Comencemos considerando una red cableada simple, por ejemplo una red doméstica, con un switch Ethernet cableado (véase la Sección 6.4) que interconecta los hosts. Si reemplazamos la Ethernet cableada por una red inalámbrica 802.11 tendríamos que sustituir las tarjetas Ethernet cableadas de los hosts por tarjetas NIC inalámbricas y cambiar el switch Ethernet por un punto de acceso inalámbrico, pero no haría falta prácticamente ningún cambio en la capa de red ni en las capas superiores. Esto sugiere que debemos centrar nuestra atención en la capa de enlace a la hora de buscar diferencias importantes entre las redes cableadas e inalámbricas. De hecho, podemos encontrar varias distinciones de importancia entre un enlace cableado y un enlace inalámbrico:

- *Intensidad decreciente de la señal.* La radiación electromagnética se atenúa a medida que va atravesando la materia (por ejemplo, una señal de radio que atraviesa una pared). Incluso en el espacio vacío la señal se dispersará, lo que da como resultado una intensidad de señal decreciente (en ocasiones denominada **pérdida de propagación, path loss**) a medida que se incrementa la distancia entre el emisor y el receptor.
- *Interferencias de otros orígenes.* Los orígenes de radio que transmiten en la misma banda de frecuencia interferirán entre sí. Por ejemplo, los teléfonos inalámbricos a 2,4 GHz y las redes LAN inalámbricas 802.11b transmiten en la misma banda de frecuencias. Por tanto, el usuario de una red LAN inalámbrica 802.11b que esté hablando a través de un teléfono inalámbrico a 2,4 GHz puede esperar que ni la red ni el teléfono tengan un comportamiento particularmente satisfactorio. Además de las interferencias de los orígenes de transmisión, el ruido electromagnético presente en el entorno (por ejemplo, un motor cercano o un microondas) también pueden provocar interferencias.
- *Propagación multicamino.* La **propagación multicamino (multipath)** tiene lugar cuando partes de la onda electromagnética se reflejan en los objetos y en el suelo, tomando caminos de diferentes longitudes entre un emisor y un receptor. Esto hace que la señal recibida sea menos limpia en el receptor. El desplazamiento de objetos situados entre el emisor y el receptor puede hacer que la propagación multicamino varíe a lo largo del tiempo.

Para ver una explicación detallada de las características, modelos y medidas de los canales inalámbricos, consulte [Anderson 1995].

La exposición anterior sugiere que los errores de bit serán más comunes en los enlaces inalámbricos que en los enlaces cableados. Por esta razón, no resulta sorprendente que los protocolos de enlace inalámbrico (como el protocolo 802.11 que examinaremos en la siguiente sección) no solo empleen potentes códigos CRC para la detección de errores, sino también protocolos de la capa de enlace con transferencia de datos fiable que se encargan de retransmitir las tramas corrompidas.

Habiendo considerado los problemas que pueden afectar a un canal inalámbrico, volvamos nuestra atención al host que recibe la señal inalámbrica. El host recibe una señal electromagnética que es una combinación de una forma degradada de la señal original transmitida por el emisor (degradada debido a los efectos de la atenuación y de la propagación multicamino que hemos visto anteriormente, entre otros) y del ruido de fondo presente en el entorno. La **relación señal-ruido (SNR, Signal-to-Noise Ratio)** es una medida relativa de la intensidad de la señal recibida (es decir, de la información que se está transmitiendo) y de este ruido. Normalmente, la relación señal-ruido se mide en unidades de decibelios (dB), que es una unidad de medida que algunos creen que los ingenieros eléctricos utilizan principalmente para confundir a los informáticos. La SNR, medida en dB, es veinte veces la relación del logaritmo en base 10 de la amplitud de la señal recibida y la amplitud del ruido. De cara a lo que aquí nos ocupa, lo único que necesitamos saber es que, cuanto mayor sea la relación SNR, más fácil le será al receptor extraer la señal transmitida del ruido de fondo.

La Figura 7.3 (adaptada de [Holland 2001]) muestra la tasa de errores de bit (BER, Bit Error Rate) que, dicho de forma simple, es la probabilidad de que un bit transmitido llegue de forma errónea al receptor, en función de la SNR para tres técnicas de modulación distintas utilizadas para codificar la información para su transmisión a través de un canal inalámbrico idealizado. La teoría de la modulación y de la codificación, así como las técnicas de extracción de la señal y de análisis de BER, caen fuera del alcance de este libro (consulte [Schwartz 1980] si desea obtener más información sobre estos temas). De todos modos, la Figura 7.3 ilustra varias características de la capa física que son importantes a la hora de comprender los protocolos de comunicación inalámbrica de las capas superiores.

- Para un determinado esquema de modulación, cuanto mayor es la SNR menor es la BER. Dado que un emisor puede incrementar la SNR aumentando su potencia de transmisión, podrá

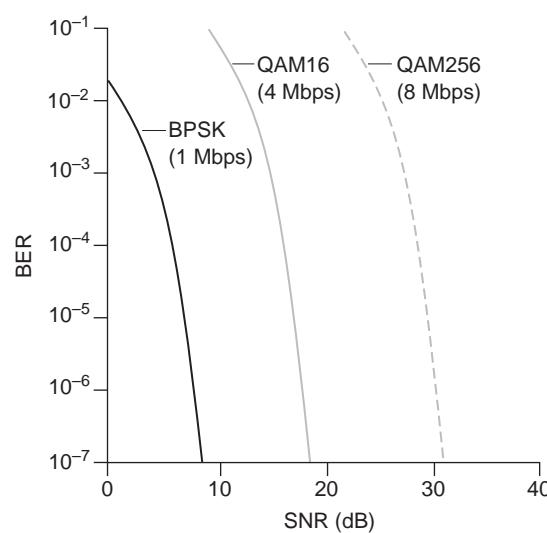


Figura 7.3 ♦ Tasa de errores de bit, velocidad de transmisión y SNR.

reducir la probabilidad de que una trama se reciba de forma errónea aumentando esa potencia de transmisión. Observe, sin embargo, que existe muy poca ventaja práctica cuando se incrementa la potencia más allá de un cierto umbral para, por ejemplo, reducir la tasa de errores de bit (BER) de 10^{-12} a 10^{-13} . También existen *desventajas* asociadas con ese incremento de la potencia de transmisión: el emisor tendrá que gastar más energía (lo que es una consideración de gran importancia para los usuarios móviles alimentados por baterías) y es más probable que las transmisiones del emisor interfieran con las de otros emisores (véase la Figura 7.4(b)).

- *Para una SNR dada, una técnica de modulación con una velocidad de transmisión de bit más alta (independientemente de si esos bits son erróneos o no) tendrá una tasa de errores de bit mayor.* Por ejemplo, en la Figura 7.3, con una SNR de 10 dB, la modulación BPSK con una velocidad de transmisión de 1 Mbps tiene una BER inferior a 10^{-7} , mientras que con la modulación QAM16, con una velocidad de transmisión de 4 Mbps, la BER es 10^{-1} , demasiado alta como para resultar útil en la práctica. Sin embargo, con una SNR de 20 dB, la modulación QAM16 tiene una velocidad de transmisión de 4 Mbps y una BER de 10^{-7} , mientras que la modulación BPSK tiene una velocidad de transmisión de solo 1 Mbps y una BER que es demasiado baja y cae (literalmente) “fuera del gráfico”. Si podemos tolerar una BER de 10^{-7} , la mayor velocidad de transmisión ofrecida por QAM16 haría que fuera la técnica de modulación preferida para esa situación. Estas consideraciones nos conducen a la característica final que describimos a continuación.
- *Puede utilizarse una selección dinámica de la técnica de modulación de la capa física para adaptar la técnica de modulación a las condiciones del canal.* La SNR (y por tanto la BER) puede variar como resultado de la movilidad o debido a cambios en el entorno. En los sistemas celulares de transmisión de datos y en las redes 802.11 WiFi y 4G, que estudiaremos en las Secciones 7.3 y 7.4, se usan técnicas de codificación y modulación adaptativas. Esto permite, por ejemplo, la selección de una técnica de modulación que proporcione la máxima velocidad de transmisión, sujeta a una restricción relativa a la BER, para unas determinadas características del canal.

Una tasa de errores de bit más alta y variable en el tiempo no es la única diferencia entre un enlace cableado y un enlace inalámbrico. Recuerde que en el caso de los enlaces cableados de difusión, todos los nodos reciben las transmisiones realizadas por los restantes nodos. En el caso de los enlaces inalámbricos, la situación no es tan simple como se muestra en la Figura 7.4. Suponga que la estación A está transmitiendo hacia la estación B. Suponga también que la estación C está transmitiendo hacia la estación B. Con el denominado **problema del terminal oculto**, las

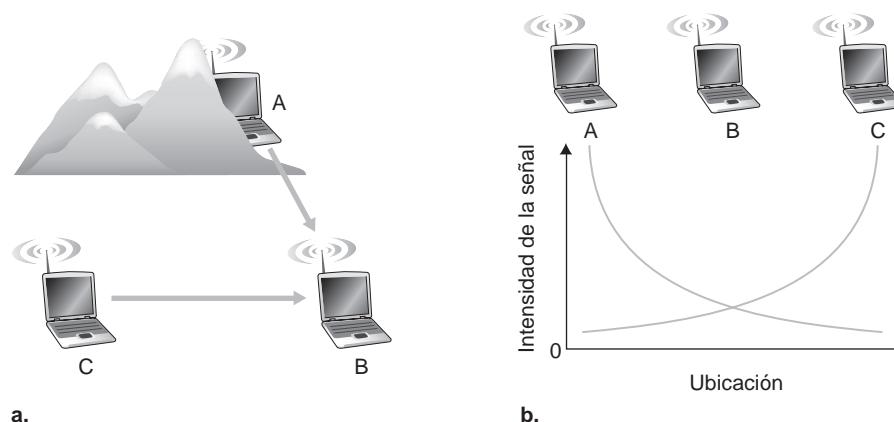


Figura 7.4 ♦ Problema del terminal oculto causado por la existencia de un obstáculo (a) y problema del desvanecimiento (b).

obstrucciones físicas presentes en el entorno (por ejemplo, una montaña o un edificio) pueden impedir que A y C escuchen las transmisiones del otro, incluso aún cuando las transmisiones de A y C estén interfiriéndose mutuamente en el destino B. Esto se muestra en la Figura 7.4(a). Un segundo escenario que da como resultado la presencia de colisiones indetectables en el receptor es el debido al **desvanecimiento** de la intensidad de la señal a medida que esta se propaga a través del medio inalámbrico. La Figura 7.4(b) ilustra el caso en el que A y C están colocadas de tal forma que sus señales no son lo suficientemente intensas como para que puedan ambas estaciones detectar las transmisiones de la otra, a pesar de lo cual esas señales *tienen* una intensidad suficiente como para interferir entre sí en la estación B. Como veremos en la Sección 7.3, los problemas del terminal oculto y del desvanecimiento hacen que el acceso múltiple en una red inalámbrica sea considerablemente más complejo que en una red cableada.

7.2.1 CDMA

Recuerde del Capítulo 6 que cuando los hosts se comunican a través de un medio compartido, se necesita un protocolo para que las señales enviadas por varios emisores no interfieran en los receptores. En el Capítulo 6 hemos descrito tres clases de protocolos de acceso al medio: particionamiento del canal, acceso aleatorio y toma de turnos. El Acceso múltiple por división de código (CDMA, *Code Division Multiple Access*) pertenece a la familia de protocolos de particionamiento del canal. Es el protocolo prevalente en las tecnologías celulares y de redes LAN inalámbricas. Puesto que CDMA es tan importante en el mundo inalámbrico, vamos a echar ahora un rápido vistazo a CDMA antes de entrar en las secciones posteriores a analizar tecnologías específicas del acceso inalámbrico.

En un protocolo CDMA, cada bit enviado se codifica multiplicándolo por una señal (el código) que varía a una velocidad mucho mayor (conocida con el nombre de **velocidad de chip**, *chipping rate*) que la secuencia original de bits de datos. La Figura 7.5 muestra un escenario idealizado de codificación/decodificación CDMA. Suponga que definimos la unidad de tiempo según la velocidad a la que llegan al codificador CDMA los bits de datos originales; es decir, cada bit original de datos que haya que transmitir requiere una partición de tiempo de un bit. Sea d_i el valor del bit de datos para la i -ésima partición de bit. Por comodidad matemática, vamos a representar los bits de datos que tengan un valor 0 como -1 . Cada partición de bit se subdivide a su vez en M mini-particiones; en la Figura 7.5 $M = 8$, aunque en la práctica M es mucho mayor. El código CDMA utilizado por el emisor está compuesto por una serie de M valores, c_m , $m = 1, \dots, M$, cada uno de los cuales tiene el valor $+1$ o -1 . En el ejemplo de la Figura 7.5, el código CDMA de M bits que está utilizando el emisor es $(1, 1, 1, -1, 1, -1, -1, -1)$.

Para ilustrar cómo funciona CDMA, vamos a centrarnos en el i -ésimo bit de datos, d_i . Para la m -ésima mini-partición del tiempo de transmisión de bit de d_i , la salida del codificador CDMA, $Z_{i,m}$, es el valor de d_i multiplicado por el m -ésimo bit del código asignado CDMA, c_m :

$$Z_{i,m} = d_i \cdot c_m \quad (7.1)$$

En un mundo ideal en el que no existieran otros emisores interfiriendo, el receptor recibiría los bits codificados, $Z_{i,m}$, y recuperaría el bit de datos original, d_i , realizando el cálculo:

$$d_i = \frac{1}{M} \sum_{m=1}^M Z_{i,m} \cdot c_m \quad (7.2)$$

El lector puede trabajar los detalles del ejemplo de la Figura 7.5 para ver que los bits de datos originales se recuperan, efectivamente, de modo correcto en el receptor utilizando la Ecuación 7.2.

Sin embargo, el mundo dista mucho de ser ideal y, como hemos dicho anteriormente, CDMA debe trabajar en presencia de otros emisores que interfieren y que están codificando y transmitiendo sus datos utilizando otro código asignado diferente. Pero, ¿cómo puede un receptor CDMA recuperar los bits de datos originales del emisor cuando esos bits de datos están entremezclados con los bits transmitidos por otros emisores? CDMA funciona bajo la suposición de que las señales interferentes

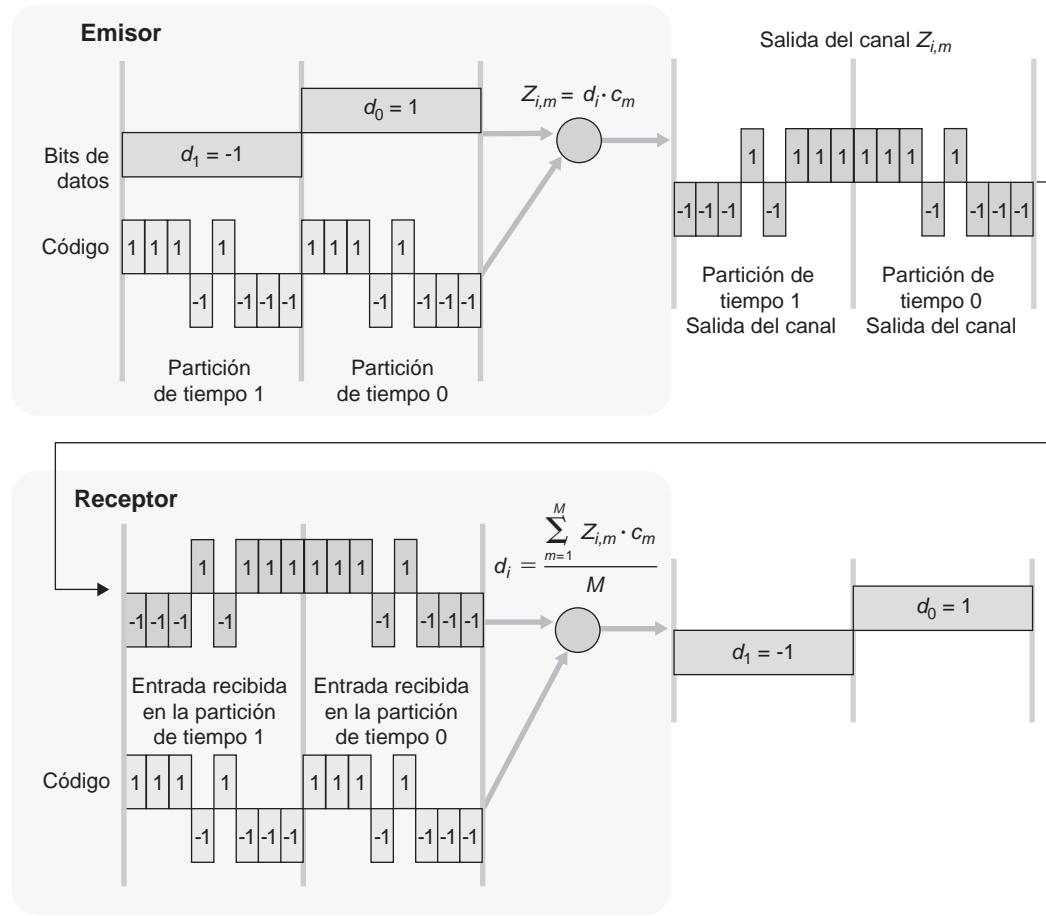


Figura 7.5 ♦ Un ejemplo simple de CDMA: codificación del emisor y decodificación del receptor.

de los bits transmitidos son aditivas. Esto significa, por ejemplo, que si los tres emisores envían un valor 1 y un cuarto emisor envía un valor -1 durante la misma minipartición, entonces la señal recibida en todos los receptores durante esa minipartición será un 2 (dado que $1 + 1 + 1 - 1 = 2$). En presencia de múltiples emisores, el emisor s calcula sus transmisiones codificadas, $Z_{i,m}^s$, exactamente de la misma forma que en la Ecuación 7.1. El valor que llega a un receptor durante la m -ésima minipartición de la i -ésima partición de bit será ahora, sin embargo, la *suma* de los bits transmitidos por los N emisores durante dicha minipartición:

$$Z_{i,m}^* = \sum_{s=1}^N Z_{i,m}^s$$

Sorprendentemente, si se eligen cuidadosamente los códigos de los emisores, cada receptor puede recuperar los datos enviados por un determinado emisor a partir de la señal agregada, simplemente utilizando el código del emisor exactamente en la misma forma que en la Ecuación 7.2:

$$d_i = \frac{1}{M} \sum_{m=1}^M Z_{i,m}^* \cdot c_m \quad (7.3)$$

como se muestra en la Figura 7.6 para un ejemplo de CDMA con dos emisores. El código CDMA de M bits utilizado por el emisor de la parte superior es $(1, 1, 1, -1, 1, -1, -1, -1)$, mientras que el código CDMA empleado por el emisor de la parte inferior es $(1, -1, 1, 1, 1, -1, 1, 1)$. La Figura 7.6

ilustra el caso de un receptor que recupera los bits de datos originales correspondientes al emisor de la parte superior. Observe que el receptor es capaz de extraer los datos del emisor 1 a pesar de que estén siendo interferidos por la transmisión correspondiente al emisor 2.

Recuerde la analogía del cóctel del Capítulo 6. Un protocolo CDMA es similar al caso en que los participantes en la reunión hablen en múltiples idiomas; en tales circunstancias, los seres humanos somos bastante buenos a la hora de centrarnos en la conversación que se está manteniendo en el idioma que comprendemos, al mismo tiempo que filtramos las restantes conversaciones. Podemos ver, a partir de estas explicaciones, que CDMA es un protocolo de particionamiento, en el sentido de que participa el espacio de códigos (en lugar del tiempo o la frecuencia) y asigna a cada nodo una parte dedicada de ese espacio de códigos.

El análisis que aquí hemos realizado de CDMA es necesariamente breve; en la práctica, es preciso contemplar diversas cuestiones relativamente complicadas. En primer lugar, para que los receptores CDMA sean capaces de extraer una señal de un emisor concreto, los códigos CDMA deben elegirse cuidadosamente. En segundo lugar, nuestro análisis partía de la suposición de que las

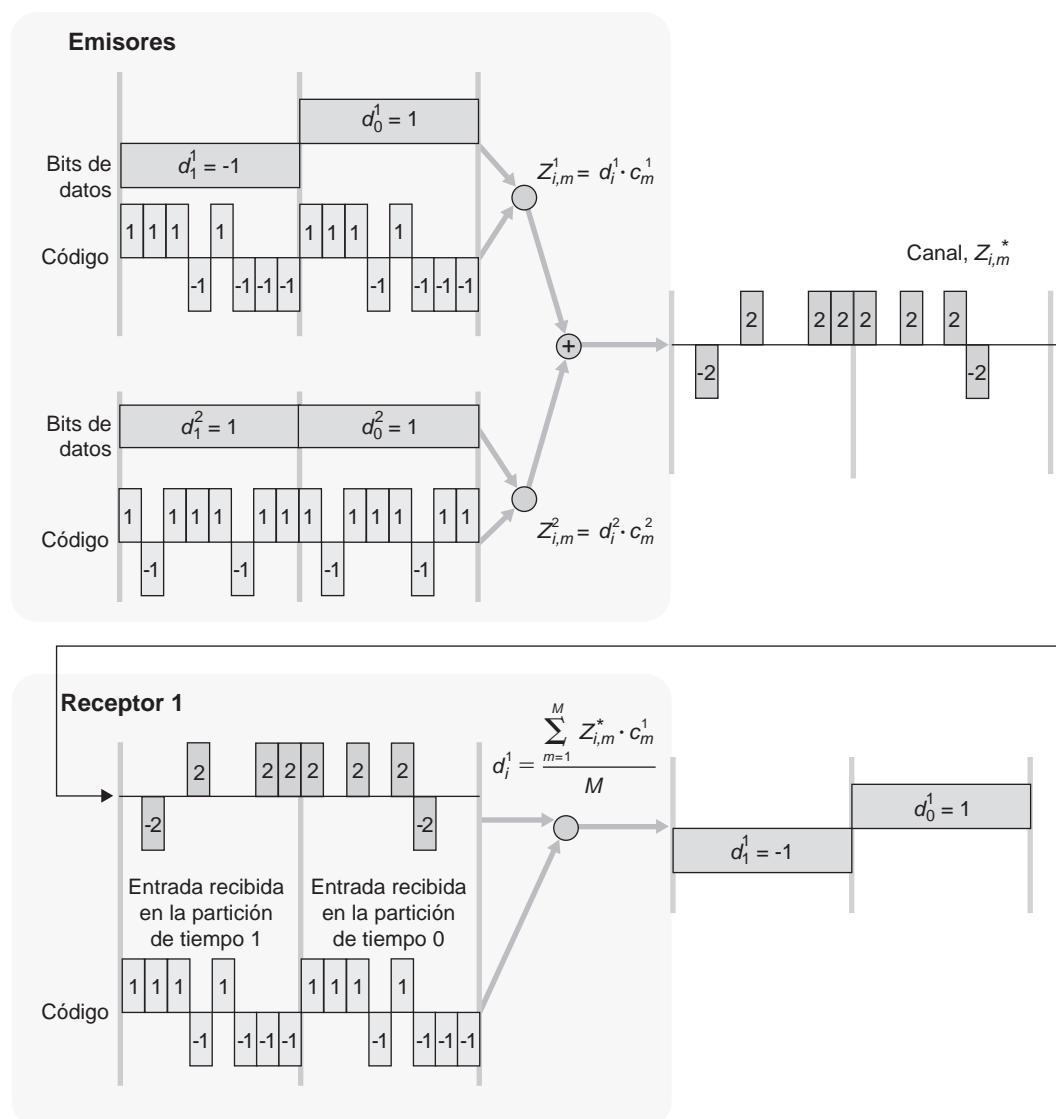


Figura 7.6 ♦ Un ejemplo de CDMA con dos emisores.

intensidades de las señales recibidas de los diversos emisores eran iguales, pero en la realidad esto puede ser difícil de conseguir. Existe bastante literatura técnica en la que se analizan estas y otras cuestiones relacionadas con CDMA; consulte [Pickholtz 1982; Viterbi 1995] para ver más detalles.

7.3 WiFi: redes LAN inalámbricas 802.11

Presentes por todas partes, en las oficinas, en los domicilios particulares, en las instituciones educativas, en la cafeterías, en los aeropuertos e incluso en la esquina de cualquier calle, las redes LAN inalámbricas son hoy en día una de las tecnologías más importantes de redes de acceso para Internet. Aunque en la década de 1990 se desarrollaron muchas tecnologías y estándares para redes LAN inalámbricas, hay una clase concreta de estándares que ha terminado por emerge como ganador indiscutible: la **red LAN inalámbrica IEEE 802.11**, también conocida como red **WiFi**. En esta sección vamos a examinar en detalle las redes LAN inalámbricas 802.11, su estructura de trama, su protocolo de acceso al medio y la interconexión de las redes LAN 802.11 con las redes LAN Ethernet cableadas.

Existen varios estándares 802.11 para la tecnología LAN inalámbrica en la familia IEEE 802.11 (“WiFi”), como se resume en la Tabla 7.1. Los diferentes estándares 802.11 comparten todos ellos algunas características comunes. Todos ellos emplean el mismo protocolo de acceso al medio, CSMA/CA, que vamos a ver brevemente. Además, los tres utilizan la misma estructura de trama para sus tramas de la capa de enlace. Asimismo, tienen la capacidad de reducir su velocidad de transmisión con el fin de alcanzar mayores distancias. Y, lo más importante, los productos 802.11 son también compatibles en sentido descendente, lo que significa, por ejemplo, que un móvil que usa solo el estándar 802.11g puede interactuar con una moderna estación base 802.11ac.

Sin embargo, como se muestra en la Tabla 7.1, los estándares presentan algunas diferencias importantes en la capa física. Los dispositivos 802.11 operan en dos rangos de frecuencia distintos: 2,4–2,4835 GHz (al que se hace referencia como rango de 2,4 GHz) y 5,1 – 5,8 GHz (el rango de 5 GHz). El rango de 2,4 GHz es una banda de frecuencias sin licencia, en la que los dispositivos 802.11 pueden competir por el espectro de frecuencias con los hornos microondas y teléfonos a 2,4 GHz. A 5 GHz, las redes LAN 802.11 proporcionan una distancia de transmisión más corta para un determinado nivel de potencia y se ven más afectadas por la propagación multicamino. Los dos estándares más recientes, 802.11n [IEEE 802.11n 2012] y 802.11ac [IEEE 802.11ac 2013; Cisco 802.11ac 2015] utilizan antenas MIMO (*Multiple Input Multiple-Output*, múltiples entradas y múltiples salidas); es decir, dos o más antenas en el lado emisor y dos o más antenas en el lado receptor que transmiten/reciben distintas señales [Diggavi 2004]. Las estaciones base 802.11ac pueden transmitir simultáneamente a varias estaciones y usan antenas “inteligentes” para conformar adaptativamente el haz, con el fin de dirigir las transmisiones hacia un determinado receptor. Esto hace que disminuya la interferencia y aumente la distancia alcanzada a una velocidad de datos dada. Las velocidades de datos mostradas en la Tabla 7.1 son para un entorno ideal, por ejemplo, un receptor colocado a 1 metro de la estación base, sin ninguna interferencia—un escenario muy improbable en la práctica! Como dice el refrán: una cosa es la teoría y otra la práctica (o, en este caso, nuestra velocidad de datos inalámbricos).

7.3.1 La arquitectura 802.11

La Figura 7.7 ilustra los principales componentes de la arquitectura de una red LAN inalámbrica 802.11. El componente fundamental de la arquitectura 802.11 es el **conjunto de servicio básico (BSS, Basic Service Set)**. Un BSS contiene una o más estaciones inalámbricas y una **estación base** central, conocida con el nombre de **punto de acceso (AP, Access Point)** en terminología 802.11. La Figura 7.7 muestra el punto de acceso en cada uno de los dos BSS; los puntos de acceso se interconectan a un dispositivo de interconexión (como un switch o un router), que a su vez lleva hacia

Estándar	Rango de frecuencias	Velocidad de datos
802.11b	2,4 GHz	hasta 11 Mbps
802.11a	5 GHz	hasta 54 Mbps
802.11g	2,4 GHz	hasta 54 Mbps
802.11n	2,5 GHz y 5 GHz	hasta 450 Mbps
802.11ac	5 GHz	hasta 1300 Mbps

Tabla 7.1 ♦ Resumen de los estándares IEEE 802.11.

Internet. En una red doméstica típica existirá un punto de acceso y un router (normalmente integrados en una misma unidad), que conectarán el BSS con Internet.

Al igual que sucede con los dispositivos Ethernet, cada estación inalámbrica 802.11 tiene una dirección MAC de 6 bytes que está almacenada en el firmware de la tarjeta adaptadora de la estación (es decir, en la tarjeta de interfaz de red 802.11). Cada punto de acceso tiene también una dirección MAC para su interfaz inalámbrica. Al igual que sucede con Ethernet, estas direcciones MAC son administradas por el IEEE y son (en teoría) globalmente únicas.

Como hemos dicho en la Sección 7.1, las redes LAN inalámbricas que incorporan puntos de acceso suelen denominarse **redes LAN inalámbricas de infraestructura**, siendo la “infraestructura” los puntos de acceso junto con la infraestructura de Ethernet cableada que interconecta los puntos de acceso y un router. La Figura 7.8 muestra que las estaciones IEEE 802.11 también se pueden agrupar para formar una red ad hoc: una red sin ningún control central y que no tiene conexiones con el “mundo exterior”. En este caso, la red es formada “sobre la marcha” por una serie de dispositivos móviles que se han encontrado con que están próximos entre sí, que tienen una necesidad de comunicarse y que no encuentran ninguna infraestructura de red preexistente en la ubicación en la que están. Una red ad hoc podría formarse cuando una serie de personas con computadoras portátiles se juntan (por ejemplo, en una sala de conferencias, en un tren o en un vehículo) y quieren intercambiar datos en ausencia de un punto acceso centralizado. Se ha generado un enorme interés en las redes ad hoc, ya que los dispositivos portátiles capaces de comunicarse entre sí continúan proliferando. En esta sección, sin embargo, centraremos la atención en las redes LAN inalámbricas de infraestructura.

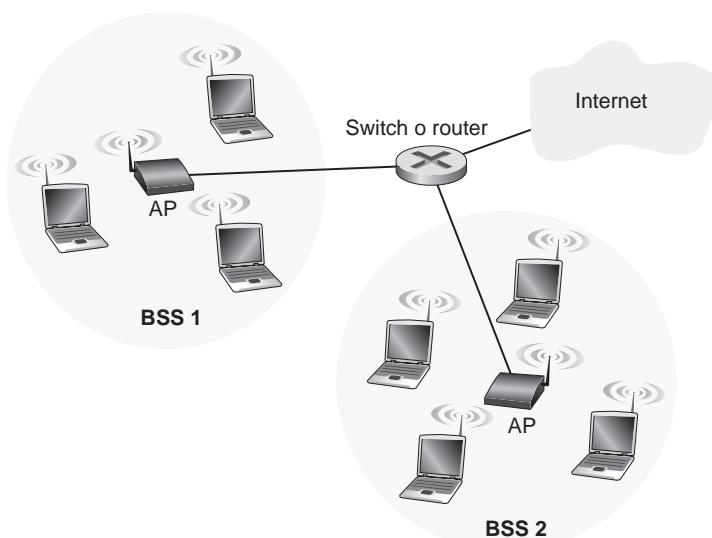


Figura 7.7 ♦ Arquitectura de una red LAN IEEE 802.11.

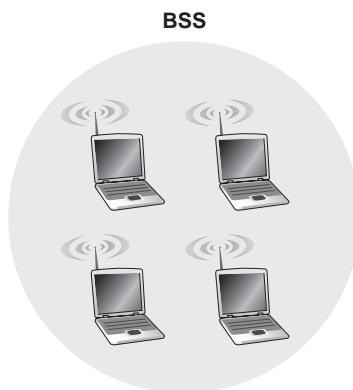


Figura 7.8 ♦ Una red ad hoc IEEE 802.11.

Canales y asociación

En la arquitectura 802.11 cada estación inalámbrica necesita asociarse con un punto de acceso antes de poder enviar o recibir datos de la capa de red. Aunque todos los estándares 802.11 utilizan el mecanismo de asociación, analizaremos el tema específicamente en el contexto de IEEE 802.11b/g.

Cuando un administrador de red instala un punto de acceso, asigna un **Identificador de conjunto de servicio (SSID, Service Set Identifier)** de una o dos palabras a ese punto de acceso. (Por ejemplo, cuando selecciona Wi-Fi en la configuración de su iPhone, aparece una lista que muestra el identificador SSID de cada punto de acceso que esté dentro del alcance.) El administrador debe también asignar un número de canal a ese punto de acceso. Para comprender los números de canal, recuerde que 802.11 opera en el rango de frecuencias de 2,4 GHz a 2,4835 GHz. Dentro de esta banda de 85 MHz, 802.11 define 11 canales parcialmente solapados. Dados dos canales cualesquiera diremos que no se solapan si y solo si están separados por cuatro o más canales. En particular, el conjunto de canales 1, 6 y 11 es el único conjunto de tres canales no solapados. Esto quiere decir que un administrador podría crear una red LAN inalámbrica con una velocidad máxima de transmisión agregada de 33 Mbps instalando tres puntos de acceso 802.11b en la misma ubicación física, asignando los canales 1, 6 y 11 a los puntos de acceso e interconectando todos los puntos de acceso mediante un switch.

Ahora que tenemos unos conocimientos básicos de los canales 802.11, vamos a describir una situación interesante (y que no resulta tan rara): la de la jungla WiFi. Una **jungla WiFi** es cualquier ubicación física en la que una estación inalámbrica está recibiendo una señal suficientemente intensa desde dos o más puntos de acceso. Por ejemplo, en muchas cafeterías de la ciudad de Nueva York una estación inalámbrica puede captar la señal de numerosos puntos de acceso cercanos. Uno de los puntos de acceso puede ser gestionado por la propia cafetería, mientras que los otros pueden encontrarse en viviendas situadas cerca de la misma. Cada uno de estos puntos de acceso estará, probablemente, ubicado en una subred IP diferente y se le habrá asignado un canal de manera independiente.

Suponga ahora que entramos en dicha jungla WiFi con nuestro teléfono, tableta o computadora portátil, buscando una taza de café y poder acceder a Internet de manera inalámbrica. Suponga que existen cinco puntos de acceso en esa jungla WiFi. Para poder obtener acceso a Internet, nuestro dispositivo inalámbrico necesita unirse a exactamente una de las subredes y, por tanto, necesitará **asociarse** con exactamente uno de los puntos de acceso. Asociarse quiere decir que el dispositivo inalámbrico creará un cable virtual entre él mismo y el punto de acceso. Específicamente, solo el punto de acceso asociado enviará tramas de datos (es decir, tramas que contienen datos, como por ejemplo un datagrama) a nuestro dispositivo inalámbrico y este enviará tramas de datos hacia Internet solamente a través del punto de acceso asociado. Pero, ¿cómo se asocia un dispositivo

inalámbrico con un punto de acceso concreto? Y todavía más importante: ¿cómo sabe un dispositivo inalámbrico qué puntos de acceso hay en esa jungla, si es que hay alguno?

El estándar 802.11 requiere que un punto de acceso envíe de forma periódica **tramas baliza** (*beacon frames*), cada una de las cuales incluye la dirección MAC y el identificador SSID del punto de acceso. El dispositivo inalámbrico, que sabe que los puntos de acceso están enviando tramas baliza, explora los once canales buscando las tramas baliza de cualquier punto de acceso que pueda haber en las proximidades (algunos de los cuales pueden estar transmitiendo a través del mismo canal, ya que estamos en una jungla). Habiendo determinado qué puntos de acceso hay disponibles a partir de las tramas baliza, seleccionamos (o nuestro dispositivo inalámbrico selecciona) uno de los puntos de acceso para llevar a cabo la asociación.

El estándar 802.11 no especifica un algoritmo para seleccionar con cuál de los puntos de acceso disponibles asociarse; dicho algoritmo se deja al arbitrio de los diseñadores del software y el firmware 802.11 del dispositivo inalámbrico. Normalmente, el dispositivo elige el punto de acceso cuya trama baliza se recibe con la máxima intensidad de señal. Pero, aunque una alta intensidad de señal resulta conveniente (vea, por ejemplo, la Figura 7.3), la intensidad de la señal no es la única característica del punto de acceso que influirá en el rendimiento que un dispositivo perciba. En particular, es posible que el punto de acceso seleccionado pueda tener una gran intensidad de señal, pero que pueda estar sobrecargado por otra serie de dispositivos asociados (que necesitarán compartir el ancho de banda inalámbrico disponible en dicho punto de acceso), mientras que se deja sin seleccionar un punto de acceso bastante descargado, debido a que la intensidad de la señal es ligeramente menor. Por esto, recientemente se han propuesto diversas formas de elección de los puntos de acceso [Vasudevan 2005; Nicholson 2006; Sudaresan 2006]. Para ver una interesante discusión práctica acerca de cómo se mide la intensidad de la señal, consulte [Bardwell 2004].

El proceso de exploración de los canales y de escucha de las tramas baliza se conoce con el nombre de **exploración pasiva** (véase la Figura 7.9a). Un dispositivo inalámbrico también puede realizar una **exploración activa**, difundiendo una trama de sondeo que será recibida por todos los puntos de acceso que caigan dentro del alcance del dispositivo inalámbrico, como se muestra en la Figura 7.9b. Los puntos de acceso responden a la trama de la solicitud de sondeo con una trama de respuesta de sondeo. El dispositivo inalámbrico puede entonces elegir el punto de acceso con el que asociarse de entre todos aquellos que hayan respondido.

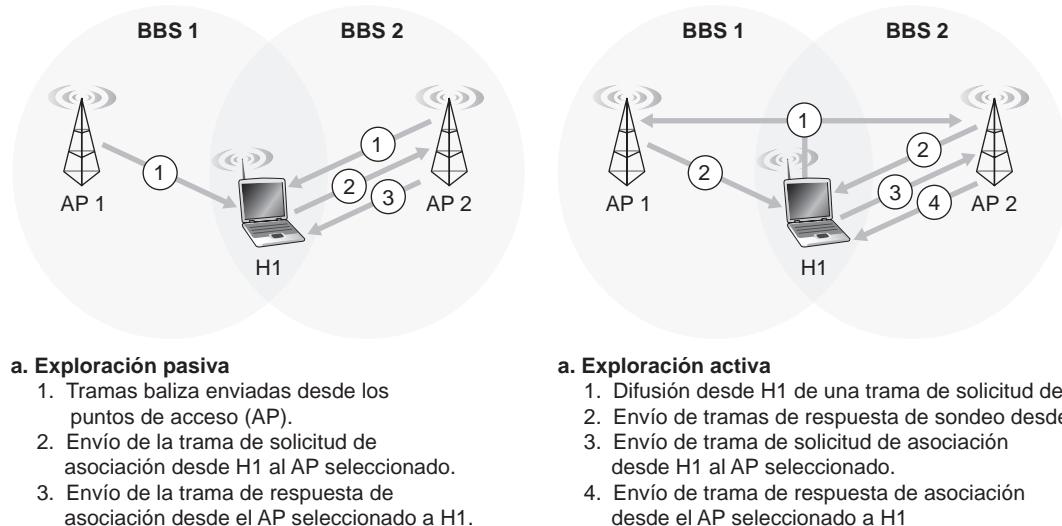


Figura 7.9 ♦ Exploración activa y pasiva de los puntos de acceso.

Después de seleccionar el AP con el que asociarse, el dispositivo inalámbrico envía una trama de solicitud de asociación a ese punto de acceso, el cual responde con una trama de respuesta de asociación. Observe que este segundo acuerdo de solicitud/respuesta también es necesario cuando se utiliza la exploración activa, dado que un punto de acceso que está respondiendo a la trama inicial de solicitud de sondeo no sabe cuál de los (posiblemente numerosos) puntos de acceso que hayan respondido va a seleccionar el host para asociarse, de la misma manera que un cliente DHCP puede seleccionar entre múltiples servidores DHCP (véase la Figura 4.21). Una vez asociado con un punto de acceso, el dispositivo se unirá a la subred (en el sentido de direccionamiento IP de la Sección 4.3.3) a la que pertenezca el punto de acceso. Normalmente el dispositivo enviará un mensaje de descubrimiento DHCP (véase la Figura 4.21) hacia la subred a través del punto de acceso para obtener una dirección IP de esa subred. Una vez obtenida la dirección, el resto del mundo verá entonces a dicho dispositivo simplemente como otro host cualquiera con una dirección IP perteneciente a dicha subred.

Para poder crear una asociación con un punto de acceso concreto, puede que el dispositivo inalámbrico tenga que autenticarse ante el punto de acceso. Las redes LAN inalámbricas 802.11 proporcionan diversas alternativas para la autenticación y el acceso. Una técnica, utilizada por muchas compañías, permite el acceso a una red inalámbrica basándose en la dirección MAC del dispositivo. Una segunda técnica, empleada en muchos cafés Internet, utiliza nombres de usuario y contraseñas. En ambos casos, el punto de acceso se comunica normalmente con un servidor de autenticación, reenviando la información intercambiada entre el dispositivo inalámbrico y el servidor de autenticación utilizando un protocolo como RADIUS [RFC 2865] o DIAMETER [RFC 3588]. Separar el servidor de autenticación del punto de acceso permite que un único servidor de autenticación proporcione servicio a muchos puntos de acceso distintos, centralizando las (a menudo sensibles) decisiones relativas a la autenticación y al acceso en ese único servidor y manteniendo los costes y la complejidad de los puntos de acceso relativamente bajos. Veremos en el Capítulo 8 que el nuevo protocolo IEEE 802.11i, que define los aspectos de seguridad de la familia de protocolos 802.11, adopta precisamente este enfoque.

7.3.2 El protocolo MAC 802.11

Una vez asociado un dispositivo inalámbrico con un punto de acceso, el dispositivo puede comenzar a enviar y recibir tramas de datos hacia y desde el punto de acceso. Pero, dado que múltiples dispositivos inalámbricos, o el propio punto de acceso, pueden desear transmitir tramas de datos al mismo tiempo a través del mismo canal, es preciso utilizar un protocolo de acceso múltiple para coordinar esas transmisiones. De aquí en adelante vamos a referirnos a los dispositivos o a los puntos de acceso como “estaciones” inalámbricas que comparten el canal de acceso múltiple. Como se explica en el Capítulo 6 y en la Sección 7.2.1, en términos generales existen tres clases de protocolos de acceso múltiple: particionamiento del canal (incluyendo CDMA), acceso aleatorio y toma por turnos. Inspirados por el enorme éxito de Ethernet y su protocolo de acceso aleatorio, los diseñadores de la arquitectura 802.11 seleccionaron un protocolo de acceso aleatorio para las redes LAN inalámbricas 802.11. Este protocolo de acceso aleatorio se conoce como **CSMA con evitación de colisiones** o, más sucintamente, **CSMA/CA (Collision Avoidance)**. Al igual que sucede con el protocolo CSMA/CD de Ethernet, las siglas “CSMA” en CSMA/CA hacen referencia al “acceso múltiple por sondeo de portadora”, lo que significa que cada estación sondea el canal antes de transmitir y se abstiene de transmitir cuando detecta que el canal está ocupado. Aunque tanto Ethernet como 802.11 utilizan acceso aleatorio con sondeo de portadora, los dos protocolos MAC presentan diferencias importantes. En primer lugar, en vez de utilizar detección de colisiones, 802.11 utiliza técnicas de evitación de las colisiones. En segundo lugar, debido a las relativamente altas tasas de errores de bit de los canales inalámbricos, 802.11 (a diferencia de Ethernet) utiliza un esquema de reconocimiento/retransmisión (ARQ) de la capa de enlace. Más adelante se describen los esquemas de evitación de las colisiones y de reconocimiento en la capa de enlace de 802.11.

Recuerde de las Secciones 6.3.2 y 6.4.2 que con el algoritmo de detección de colisiones de Ethernet una estación Ethernet escucha el canal a medida que transmite. Si detecta mientras está transmitiendo que hay otra estación transmitiendo también, aborta su transmisión y trata de transmitir de nuevo después de esperar un periodo de tiempo pequeño y aleatorio. A diferencia del protocolo Ethernet 802.3, el protocolo MAC 802.11 *no* implementa ningún mecanismo de detección de colisiones. Hay dos razones importantes para esto:

- La capacidad de detectar colisiones requiere la capacidad de enviar (la propia señal de la estación) y de recibir (para determinar si otra estación también está transmitiendo) al mismo tiempo. Puesto que la intensidad de la señal recibida es normalmente muy pequeña si la comparamos con la intensidad de la señal transmitida por el adaptador 802.11, resulta muy costoso construir un hardware que pueda detectar una colisión.
- Todavía más importante es que, incluso si el adaptador pudiera transmitir y escuchar al mismo tiempo (y supuestamente abortar la transmisión cuando detecte que el canal está ocupado), el adaptador seguiría sin ser capaz de detectar todas las colisiones, debido a los problemas del terminal oculto y del desvanecimiento que hemos explicado en la Sección 7.2.

Puesto que las redes LAN inalámbricas 802.11 no utilizan la detección de colisiones, una vez que una estación empieza a transmitir una trama la transmite en su totalidad; es decir, una vez que una estación comienza a transmitir no hay vuelta atrás. Como cabría esperar, la transmisión de tramas completas (en especial, tramas largas) cuando las colisiones son abundantes puede degradar significativamente el rendimiento de un protocolo de acceso múltiple. Para reducir la probabilidad de colisión, 802.11 emplea varias técnicas de evitación de colisiones que explicaremos en breve.

Sin embargo, antes de entrar en el tema de la evitación de las colisiones, necesitamos examinar en primer lugar el esquema de **reconocimiento de la capa de enlace** en 802.11. Recuerde de la Sección 7.2 que, cuando una estación en una LAN inalámbrica envía una trama, esta puede no llegar intacta a la estación de destino por diversas razones. Para resolver esta probabilidad no desdeñable de fallo, el protocolo MAC 802.11 utiliza reconocimientos de la capa de enlace. Como se muestra en la Figura 7.10, cuando la estación de destino recibe una trama que pasa la prueba de comprobación de CRC, espera un corto periodo de tiempo conocido con el nombre de **espaciado corto entre tramas (SIFS, Short Inter-frame Spacing)** y luego devuelve una trama de reconocimiento. Si la estación transmisora no recibe una trama de reconocimiento dentro de un periodo de tiempo especificado, supone que se ha producido un error y retransmite la trama, utilizando el protocolo CSMA/CA para acceder al canal. Si no se recibe una trama de reconocimiento después de un número fijo de retransmisiones, la estación transmisora se da por vencida y descarta la trama.

Habiendo explorado cómo emplea 802.11 los reconocimientos de la capa de enlace, estamos en disposición de describir el protocolo CSMA/CA 802.11. Suponga que una estación (estación inalámbrica o punto de acceso) dispone de una trama para transmitir.

1. Si la estación detecta inicialmente que el canal está inactivo, transmite la trama después de un corto periodo de tiempo, conocido con el nombre de **espacio distribuido entre tramas (DIFS, Distributed Inter-frame Space)**; véase la Figura 7.10.
2. En caso contrario, la estación selecciona un valor de espera (*backoff*) aleatorio (como hemos visto en la Sección 6.3.2) y efectúa una cuenta atrás con este valor desde DIFS mientras detecta que el canal está inactivo. Cuando detecta que el canal está ocupado, el valor del contador permanece congelado.
3. Cuando el contador alcanza el valor cero (observe que esto solo puede suceder mientras se detecta que el canal está inactivo), la estación transmite la trama completa y luego espera a recibir un reconocimiento.
4. Si se recibe una trama de reconocimiento, la estación transmisora sabe que su trama ha sido recibida correctamente en la estación de destino. Si la estación tiene otra trama que enviar, comienza de nuevo el protocolo CSMA/CA en el paso 2. Si no se recibe una trama de recono-

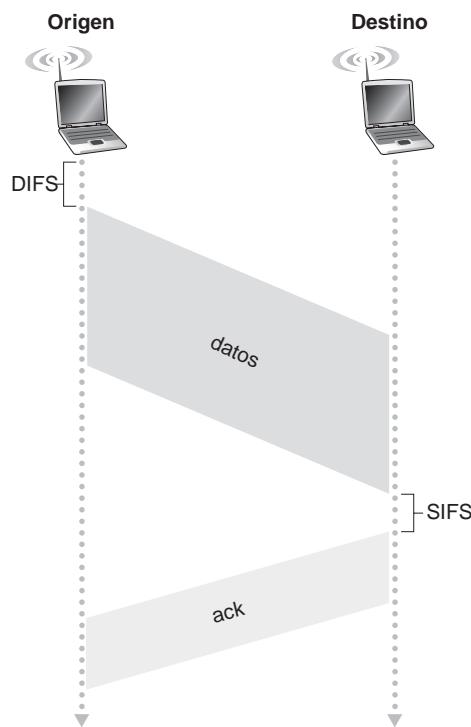


Figura 7.10 ♦ 802.11 utiliza reconocimientos (ACK) de la capa de enlace.

cimiento, la estación transmisora vuelve a entrar en la fase de *backoff* del paso 2, seleccionando el valor aleatorio de un intervalo más largo.

Recuerde que con el protocolo de acceso múltiple CSMA/CD de Ethernet (Sección 6.3.2) una estación comienza a transmitir en cuanto detecta que el canal está inactivo. Sin embargo, con CSMA/CA la estación se abstiene de transmitir mientras efectúa la cuenta atrás, aún cuando detecte que no hay actividad en el canal. ¿Por qué CSMA/CD y CSMA/CA adoptan enfoques tan distintos en este punto?

Para responder a esta cuestión, consideremos un escenario en el que hay dos estaciones, cada una de ellas con una trama para transmitir, pero ninguna de las estaciones transmite inmediatamente, porque ambas detectan que hay una tercera estación que ya está transmitiendo. Con CSMA/CD de Ethernet, ambas estaciones transmitirían tan pronto como detectaran que la tercera estación ha dejado de transmitir. Esto provocaría una colisión, lo cual no es un serio problema en CSMA/CD, ya que ambas estaciones abortarían sus transmisiones y por tanto evitarían transmitir inútilmente el resto de sus tramas. Sin embargo, en 802.11 la situación es muy distinta. Puesto que 802.11 no detecta las colisiones y no aborta en consecuencia las transmisiones, la trama que sufre una colisión será transmitida en su totalidad. Por tanto, el objetivo de 802.11 es evitar las colisiones siempre que sea posible. En 802.11, si las dos estaciones detectan que el canal está ocupado ambas entran inmediatamente en un estado de espera aleatoria, con lo que cabe esperar que ambas seleccionen valores de espera (*backoff*) diferentes. Si dichos valores son verdaderamente distintos, una vez que el canal pase a estar inactivo una de las dos estaciones empezará a transmitir antes que la otra y (si las dos estaciones no están ocultas a ojos una de otra) la “estación perdedora” escuchará la señal de la estación “ganadora”, congelará su cuenta atrás y se abstendrá de transmitir hasta que la estación ganadora haya completado su transmisión. De esta forma se evita una costosa colisión. Por supuesto, en 802.11 siguen pudiéndose producir colisiones con este escenario: las dos estaciones podrían estar

ocultas a ojos una de otra o las dos estaciones podrían seleccionar valores de espera aleatorios lo suficientemente próximos como para que la transmisión procedente de la estación que comience primero pueda alcanzar a la segunda estación. Recuerde que ya nos hemos encontrado anteriormente con este problema al hablar de los algoritmos de acceso aleatorio en el contexto de la Figura 6.12.

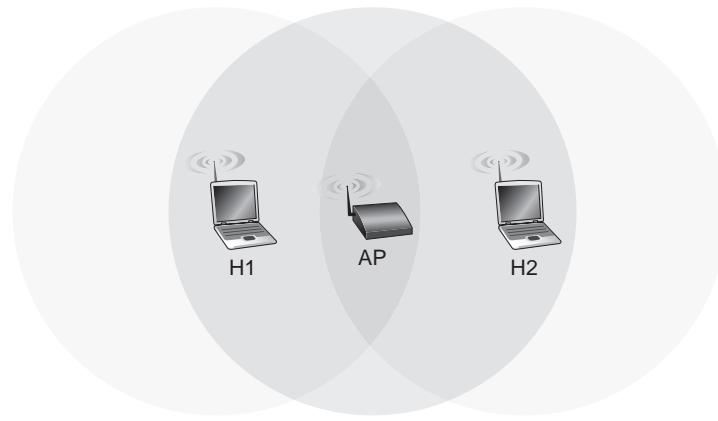
Enfrentándose al problema de los terminales ocultos: RTS y CTS

El protocolo MAC 802.11 también incluye un excelente (pero opcional) esquema de reserva, que ayuda a evitar las colisiones incluso en presencia de terminales ocultos. Vamos a analizar este esquema en el contexto de la Figura 7.11, que muestra dos estaciones inalámbricas y un punto de acceso. Ambas estaciones inalámbricas caen dentro del alcance del punto de acceso (cuya área de cobertura se muestra como un círculo sombreado) y ambas están asociadas con el punto de acceso. Sin embargo, debido al desvanecimiento, los rangos de señal de las estaciones inalámbricas están limitados al interior de los círculos sombreados de la Figura 7.11. En consecuencia, cada una de las estaciones inalámbricas está oculta a ojos de la otra, aunque ninguna de las dos está oculta para el punto de acceso.

Consideremos ahora por qué los terminales ocultos pueden ser problemáticos. Suponga que la estación H1 está transmitiendo una trama y que, en mitad de la transmisión de H1, la estación H2 quiere enviar una trama al punto de acceso. H2, al no escuchar la transmisión de H1, esperará primero un intervalo DIFS y luego transmitirá la trama, provocando una colisión. Por tanto, el canal se desperdiciará durante el periodo completo de transmisión de H1, así como durante la transmisión de H2.

Para evitar este problema, el protocolo IEEE 802.11 permite a una estación utilizar una corta trama de control de **solicitud de transmisión (RTS, Request to Send)** y otra corta trama de control de **preparado para enviar (CTS, Clear to Send)** para *reservar* el acceso al canal. Cuando un emisor quiere enviar una trama DATA, puede enviar primero una trama RTS al punto de acceso, indicando el tiempo total requerido para transmitir la trama DATA y la trama de reconocimiento (ACK). Cuando el punto de acceso recibe la trama RTS, responde difundiendo una trama CTS. Esta trama CTS sirve a dos propósitos distintos: proporciona al emisor un permiso explícito para enviar y también informa a las otras estaciones de que no deben transmitir durante ese periodo de tiempo reservado.

Por tanto, en la Figura 7.12, antes de transmitir una trama DATA, H1 difunde primero una trama RTS, que será escuchada por todas las estaciones situadas en su área de cobertura, incluyendo al punto de acceso. A continuación, el punto de acceso responde con una trama CTS, que será escuchada por todas las estaciones dentro de su área de cobertura, incluyendo a H1 y H2. La estación H2, habiendo escuchado la trama CTS, se abstendrá de transmitir durante el tiempo especificado en la trama CTS. Las tramas RTS, CTS, DATA y ACK se muestran en la Figura 7.12.



Descargado en: eybooks.com

Figura 7.11 ♦ Ejemplo de terminales ocultos: H1 está oculto a ojos de H2 y viceversa.

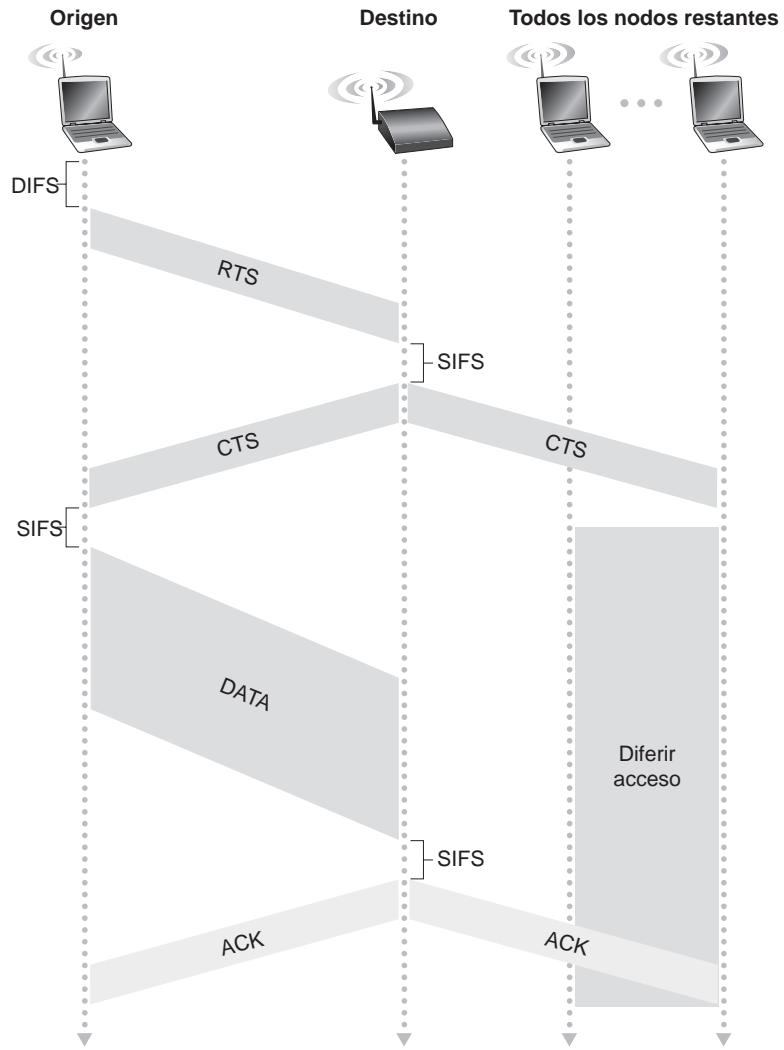


Figura 7.12 ♦ Evitación de colisiones utilizando las tramas RTS y CTS.

El uso de las tramas RTS y CTS puede mejorar el rendimiento de dos formas importantes:

- El problema de las estaciones ocultas queda mitigado, ya que una trama DATA larga solo se transmitirá después de haber reservado el canal.
- Puesto que las tramas RTS y CTS son cortas, una colisión que implique a una trama RTS o CTS solo durará mientras duren esas tramas cortas RTS o CTS. Una vez transmitidas las tramas RTS y CTS correctamente, las tramas DATA y ACK siguientes deberían poder transmitirse sin colisiones.

Le animamos a comprobar el funcionamiento del applet 802.11 disponible en el sitio web del libro. Este applet interactivo ilustra el protocolo CSMA/CA, incluyendo la secuencia de intercambio RTS/CTS.

Aunque el intercambio RTS/CTS puede ayudar a reducir las colisiones, también introduce un retardo y consume recursos del canal. Por esta razón, el intercambio RTS/CTS solamente se utiliza (si es que se utiliza en absoluto) para reservar el canal para la transmisión de una trama DATA larga. En la práctica, cada estación inalámbrica puede establecer un umbral RTS, de modo que la secuencia

RTS/CTS se utilice únicamente cuando la trama que hay que transmitir sea mayor que el umbral. Para muchas estaciones inalámbricas, el valor umbral RTS predeterminado es mayor que la longitud máxima de trama, por lo que la secuencia RTS/CTS se omite para todas las tramas DATA enviadas.

Utilización de 802.11 como un enlace punto a punto

Nuestro análisis se ha centrado hasta el momento en el uso de 802.11 en una configuración de acceso múltiple. Es necesario mencionar que si hay dos nodos, cada uno de los cuales dispone de una antena direccional, ambos pueden apuntar sus antenas hacia el otro nodo y ejecutar el protocolo 802.11 sobre lo que es, esencialmente, un enlace punto a punto. Dado el bajo coste del hardware 802.11 comercial, la utilización de antenas direccionales y una potencia de transmisión incrementada permiten utilizar 802.11 como un medio barato de proporcionar conexiones punto a punto inalámbricas a distancias de decenas de kilómetros. [Raman 2007] describe dichas redes inalámbricas multisalto operando en la llanuras rurales del Ganges en la India usando enlaces 802.11 punto a punto.

7.3.3 La trama IEEE 802.11

Aunque la trama 802.11 comparte muchas similitudes con una trama Ethernet, también contiene diversos campos que son específicos para su uso en enlaces inalámbricos. La trama 802.11 se muestra en la Figura 7.13. Los números situados encima de cada campo de la trama representan las longitudes de los campos en *bytes*; los números situados por encima de cada uno de los subcampos en el campo de control de trama representan las longitudes de los subcampos en *bits*. Examinemos ahora los campos de la trama, así como algunos de los subcampos más importantes del campo de control de trama.

Campos de carga útil y CRC

En el corazón de la trama se encuentra la carga útil, que normalmente estará compuesta por un datagrama IP o un paquete ARP. Aunque el campo puede tener una longitud de hasta 2.312 bytes, normalmente la longitud es inferior a 1.500 bytes, conteniendo el campo un datagrama IP o un paquete ARP. Al igual que con una trama Ethernet, una trama 802.11 incluye un código de redundancia cíclica (CRC) de 32 bits, de modo que el receptor pueda detectar errores de bit en la trama recibida. Como ya hemos visto, los errores de bit son mucho más comunes en las redes LAN inalámbricas que en las redes LAN cableadas, por lo que el CRC aquí es todavía más útil.

Campos de dirección

Quizá la diferencia más llamativa en la trama 802.11 es que tiene *cuatro* campos de dirección, cada uno de los cuales puede contener una dirección MAC de 6 bytes. ¿Pero por qué se utilizan cuatro campos de dirección? ¿No bastaría con un campo MAC de origen y un campo MAC de destino, como sucede en Ethernet? Resulta que tres de los campos de dirección son necesarios para propósitos de la comunicación por la red, específicamente para mover el datagrama de la capa de red de una estación inalámbrica hasta una interfaz de router a través de un punto de acceso. El cuarto campo de dirección se utiliza cuando los puntos de acceso se reenvían tramas entre sí en modo ad hoc. Puesto que solo estamos considerando aquí las redes de infraestructura, vamos a centrarnos en los tres primeros campos de dirección. El estándar 802.11 define estos campos como sigue:

- Dirección 2 es la dirección MAC de la estación que transmite la trama. Por tanto, si una estación inalámbrica transmite la trama, la dirección MAC de dicha estación se insertará en el campo Dirección 2. De forma similar, si es un punto de acceso el que transmite la trama, la dirección MAC de dicho punto de acceso se insertará en el campo Dirección 2.
- El campo Dirección 1 contiene la dirección MAC de la estación inalámbrica que tiene que recibir la trama. Por tanto, si una estación inalámbrica móvil transmite la trama, Dirección 1

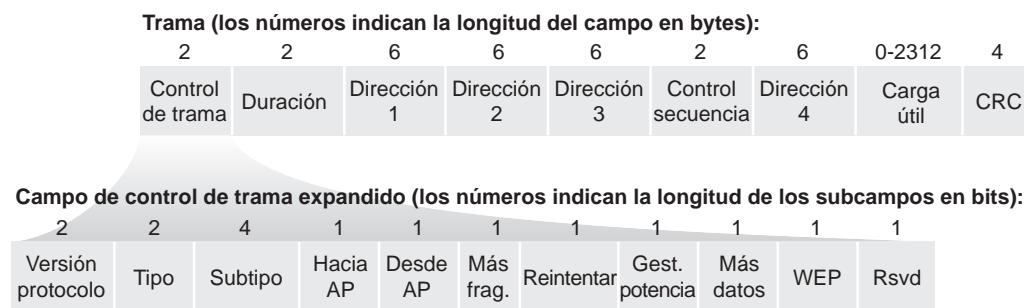


Figura 7.13 ♦ La trama 802.11.

contendrá la dirección MAC del punto de acceso de destino. De forma similar, si un punto de acceso transmite la trama, Dirección 1 contendrá la dirección MAC de la estación inalámbrica de destino.

- Para comprender el campo Dirección 3, recuerde que el BSS (que consta del punto de acceso y las estaciones inalámbricas) forma parte de una subred y que dicha subred se conecta a otras subredes a través de alguna interfaz de router. El campo Dirección 3 contiene la dirección MAC de esa interfaz de router.

Para entender mejor el propósito del campo Dirección 3, veamos un ejemplo de comunicación por red en el contexto de la Figura 7.14. En esta figura hay dos puntos de acceso, cada uno de los cuales es responsable de una serie de estaciones inalámbricas. Cada uno de los puntos de acceso tiene una conexión directa con un router, el cual a su vez se conecta a la red Internet global. Debemos recordar que un punto de acceso es un dispositivo de la capa de enlace y que, por tanto, nunca “habla” IP ni comprende las direcciones IP. Considere ahora el proceso de transferir un datagrama desde la interfaz del router R1 hasta la estación inalámbrica H1. El router no es consciente de que existe un punto de acceso entre él y H1; desde la perspectiva del router, H1 es simplemente un host en una de las subredes a la que el router está conectado.

- El router, que conoce la dirección IP de H1 (a partir de la dirección de destino del datagrama), utiliza ARP para determinar la dirección MAC de H1, al igual que en una red LAN Ethernet normal. Después de obtener la dirección MAC de H1, la interfaz del router R1 encapsula el datagrama dentro de una trama Ethernet. El campo de dirección de origen de esta trama contiene la dirección MAC de R1 y el campo de la dirección de destino contiene la dirección MAC de H1.
- Cuando la trama Ethernet llega al punto de acceso, este convierte la trama Ethernet 802.3 en una trama 802.11 antes de transmitirla por el canal inalámbrico. El punto de acceso rellena los campos Dirección 1 y Dirección 2 con la dirección MAC de H1 y su propia dirección MAC, respectivamente, como hemos descrito anteriormente. Como Dirección 3, el punto de acceso inserta la dirección MAC de R1. De esta manera, H1 puede determinar (a partir de la dirección 3) la dirección MAC de la interfaz del router que envió el datagrama hacia la subred.

Considere ahora lo que sucede cuando la estación inalámbrica H1 responde transfiriendo un datagrama desde H1 a R1.

- H1 crea una trama 802.11, llenando los campos Dirección 1 y Dirección 2 con la dirección MAC del punto de acceso y la de H1, respectivamente, como hemos descrito anteriormente. Como Dirección 3, H1 inserta la dirección MAC de R1.
- Cuando el punto de acceso recibe la trama 802.11, la convierte en una trama Ethernet. El campo de dirección de origen de esta trama será la dirección MAC de H1, mientras que el campo de

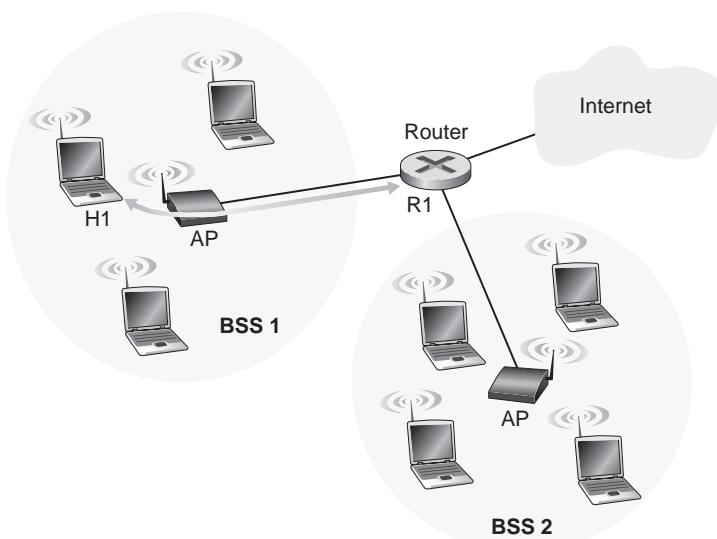


Figura 7.14 ♦ Uso de los campos de dirección en las tramas 802.11: envío de tramas entre H1 y R1.

dirección de destino será la dirección MAC de R1. Por tanto, el campo Dirección 3 permite al punto de acceso determinar la dirección MAC de destino apropiada a la hora de construir la trama Ethernet.

En resumen, el campo Dirección 3 desempeña un papel crucial para la comunicación por red entre el BSS y una red LAN cableada.

Campos Número de secuencia, Duración y Control de trama

Recuerde que en 802.11, cada vez que una estación recibe correctamente una trama procedente de otra estación devuelve un mensaje de reconocimiento. Puesto que los reconocimientos pueden perderse, la estación transmisora podría enviar múltiples copias de una determinada trama. Como vimos en nuestro análisis del protocolo rdt2.1 (Sección 3.4.1), el uso de números de secuencia permite al receptor distinguir entre una trama recién transmitida y la retransmisión de una trama anterior. El campo de número de secuencia en la trama 802.11 sirve aquí, en la capa de enlace, exactamente al mismo propósito que servía a la capa de transporte en el Capítulo 3.

Recuerde que el protocolo 802.11 permite a una estación transmisora reservar el canal durante un periodo de tiempo, que incluye el tiempo para transmitir su trama de datos y el tiempo para transmitir una trama de reconocimiento. Este valor de duración está incluido en el campo Duración de la trama (tanto para las tramas de datos como para las tramas RTS y CTS).

Como se muestra en la Figura 7.13, el campo de control de trama incluye muchos subcampos. Vamos a limitarnos a comentar algunas cosas acerca de los subcampos más importantes; para ver una exposición más completa, puede consultar la especificación 802.11 [Held 2001; Crow 1997; IEEE 802.11 1999]. Los campos *tipo* y *subtipo* se utilizan para distinguir las tramas de asociación, RTS, CTS, ACK y de datos. Los campos *hacia* y *desde* se emplean para definir los significados de los diferentes campos de dirección (estos significados cambian dependiendo de si se está utilizando el modo ad hoc o el modo de infraestructura, y en este último caso dependiendo de si quien está enviando la trama es una estación inalámbrica o un punto de acceso). Finalmente, el campo WEP indica si se está empleando cifrado o no (hablaremos de WEP en el Capítulo 8).

7.3.4 Movilidad dentro de la misma subred IP

Para incrementar el rango físico de una red LAN inalámbrica, las empresas y universidades suelen implantar varios BSS dentro de la misma subred IP. Esto plantea naturalmente el problema de la movilidad entre los distintos BSS: ¿cómo pueden moverse las estaciones inalámbricas de forma transparente de un BSS a otro, mientras mantienen una serie de sesiones TCP activas? Como veremos en esta subsección, la movilidad puede gestionarse de forma relativamente sencilla cuando los BSS forman parte de una misma subred. Cuando las estaciones se desplazan entre subredes contiguas, hacen falta protocolos más complejos de gestión de la movilidad, como los que estudiaremos en las Secciones 7.5 y 7.6.

Examinemos ahora un ejemplo específico de movilidad entre diversos BSS dentro de la misma subred. La Figura 7.15 muestra dos BSS interconectados con un host, H1, que se desplaza desde BSS1 a BSS2. Puesto que en este ejemplo el dispositivo de interconexión que conecta a los dos BSS *no* es un router, todas las estaciones de los dos BSS, incluyendo los puntos de acceso, pertenecen a la misma subred IP. Por tanto, cuando H1 se mueve desde BSS1 a BSS2 puede conservar su dirección IP y todas sus conexiones TCP activas. Si el dispositivo de interconexión fuera un router, entonces H1 tendría que obtener una nueva dirección IP en la subred hacia la cual se está moviendo. Este cambio de dirección interrumpiría (y haría que se terminara) cualquier conexión TCP activa en H1. En la Sección 7.6 veremos cómo puede utilizarse un protocolo de movilidad de la capa red, como por ejemplo IP móvil, para evitar este problema.

¿Pero qué es lo que sucede específicamente cuando H1 se mueve de BSS1 a BSS2? A medida que H1 se aleja de AP1, H1 detecta que la señal de AP1 comienza a debilitarse y empieza entonces a explorar en busca de una señal de mayor intensidad. H1 recibe tramas baliza de AP2 (que en muchos entornos corporativos y universitarios tendrán el mismo identificador SSID que AP1). H1 se desasocia entonces de AP1 y se asocia con AP2, al mismo tiempo que mantiene su dirección IP y sus sesiones TCP activas.

Esto resuelve el problema de la transferencia desde el punto de vista del host y del punto de acceso. ¿Pero qué sucede con el switch de la Figura 7.15? ¿Cómo sabe que el host se ha desplazado de un punto de acceso a otro? Como recordará del Capítulo 6, los switches disponen de una característica de “auto-aprendizaje”, que les permite construir automáticamente sus tablas de reenvío. Esta característica de auto-aprendizaje gestiona de forma eficiente los desplazamientos ocasionales (por ejemplo, cuando se transfiere a un empleado de un departamento a otro); sin embargo, los switches no fueron diseñados para dar soporte a usuarios extremadamente móviles que deseen mantener las conexiones TCP mientras se desplazan entre varios BSS. Para apreciar cuál es aquí el problema, recuerde que antes del desplazamiento el switch tiene una entrada en su tabla de reenvío que empareja la dirección MAC de H1 con la interfaz saliente del comutador a través de la cual se puede alcanzar a H1. Si H1 se encuentra inicialmente en BSS1, entonces un datagrama

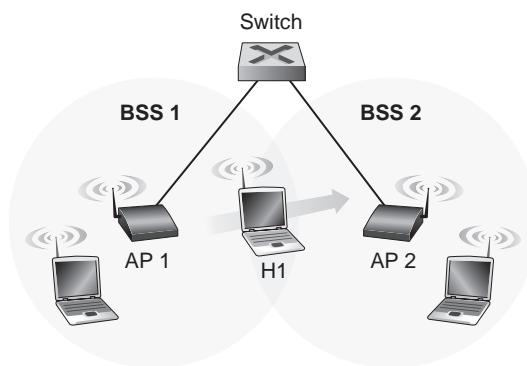


Figura 7.15 ♦ Movilidad dentro de la misma subred.

destinado a H1 tendrá que ser dirigido hacia H1 a través del punto de acceso AP1. Sin embargo, una vez que H1 se asocia con BSS2 sus tramas deben ser dirigidas hacia AP2. Una solución (que en cierto modo es un truco) es que AP2 envíe al switch una trama Ethernet de difusión con la dirección de origen de H1 justo después de la nueva asociación. Cuando el switch reciba la trama actualizará su tabla de reenvío, permitiendo alcanzar a H1 a través de AP2. El grupo de estándares 802.11f está desarrollando un protocolo entre puntos de acceso (inter-AP) para gestionar este problema y otros problemas relacionados.

Hasta aquí nuestras explicaciones se han centrado en la movilidad dentro de la misma subred LAN. Recuerde que las redes VLAN, que hemos estudiado en la Sección 6.4.4, se pueden emplear para conectar islas de redes LAN para formar una red LAN virtual más grande, la cual puede expandirse sobre una región geográfica de mayor tamaño. La movilidad entre estaciones base de una VLAN así puede gestionarse exactamente del mismo modo que acabamos de exponer [Yu 2011].

7.3.5 Características avanzadas de 802.11

Vamos a terminar nuestro estudio sobre 802.11 con un breve análisis de dos capacidades avanzadas que podemos encontrar en las redes 802.11. Como veremos, estas capacidades *no* están completamente especificadas en el estándar 802.11, sino que son una serie de mecanismos especificados en el estándar los que hacen posible dichas capacidades. Esto permite que los diferentes fabricantes implementen dichas capacidades utilizando sus propias técnicas (propietarias), lo que supuestamente les puede proporcionar ventajas sobre sus competidores.

Adaptación de la velocidad

Hemos visto anteriormente, en la Figura 7.3, que las diferentes técnicas de modulación (con las diferentes velocidades de transmisión que proporcionan) pueden resultar apropiadas para diferentes escenarios de SNR. Considere por ejemplo un usuario 802.11 móvil que se encuentra inicialmente a 20 metros de la estación base con una alta relación señal-ruido. Dado el alto valor de SNR, el usuario puede comunicarse con la estación base utilizando una técnica de modulación de la capa física que proporcione altas velocidades de transmisión, al mismo tiempo que se mantiene una baja BER. Este usuario podrá operar en las mejores condiciones. Suponga ahora que el usuario comienza a moverse alejándose de la estación base, con lo que la SNR disminuye a medida que la distancia con la estación base se incrementa. En este caso, si no cambia la técnica de modulación utilizada en el protocolo 802.11 que opera entre la estación base y el usuario, la BER comenzará a ser inaceptablemente alta a medida que la relación SNR se reduzca, llegando eventualmente a un punto en que ninguna de las tramas transmitidas se reciba correctamente.

Por esta razón, algunas implementaciones de 802.11 tienen una capacidad de adaptación de la velocidad que permite seleccionar adaptativamente la técnica subyacente de modulación de la capa física que hay que utilizar, basándose en las características pasadas o recientes del canal. Si un nodo envía dos tramas consecutivas sin recibir una trama de reconocimiento (una indicación implícita de que hay errores de bit en el canal), la velocidad de transmisión se reduce al siguiente nivel inferior. Si se confirman 10 tramas consecutivas o si finaliza el recuento de un temporizador que controla el tiempo transcurrido desde la última reducción, la velocidad de transmisión se incrementa al nivel inmediatamente superior. Este mecanismo de adaptación de la velocidad comparte la misma filosofía de “prueba” que el mecanismo de control de congestión de TCP: cuando las condiciones son buenas (lo que está indicado por la recepción de tramas ACK), la velocidad de transmisión se incrementa, hasta que sucede algo “malo” (la falta de recepción de tramas ACK); cuando sucede algo “malo”, la velocidad de transmisión se reduce. La adaptación de velocidad en 802.11 y en el control de congestión TCP son, por tanto, similares al niño que está constantemente pidiendo más y más a sus padres (por ejemplo, caramelos, en caso de un niño de corta edad, o una hora de llegada a casa más tardía en el caso de los adolescentes) hasta que los padres dicen finalmente que ¡Ya basta!, momento en que el niño echa marcha atrás (solo para intentarlo posteriormente cuando la situación

haya posiblemente mejorado). También se han propuesto varios otros esquemas para mejorar este sistema básico de ajuste automático de la velocidad [Kameran 1997; Holland 2001; Lacage 2004].

Gestión de la potencia

La potencia es un recurso escaso en los dispositivos móviles, por lo que el estándar 802.11 proporciona capacidades de gestión de la potencia que permiten a los nodos 802.11 minimizar la cantidad de tiempo que sus funciones de detección, transmisión y recepción, así como otros circuitos, necesitan estar “activos”. La gestión de potencia en 802.11 opera de la forma siguiente: un nodo es capaz de alternar explícitamente entre los estados dormido y despierto (a diferencia de los estudiantes que se duermen durante las clases). Un nodo indica al punto de acceso que se va a ir a dormir poniendo a 1 el bit de gestión de potencia en la cabecera de una trama 802.11. Entonces se configura un temporizador en el nodo para despertarle justo antes del momento en el que el punto de acceso tiene programado enviar su trama baliza (recuerde que un punto de acceso envía normalmente una trama baliza cada 100 milisegundos). Puesto que el punto de acceso sabe, gracias a que el bit de gestión de potencia está activado, que el nodo se va a dormir, el punto de acceso sabrá que no debe enviar ninguna trama a dicho nodo y almacenará en un buffer todas las tramas destinadas a ese host dormido para su transmisión posterior.

El nodo se despertará justo antes de que el punto de acceso envíe una trama baliza y entrará rápidamente en el estado completamente activo (a diferencia de lo que sucede con los estudiantes adormilados, este despertar requiere únicamente 250 microsegundos [Kameran 1997]). Las tramas baliza enviadas por el punto de acceso contienen una lista de nodos cuyas tramas se han guardado en buffer en el punto de acceso. Si no existen tramas en el buffer para el nodo, este puede volver a dormirse. En caso contrario, el nodo puede solicitar explícitamente que se le envíen las tramas almacenadas en buffer transmitiendo un mensaje de sondeo hacia el punto de acceso. Con un tiempo entre balizas de 100 milisegundos, un tiempo de reactivación de 250 microsegundos y un tiempo similarmente pequeño para recibir una trama baliza y comprobar que no hay ninguna trama en el buffer, un nodo que no tenga tramas que enviar o recibir puede estar durmiendo el 99 por ciento del tiempo, lo que permite un ahorro considerable de energía.

7.3.6 Redes de área personal: Bluetooth y Zigbee

Como se ilustra en la Figura 7.2, el estándar WiFi IEEE 802.11 está pensado para la comunicación entre dispositivos separados hasta 100 metros (excepto cuando se utiliza 802.11 en una configuración punto a punto con una antena direccional). Los otros dos protocolos inalámbricos en el estándar IEEE 802 son Bluetooth y Zigbee (definidos en los estándares IEEE 802.15.1 y IEEE 802.15.4 [IEEE 802.15 2012]).

Bluetooth

Una red IEEE 802.15.1 opera con un corto alcance, a baja potencia y con bajo coste. Se trata básicamente de una tecnología de “sustitución de cables” de baja potencia, corto alcance y baja velocidad que permite la interconexión de una computadora con su ratón, teclado u otro periférico inalámbrico, o la interconexión de teléfonos celulares, altavoces, cascos y otros muchos dispositivos, mientras que 802.11 es una tecnología de “acceso” de mayor potencia, de alcance medio y mayor velocidad. Por esta razón, las redes 802.15.1 se denominan en ocasiones redes inalámbricas de área personal (WPAN, *Wireless Personal Area Network*). Las capas de enlace y física de 802.15.1 están basadas en la especificación anterior de **Bluetooth** para redes de área personal [Held 2001, Bisdikian 2001]. Las redes 802.15.1 operan en la banda de radio sin licencia de 2,4 GHz en forma TDM, con particiones de tiempo de 625 microsegundos. Durante cada partición de tiempo, un emisor transmite en uno de 79 canales, cambiando el canal en una forma conocida pero pseudo-aleatoria de una partición a otra. Este tipo de saltos de canal, que es una técnica conocida con el nombre de **Espec tro disperso por**

salto de frecuencia (FHSS, Frequency-Hopping Spread Spectrum), distribuye las transmisiones a lo largo del tiempo por todo el espectro de frecuencias. 802.15.1 puede proporcionar velocidades de datos de hasta 4 Mbps.

Las redes 802.15.1 son redes ad hoc: no hace falta ninguna infraestructura (por ejemplo, un punto de acceso) para interconectar los dispositivos 802.15.1. Por tanto, estos dispositivos deben organizarse por sí mismos. Los dispositivos 802.15.1 se organizan primero en una **picored** (*piconet*), formada por hasta ocho dispositivos activos, como se muestra en la Figura 7.16. Uno de estos dispositivos se designa como maestro, actuando los dispositivos restantes como esclavos. El nodo maestro gobierna realmente la picored: es su reloj el que determina el tiempo en la picored; el dispositivo maestro puede transmitir en cada partición con número impar y un esclavo solo puede transmitir después de que el maestro se haya comunicado con él en la partición anterior e incluso entonces el esclavo solo puede transmitir hacia el maestro. Además de los dispositivos esclavos, puede haber hasta 255 dispositivos aparcados dentro de la red. Estos dispositivos no pueden comunicarse hasta que su estado sea cambiado por el nodo maestro de aparcado a activo.

Para obtener más información acerca de las redes WPAN, el lector interesado puede consultar las referencias sobre Bluetooth [Held 2001, Bisdikian 2001] o el sitio web oficial de IEEE 802.15 [IEEE 802.15 2012].

Zigbee

Una segunda red de área personal estandarizada por el IEEE es el estándar 802.15.4 [IEEE 802.15 2012], conocido como Zigbee. Mientras que las redes Bluetooth proporcionan una velocidad de datos para “sustitución del cable” de más de un Megabit por segundo, Zigbee está pensada para aplicaciones de menor consumo de potencia, menor velocidad de bit y menor ciclo de trabajo que Bluetooth. Aunque todos tendemos a pensar que “mejor cuanto más grande y más rápido”, no todas las aplicaciones de red necesitan un gran ancho de banda ni el incremento de costes asociado (costes tanto económicos como energéticos). Por ejemplo, los sensores luminosos y de temperatura para entornos domésticos, los dispositivos de seguridad y los interruptores de montaje en pared son muy simples, con bajo consumo de potencia, un pequeño ciclo de trabajo y bajo coste. Zigbee está, por tanto, bien adaptado a estos dispositivos. Zigbee define velocidades de canal de 20, 40, 100 y 250 kbps, dependiendo de la frecuencia del canal.

Hay dos tipos de nodos en una red Zigbee. Los denominados “dispositivos de función reducida” operan como dispositivos esclavos bajo el control de un único “dispositivo de función completa,” de forma bastante similar a los dispositivos esclavos Bluetooth. Un dispositivo de función completa

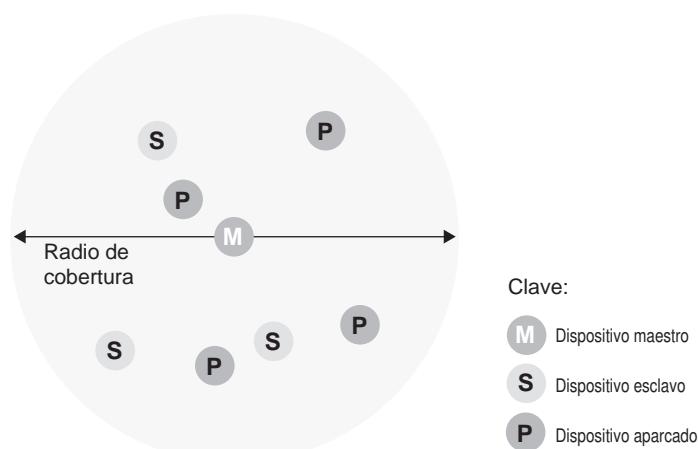


Figura 7.16 ♦ Una picored Bluetooth.

puede operar como dispositivo maestro, como en Bluetooth, controlando múltiples dispositivos esclavos; además, se pueden configurar múltiples dispositivos de función completa en una red de malla, en la que los dispositivos de función completa enrutan tramas entre ellos. Zigbee comparte muchos mecanismos de protocolo que ya nos hemos encontrado en otros protocolos de la capa de enlace: tramas baliza y confirmaciones de la capa de enlace (de forma similar a 802.11), protocolos de acceso aleatorio con detección de portadora y *backoff* exponencial binario (de forma similar a 802.11 y Ethernet) y asignación fija y garantizada de particiones temporales (de forma similar a DOCSIS).

Las redes Zigbee pueden configurarse de muchas formas distintas. Consideremos el caso sencillo de un único dispositivo de función completa que controla múltiples dispositivos de función reducida con particiones de tiempo y usando tramas baliza. La Figura 7.17 muestra el caso en el que la red Zigbee divide el tiempo en supertramas recurrentes, cada una de las cuales comienza con una trama baliza. Cada trama baliza divide la supertrama en un periodo activo (durante el que los dispositivos pueden transmitir) y un periodo inactivo (durante el que todos los dispositivos, incluido el controlador, pueden dormir y ahorrar así potencia). El periodo activo está compuesto por 16 particiones temporales, algunas de las cuales son utilizadas por los dispositivos para un acceso aleatorio tipo CSMA/CA, mientras que otras son asignadas por el controlador a dispositivos específicos, proporcionando así a esos dispositivos un acceso garantizado al canal. Puede encontrar más detalles acerca de las redes Zigbee en [Baronti 2007, IEEE 802.15.4 2012].

7.4 Acceso celular a Internet

En la sección anterior hemos examinado cómo un host puede acceder a Internet cuando entra dentro de un área de cobertura WiFi, es decir, cuando se encuentra en las vecindades de un punto de acceso 802.11. Pero la mayoría de las áreas WiFi tienen una cobertura pequeña, de entre 10 y 100 metros de diámetro. ¿Qué podemos hacer cuando necesitamos desesperadamente acceso inalámbrico a Internet y no podemos acceder a un área WiFi?

Dado que la telefonía celular es ahora omnipresente en muchas áreas de todo el mundo, una estrategia natural consiste en extender las redes celulares de modo que soporten no solo la telefonía de voz sino también el acceso inalámbrico a Internet. Idealmente, este acceso a Internet se llevaría a cabo a una velocidad razonablemente alta y permitiría una movilidad transparente, con lo que los usuarios podrían mantener sus sesiones TCP mientras están viajando, por ejemplo, en un autobús o en un tren. Con tasas de bit lo suficientemente altas tanto para subida como para bajada, el usuario podría incluso mantener sesiones de videoconferencia mientras deambula de un lado a otro (*roaming*). Este escenario no es tan futurista como puede parecer. Están empezando a estar disponibles velocidades de datos de varios megabits por segundo a medida que se van implantando cada vez más servicios de datos de banda ancha como los que aquí vamos a analizar.

En esta sección proporcionamos una breve panorámica de las tecnologías actuales y emergentes de acceso celular a Internet. Vamos a centrarnos en el primer salto inalámbrico, así como en la red que

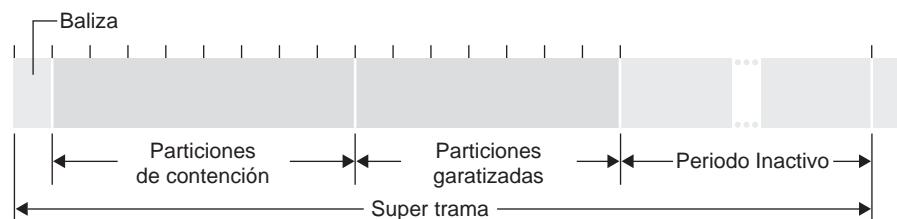


Figura 7.17 ♦ Estructura de la supertrama de Zigbee 802.15.4.

conecta este primer salto inalámbrico con la red telefónica y/o Internet; en la Sección 7.7 veremos cómo se enrutan las llamadas hacia un usuario que se está moviendo entre las distintas estaciones base. Nuestro análisis, debido a su brevedad, solo proporcionará una descripción simplificada y de nivel general de las tecnologías celulares. Por supuesto, las comunicaciones celulares modernas son un tema de gran profundidad y de gran amplitud, existiendo muchas universidades que ofrecen cursos completos sobre el mismo. Los lectores que deseen conocer más detalles pueden consultar [Goodman 1997; Kaaranen 2001; Lin 2001; Korhonen 2003, Schiller 2003; Palat 2009; Scourias 2012; Turner 2012; Akyildiz 2010], así como [Mouly 1992; Sauter 2014], que es una referencia particularmente excelente y exhaustiva.

7.4.1 Panorámica de la arquitectura de las redes celulares

En nuestra descripción de la arquitectura de las redes celulares de esta sección, adoptaremos la terminología de los estándares del *Sistema global de comunicaciones móviles (GSM, Global System for Mobile Communications)*. (Para los aficionados a las cuestiones históricas, el acrónimo GSM derivaba originalmente de *Groupe Spécial Mobile*, aunque luego se adoptó el nombre inglés, conservándose las letras originales del acrónimo.) En la década de 1980 los organismos reguladores europeos se dieron cuenta de la necesidad de un sistema de telefonía celular digital europeo que pudiera sustituir los numerosos sistemas de telefonía celular analógica, que eran incompatibles entre sí. Esta iniciativa condujo a la definición del estándar GSM [Mouly 1992]. En Europa se implantó la tecnología GSM con un gran éxito a principios de la década de 1990, y desde entonces GSM ha crecido hasta convertirse en el sistema dominante dentro de la telefonía celular; en estos momentos, más del 80 por ciento de todos los abonados de telefonía celular del mundo utilizan GSM.

Cuando las personas hablan acerca de la tecnología celular, a menudo la clasifican como perteneciente a una de varias “generaciones”. Las primeras generaciones estaban diseñadas principalmente para el tráfico de voz. Los sistemas de primera generación (1G) eran sistemas FDMA analógicos, diseñados exclusivamente para la comunicación únicamente de voz. Estos sistemas 1G están prácticamente extintos en la actualidad, habiendo sido sustituidos por los sistemas 2G digitales. Los sistemas 2G originales también estaban diseñados para voz, pero posteriormente se ampliaron (2.5G) para soportar servicios tanto de voz como de datos (es decir, Internet). Los sistemas 3G también soportan voz y datos, pero poniendo un énfasis cada vez mayor en las capacidades de datos y en los enlaces de acceso de radio de más alta velocidad. Los sistemas 4G, que se están implantando actualmente, están basados en la tecnología LTE, incorporan una red principal completamente IP y proporcionan servicios de datos y voz integrados a velocidades de varios Megabit.

Arquitectura de redes celulares, 2G: conexiones de voz con la red telefónica

El término *celular* hace referencia al hecho de que la región cubierta por una red celular está dividida en una serie de áreas geográficas de cobertura, denominadas **celdas**, que se muestran como hexágonos en la parte izquierda de la Figura 7.18. Igual que el estándar WiFi 802.11 que hemos estudiado en la Sección 7.3.1, GSM tiene su propia nomenclatura particular. Cada celda contiene una **estación transductora base (BTS, Base Transceiver Station)** que transmite y recibe señales hacia y desde las estaciones móviles que se encuentran dentro de su celda. El área de cobertura de una celda depende de muchos factores, incluyendo la potencia de transmisión de la BTS, la potencia de transmisión de los dispositivos de usuario, los edificios situados dentro de la celda que puedan obstruir las comunicaciones y la altura de las antenas de la estación base. Aunque la Figura 7.18 muestra que cada celda contiene una estación transductora base situada en el centro de la celda, muchos sistemas actuales colocan las BTS en las esquinas donde intersectan tres celdas, de modo que una única BTS con antenas direccionales pueda dar servicio a las tres.

El estándar GSM para los sistemas celulares 2G utiliza una combinación FDM/TDM (radio) para la interfaz aérea. Recuerde del Capítulo 1 que, con FDM pura el canal se partitiona en una serie de canales de frecuencia, estando cada banda dedicada a una llamada. Recuerde también de

HISTORIA

TECNOLOGÍA MÓVIL CELULAR 4G FRENTE A REDES LAN INALÁMBRICAS

Muchos operadores de telefonía móvil celular están implantando sistemas móviles celulares 4G. En algunos países (como Corea y Japón), la cobertura 4G LTE es superior al 90%, es decir, casi ubicua. En 2015, las tasa medias de descarga en los sistemas LTE implantados van desde 10Mbps en Estados Unidos e India a cerca de 40 Mbps en Nueva Zelanda. Estos sistemas 4G se están implantando en bandas de radiofrecuencia con licencia, habiendo pagado algunos operadores sumas considerables a los respectivos gobiernos por dichas licencias. Los sistemas 4G permiten a los usuarios acceder a Internet desde ubicaciones remotas en exteriores mientras están viajando, de forma similar al actual acceso a los servicios de telefonía celular. En muchos casos, un usuario puede tener acceso simultáneo a redes LAN inalámbricas y 4G. Estando la capacidad de los sistemas 4G más restringida, y siendo más cara, muchos dispositivos móviles utilizan WiFi de forma preferente a 4G, cuando ambas están disponibles. La cuestión de si el acceso inalámbrico a la red de frontera se hará principalmente a través de redes LAN inalámbricas o de sistemas celulares sigue sin tener respuesta por el momento:

- La emergente infraestructura de redes LAN inalámbricas será en el futuro prácticamente ubicua. Las redes LAN inalámbricas IEEE 802.11, que operan a 54 Mbps, disfrutan de una implantación cada vez mayor. Casi todas las computadoras portátiles, tabletas y smartphones vienen equipados de fábrica con tarjetas LAN 802.11. Además, los dispositivos Internet emergentes (como las cámaras inalámbricas y los marcos de fotografías electrónicos) también utilizarán pequeñas tarjetas LAN inalámbricas de baja potencia.
- Las estaciones base para redes LAN inalámbricas podrían también comunicarse con dispositivos de telefonía móvil. Muchos teléfonos ya son capaces de conectarse a la red de telefonía celular o a una red IP, bien de forma nativa o utilizando un servicio de voz sobre IP similar a Skype, evitando con ello los servicios de datos 4G y los servicios de voz de telefonía celular de los operadores.

Por supuesto, muchos otros expertos piensan que 4G no sólo será un gran éxito, sino que también va a revolucionar de forma importante la manera en que trabajamos y vivimos. Evidentemente, puede que tanto WiFi como 4G se conviertan en tecnologías inalámbricas prevalentes, y que los dispositivos inalámbricos puedan seleccionar automáticamente, mientras se desplazan, la tecnología de acceso que proporcione el mejor servicio en cada ubicación física concreta.

ese capítulo que, con la multiplexación TDM pura, el tiempo se divide en marcos que a su vez se subdividen en particiones y que a cada llamada se le asigna el uso de una partición concreta dentro del marco. En los sistemas FDM/TDM combinados, el canal se partitiona en una serie de sub-bandas de frecuencia y dentro de cada sub-banda el tiempo se divide en marcos y particiones. Por tanto, para un sistema FDM/TDM combinado, si el canal está partitionado en F sub-bandas y el tiempo se divide en T particiones, entonces el canal podrá soportar $F \cdot T$ llamadas simultáneas. Recuerde que en la Sección 6.3.4 hemos visto que las redes de acceso por cable también emplean la técnica FDM/TDM combinada. Los sistemas GSM están compuestos por bandas de frecuencia de 200-kHz soportando cada una de esas bandas de frecuencia ocho llamadas TDM. GSM codifica la voz a 13 kbps y 12,2 kbps.

El **controlador de la estación base (BSC, Base Station Controller)** de una red GSM normalmente da servicio a varias decenas de estaciones transductoras base. La función del BSC consiste en asignar los canales de radio de las BTS a los abonados móviles, determinar la celda en la que se encuentra un usuario móvil (*paging*) y llevar a cabo la transferencia (*handoff*) de los usuarios móviles, que es un tema del que nos ocuparemos en breve en la Sección 7.7.2. El controlador de las estaciones base y las estaciones transductoras base que controla forman, colectivamente, un **Subsistema de estaciones base (BSS, Base Station Subsystem)** GSM.

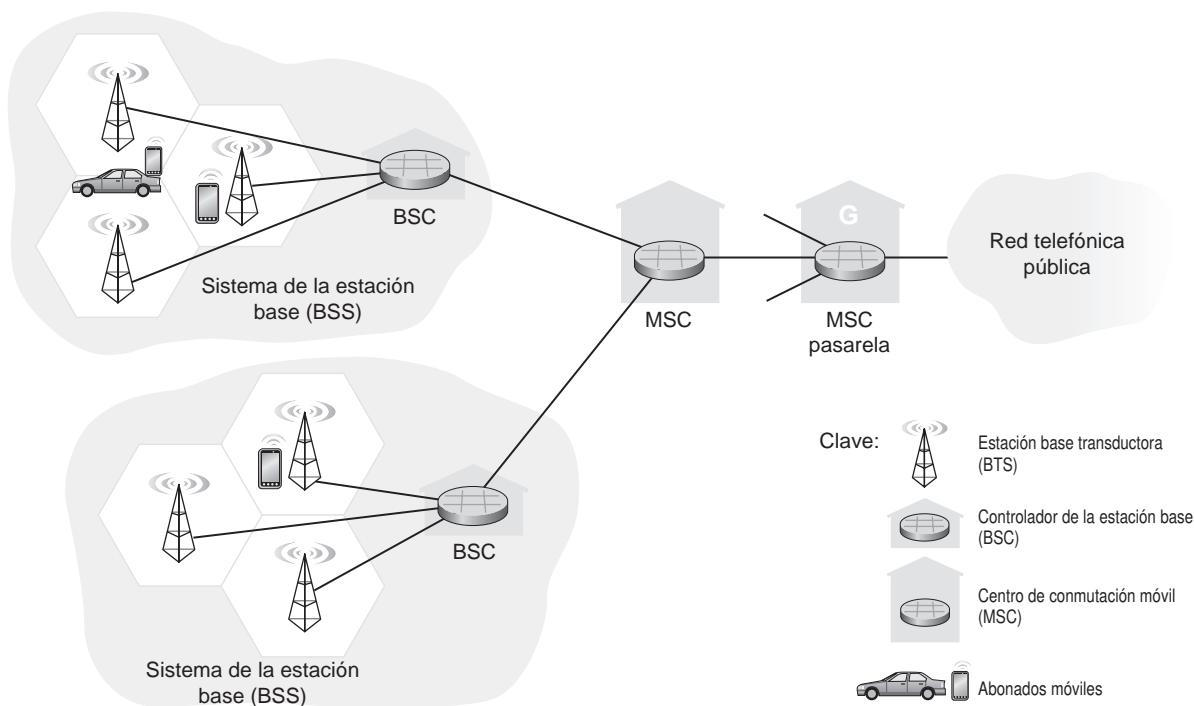


Figura 7.18 ♦ Componentes de una arquitectura de red celular GSM 2G.

Como veremos en la Sección 7.7, el **Centro de conmutación móvil (MSC, Mobile Switching Center)** desempeña el papel central en lo que respecta a la autorización de los usuarios y la facturación (por ejemplo, determinando si se permite a un cierto dispositivo móvil conectarse a la red celular), el establecimiento y finalización de llamadas y la transferencia de las mismas. Un único MSC contendrá normalmente hasta cinco BSC, lo que permite tener aproximadamente unos 200.000 abonados por MSC. La red de un proveedor de servicio celular tendrá una serie de centros MSC, siendo algunos de ellos MSC especiales conocidos con el nombre de centros MSC pasarela, los cuales sirven para conectar la red celular de ese proveedor con la red telefónica pública de mayor tamaño.

7.4.2 Redes de datos celulares 3G: llevando Internet a los abonados celulares

Nuestra exposición de la Sección 7.4.1 se ha centrado en cómo conectar a los usuarios de servicios celulares de voz con la red pública de telefonía. Pero, por supuesto, mientras nos estamos desplazando también queremos leer nuestro correo electrónico, acceder a la Web, obtener servicios dependientes de la ubicación (por ejemplo, mapas y recomendaciones de restaurantes) y quizás incluso ver una película de vídeo a través de la red. Para ello, nuestro teléfono inteligente necesitará ejecutar una pila de protocolos TCP/IP completa (que incluya las capas física, de enlace, de red, de transporte y de aplicación) y conectarse a Internet a través de la red de datos celular. El campo de las redes de datos celulares es una bastante confusa colección de estándares competitores y en perpetua evolución, a medida que cada nueva generación (y media generación) sucede a la anterior e introduce nuevas tecnologías y servicios, con nuevos acrónimos. Para empeorar las cosas, no existe ningún organismo oficial unificado que establezca los requisitos para las tecnologías 2.5G, 3G, 3.5G o 4G, haciendo difícil entender las diferencias entre los distintos estándares competidores. En esta exposición, vamos a centrarnos en los estándares 3G y 4G de UMTS (*Universal Mobile Telecommunications Service*,

Servicio universal de telecomunicaciones móviles), desarrollados por el consorcio *3rd Generation Partnership Project* (3GPP) [3GPP 2016].

Echemos primero un vistazo de arriba a abajo a la arquitectura de una red de datos celular 3G mostrada en la Figura 7.19.

Núcleo de la red 3G

El núcleo de la red de datos celular 3G conecta las redes de acceso radio a la Internet pública. El núcleo de la red interopera con componentes de la red celular de voz existente (en particular, el MSC) con los que ya nos hemos encontrado en la Figura 7.18. Dada la considerable cantidad de infraestructura (¡y de rentables servicios!) existente en la actual red celular de voz, la solución adoptada por los diseñadores de los servicios de datos 3G es muy clara: *dejar sin tocar la red celular de voz principal GSM existente, añadiendo la funcionalidad adicional de datos celulares en paralelo con la red celular de voz existente*. La alternativa —integrar los nuevos servicios de datos directamente en el núcleo de la red celular de voz existente— habría planteado los mismos desafíos con los que ya nos hemos encontrado en la Sección 4.3 al hablar de la integración de tecnologías nuevas (IPv6) y heredadas (IPv4) en Internet.

Hay dos tipos de nodos en el núcleo de la red 3G: **nodos de soporte GPRS servidor (SGSN, Serving GPRS Support Node)** y **nodos de soporte GPRS pasarela (GGSN, Gateway GPRS**

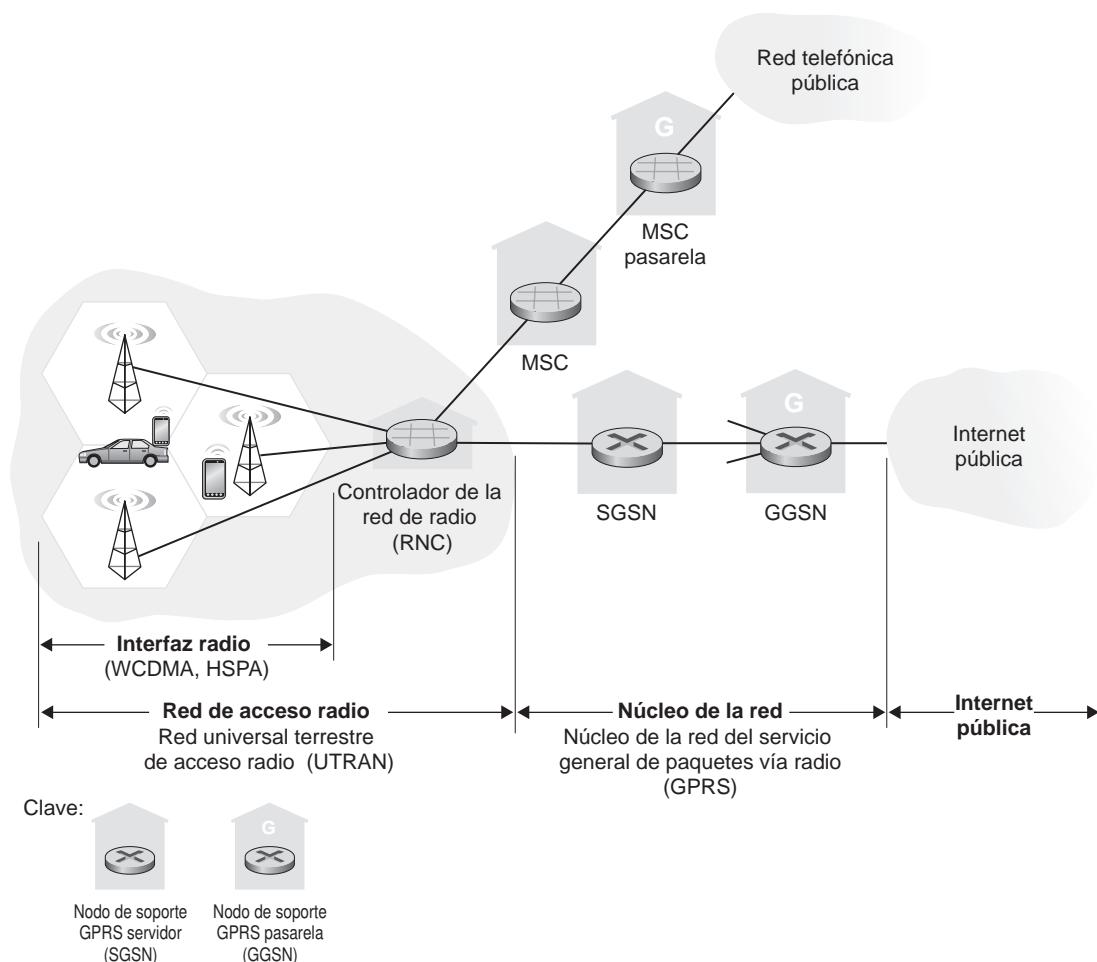


Figura 7.19 ♦ Arquitectura del sistema 3G.

Support Node). (GPRS significa *General Packet Radio Service*, Servicio general de paquetes vía radio, uno de los primeros servicios de datos celulares en las redes 2G; aquí vamos a hablar de la versión evolucionada de GPRS en las redes 3G.) Un SGSN es responsable de entregar los datagramas que viajan hacia/desde los nodos móviles de la red de acceso vía radio a la que el SGSN está conectado. El SGSN interactúa con el MSC de la red celular de voz correspondiente a dicha área, encargándose de la autorización del usuario y la cesión de llamada, de mantener la información de ubicación (celda) relativa a los nodos móviles activos y de realizar el reenvío de datagramas entre los nodos móviles de la red de acceso vía radio y un GGSN. El GGSN actúa como pasarela, conectando múltiples SGSN con Internet. Un GGSN es, por tanto, el último elemento de la infraestructura 3G con el que se encuentra un datagrama generado por un nodo móvil antes de entrar en Internet. A ojos del mundo exterior, el GGSN se parece a cualquier otro router de pasarela; el GGSN oculta al mundo exterior la movilidad de los nodos 3G existentes dentro de su red.

Red de acceso radio 3G: la frontera inalámbrica

La red de acceso radio 3G es la red inalámbrica de primer salto que vemos como usuarios de 3G. El **Controlador de red radio (RNC, Radio Network Controller)** suele controlar varias estaciones base celulares transceptoras, similares a las estaciones base que ya nos hemos encontrado en los sistemas 2G (pero a las que se conoce oficialmente en la jerga 3G UMTS con el nombre de “Nodos Bs”, un nombre bastante poco descriptivo). El enlace inalámbrico de cada celda opera entre los nodos móviles y una estación base transceptora, igual que en las redes 2G. El RNC se conecta tanto a la red celular de voz de conmutación de circuitos (a través de un MSC), como a la red Internet de conmutación de paquetes (a través de un SGSN). Por tanto, aunque los servicios celulares de voz y datos 3G utilizan diferentes redes principales, comparten una misma red de acceso radio de primer/último salto.

Un cambio significativo en 3G UMTS con respecto a las redes 2G es que, en lugar de utilizar el esquema FDMA/TDMA de GSM, UMTS emplea una técnica CDMA conocida con el nombre de CDMA de banda ancha de secuencia directa (DS-WCDMA, *Direct Sequence Wideband CDMA*) [Dahlman 1998] dentro de particiones TDMA; las particiones TDMA, a su vez, están disponibles en múltiples frecuencias, lo cual constituye una aplicación interesante de las tres soluciones dedicadas de compartición del canal que hemos identificado anteriormente en el Capítulo 6 y es similar a la solución adoptada en las redes terrestres de acceso por cable (véase la Sección 6.3.4). Este cambio requiere una nueva red celular de acceso inalámbrico que opere en paralelo con la red de radio BSS 2G mostrada en la Figura 7.19. El servicio de datos asociado con la especificación WCDMA se conoce con el nombre de HSPA (*High Speed Packet Access*, Acceso de paquetes de alta velocidad) y promete velocidades de datos de bajada de hasta 14 Mbps. Puede encontrar más detalles relativos a las redes 3G en el sitio web del consorcio 3GPP (*3rd Generation Partnership Project*) [3GPP 2016].

7.4.3 Hacia la tecnología 4G: LTE

Los sistemas celulares de cuarta generación (4G) se están implantando de forma generalizada. En 2015, más de 50 países tenían una cobertura 4G superior al 50%. El estándar 4G LTE (*Long-Term Evolution*, Evolución a largo plazo) [Sauter 2014] impulsado por el 3GPP presenta dos innovaciones importantes con respecto a los sistemas 3G: un núcleo de la red completamente IP y una red mejorada de acceso vía radio, como se explica a continuación.

Arquitectura del sistema 4G: un núcleo de la red completamente IP

La Figura 7.20 muestra la arquitectura global de una red 4G, que (desafortunadamente) introduce otro nuevo vocabulario más (bastante impenetrable), junto con su correspondiente conjunto de acrónimos, para los componentes de la red. ¡Pero no nos perdamos en los acrónimos! Hay dos observaciones importantes de alto nivel que podemos hacer sobre la arquitectura 4G:

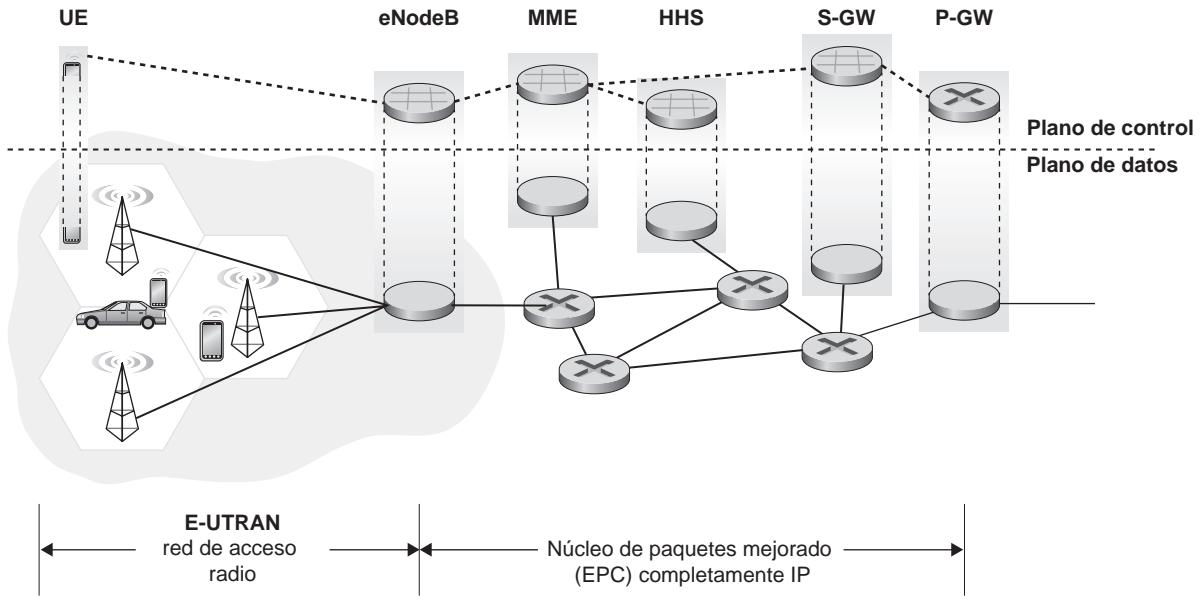


Figura 7.20 ♦ Arquitectura de red 4G.

- *Una arquitectura de red unificada, completamente IP.* A diferencia de la red 3G mostrada en la Figura 7.19, que tiene rutas y componentes de red separados para el tráfico de voz y de datos, la arquitectura 4G mostrada en la Figura 7.20 es “completamente IP”: tanto la voz como los datos son transportados en datagramas IP hacia/desde el dispositivo inalámbrico (el equipo de usuario, UE, *User Equipment*, en la jerga 4G) hasta la pasarela de paquetes (P-GW, *Packet Gateway*) que conecta la red de frontera 4G con el resto de la red. ¡Con 4G, los últimos vestigios de las raíces telefónicas de las redes celulares han desaparecido, dejando paso al servicio IP universal!
- *Una clara separación del plano de datos 4G y el plano de control 4G.* De forma similar a la distinción que hacíamos entre los planos de datos y de control de la capa de red IP en los Capítulos 4 y 5, respectivamente, la arquitectura de una red 4G también separa claramente los planos de datos y de control. Más adelante hablaremos de su funcionalidad.
- *Una clara separación entre la red de acceso vía radio y el núcleo de la red completamente IP.* Los datagramas IP que transportan los datos del usuario son reenviados entre el usuario (UE) y la pasarela (P-GW en la Figura 7.20) a través de una red IP interna a 4G, hasta la red Internet externa. A través de esta misma red interna, se intercambian paquetes de control entre los componentes de los servicios de control 4G cuyos roles se describen a continuación.

Los componentes principales de la arquitectura 4G son los siguientes:

- **eNodeB** es el descendiente lógico de la estación base 2G y del Controlador de red de radio 3G (RNC, también llamado Node B) y también aquí juega un papel crucial. Su misión en el plano de datos consiste en reenviar los datagramas entre el UE (a través de la red de acceso radio LTE) y el P-GW.

Los datagramas del UE se encapsulan en el eNodeB y se tunelizan hacia el P-GW a través del núcleo de paquetes mejorado (EPC, *Enhanced Packet Core*) completamente IP de la red 4G. Esta tunelización entre el eNodeB y el P-GW es similar a la tunelización (que ya vimos en la Sección 4.3) de datagramas IPv6 entre dos puntos terminales IPv6 a través de una red de routers IPv4. Estos túneles pueden tener asociadas ciertas garantías de calidad de servicio (QoS). Por ejemplo, una red 4G puede garantizar que el tráfico de voz experimente un retardo no superior a 100 mseg entre el UE y el P-GW, y que tenga una tasa de pérdida de paquetes inferior al 1%; el tráfico TCP,

por su parte, podría tener una garantía de 300 ms y una tasa de pérdida de paquetes inferior al 0,0001% [Palat 2009]. Hablaremos de la calidad de servicio en el Capítulo 9.

En el plano de control, el eNodeB gestiona el registro y el tráfico de señalización de movilidad por cuenta del UE.

- **LaPasarela de la red de datos empaquetados (P-GW, *Packet Data Network Gateway*)** asigna direcciones IP a los equipos UE y se encarga de imponer las garantías QoS. Como punto terminal de un túnel, también lleva a cabo la encapsulación/desencapsulación de los datagramas al reenviarlos hacia/desde un UE.
- **LaPasarela de servicio (S-GW, *Service Gateway*)** es el punto de anclaje de movilidad del plano de datos: todo el tráfico del UE pasará a través del S-GW. El S-GW también se encarga de las funciones de tarificación/facturación y de las intercepciones legalmente autorizadas del tráfico.
- **LaEntidad de gestión de la movilidad (MME, *Mobility Management Entity*)** se encarga de la gestión de conexión y de movilidad por cuenta de los UE residentes en la celda que controla. Recibe información de abonado del UE desde el HSS. Hablaremos en detalle de la movilidad en redes celulares en la Sección 7.7.
- **ElServidor de abonado doméstico (HSS, *Home Subscriber Server*)** contiene información del UE, incluyendo las capacidades de acceso itinerante, los perfiles de calidad de servicio e información de autenticación. Como veremos en la Sección 7.7, el HSS obtiene esta información del operador celular doméstico del UE.

Algunas introducciones bastante comprensibles a la arquitectura de una red 4G y su EPC son [Motorola 2007; Palat 2009; Sauter 2014].

Red de acceso vía radio LTE

LTE usa en el canal de bajada una combinación de multiplexación por división de frecuencia y multiplexación por división del tiempo, conocida con el nombre de multiplexación por división de frecuencia ortogonal (OFDM, *Orthogonal Frequency Division Multiplexing*) [Rohde 2008; Ericsson 2011]. (El término “ortogonal” proviene del hecho de que las señales que se envían a través de distintos canales de frecuencia se crean de tal modo que interfieren muy poco unas con otras, incluso aunque las frecuencias de los canales estén muy próximas entre sí.) En LTE, a cada nodo móvil activo se le asignan una o más particiones de tiempo de 0,5 milisegundos en una o más de las frecuencias de canal. La Figura 7.21 muestra una asignación de ocho particiones temporales en cuatro frecuencias distintas. A medida que se le asignan más particiones temporales (en la misma frecuencia o en frecuencias diferentes), un nodo móvil puede conseguir velocidades de transmisión más altas. La (re)asignación de particiones entre nodos móviles puede realizarse cada milisecondo. También pueden usarse diferentes esquemas de modulación para cambiar la velocidad de transmisión; véanse las explicaciones anteriores de la Figura 7.3 y la selección dinámica de esquemas de modulación en las redes WiFi.

La asignación concreta de particiones temporales a los nodos móviles no está regulada por el estándar LTE. En lugar de ello, la decisión de a qué nodos móviles se les permitirá transmitir en una cierta partición temporal de una cierta frecuencia está determinada por los algoritmos de planificación proporcionados por el fabricante de los equipos LTE y/o por el operador de la red. Con una planificación oportunista [Bender 2000; Kolding 2003; Kulkarni 2005], que ajuste el protocolo de la capa física a las condiciones del canal existente entre el emisor y el receptor y seleccione los receptores a los que se enviarán los paquetes basándose en las condiciones del canal, el controlador de la red de radio puede hacer un uso óptimo del medio inalámbrico. Además, pueden emplearse prioridades de usuario y niveles contractuales de servicio (por ejemplo, plata, oro o platino) a la hora de planificar las transmisiones de paquetes en el sentido de bajada. Además de las capacidades de LTE descritas anteriormente, LTE-Advanced permite anchos de banda de bajada de centenares de Mbps por el procedimiento de asignar canales agregados a un nodo móvil [Akyildiz 2010].

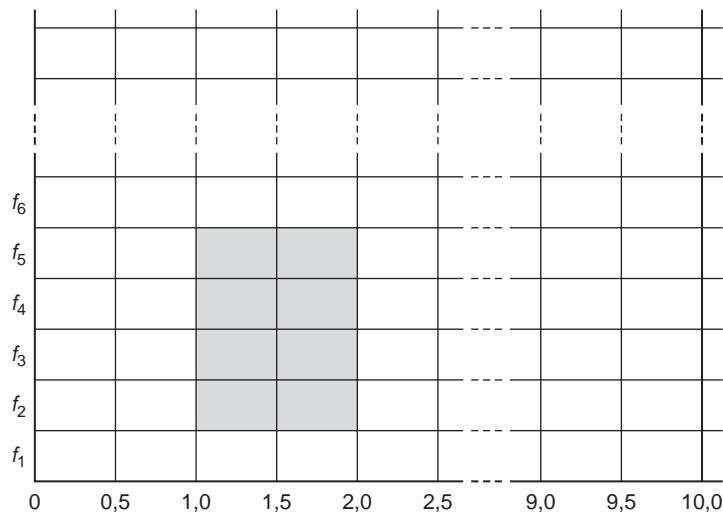


Figura 7.21 ♦ Veinte particiones de 0,5 ms organizadas en tramas de 10 ms en cada frecuencia. Se indica sombreada una asignación de ocho particiones.

Otra tecnología inalámbrica 4G —WiMAX (*World Interoperability for Microwave Access*, Interoperabilidad mundial para acceso por microondas)— es una familia de estándares IEEE 802.16 que difiere significativamente de LTE. WiMAX no goza todavía de la amplia implantación de LTE. En el sitio web del libro puede encontrar una explicación detallada de WiMAX.

7.5 Gestión de la movilidad: principios

Habiendo cubierto la naturaleza *inalámbrica* de los enlaces de comunicaciones existentes en una red inalámbrica, ahora es el momento de volver nuestra atención hacia la *movilidad* que estos enlaces inalámbricos permiten. En el sentido más amplio, un nodo móvil es aquel que cambia su punto de conexión con la red a lo largo del tiempo. Puesto que el término *movilidad* ha adoptado muchos significados tanto en el mundo de las computadoras como en el de la telefonía, conviene primero considerar con cierto detalle diversas dimensiones de la movilidad.

- *Desde el punto de vista de la capa de red, ¿cómo de móvil es un usuario?* Un usuario físicamente móvil planteará un conjunto muy distinto de desafíos a la capa de red dependiendo de cómo se mueva entre los puntos de conexión con la red. En un extremo del espectro de la Figura 7.22, un usuario puede llevar consigo una computadora portátil con una tarjeta de interfaz de red inalám-

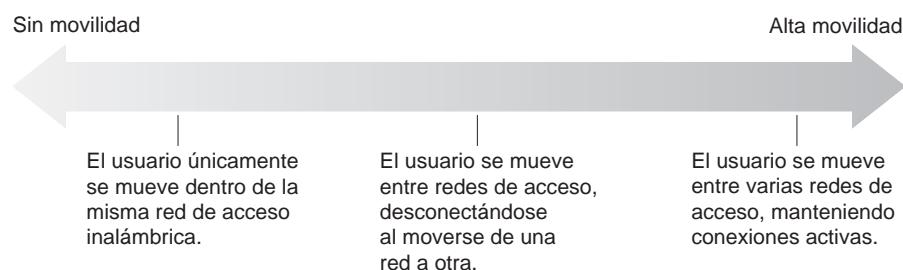


Figura 7.22 ♦ Diversos grados de movilidad desde el punto de vista de la capa de red.

brica mientras pasea por un edificio. Como vimos en la Sección 7.3.4, este usuario *no* es móvil desde la perspectiva de la capa de red. Además, si el usuario se asocia con el mismo punto de acceso, independientemente de su ubicación, ese usuario no es ni siquiera móvil desde la perspectiva de la capa de enlace.

En el otro extremo del espectro, considere el usuario que está viajando a 150 kilómetros por hora por una autovía en un BMW o un Tesla, pasando por múltiples redes de acceso inalámbricas y que quiere mantener una conexión TCP ininterrumpida con alguna aplicación remota a todo lo largo del viaje. ¡Este usuario sí que es móvil sin ningún tipo de dudas! Entre estos dos extremos se encontraría el usuario que traslada una computadora portátil desde una ubicación (por ejemplo, su oficina o su domicilio) a otra (por ejemplo, una cafetería o un aula) y quiere conectarse a la red en esa nueva ubicación. Este usuario también es móvil (aunque menos que el conductor del BMW), pero no necesita mantener una conexión activa mientras se está moviendo entre los puntos de conexión con la red. La Figura 7.22 ilustra este espectro de movilidad del usuario desde la perspectiva de la capa de red.

- *¿Hasta qué punto es importante que la dirección del nodo móvil sea siempre la misma?* Con la telefonía móvil, nuestro número de teléfono (que es básicamente la dirección de la capa de red de nuestro teléfono) continúa siendo el mismo a medida que nos desplazamos desde la red de telefonía móvil de un proveedor a la de otro. ¿Deben las computadoras portátiles mantener de forma similar la misma dirección IP mientras se están desplazando entre redes IP?

La respuesta a esta cuestión dependerá en gran medida de las aplicaciones que se estén ejecutando. Para el conductor del BMW o Tesla que quiere mantener una conexión TCP ininterrumpida con una aplicación remota mientras viaja por la autovía sería conveniente mantener la misma dirección IP. Recuerde del Capítulo 3 que una aplicación Internet necesita conocer la dirección IP y el número de puerto de la entidad remota con la que se está comunicando. Si una entidad móvil es capaz de mantener su dirección IP a medida que se desplaza, la movilidad se convertirá en algo transparente desde el punto de vista de la aplicación. Esta transparencia tiene un enorme valor, ya que la aplicación no tendrá que preocuparse de la cuestión de que las direcciones IP puedan potencialmente cambiar, y un mismo código de aplicación permitirá dar servicio tanto a conexiones móviles como no móviles. Veremos en la siguiente sección que la tecnología de IP móvil proporciona esta transparencia, permitiendo a un nodo móvil mantener su dirección IP permanente mientras se está desplazando de una red a otra.

Por otro lado, un usuario móvil menos glamuroso podría simplemente querer apagar su computadora portátil en la oficina, llevarla a su domicilio, volver a encenderla y trabajar desde su casa. Si la computadora portátil funciona principalmente como un cliente en aplicaciones cliente-servidor (por ejemplo, enviar/leer correo electrónico, navegar por la Web, conectarse mediante Telnet a un host remoto) desde su domicilio, la dirección IP concreta utilizada por la computadora portátil no tiene tanta importancia. En particular, el usuario podría perfectamente funcionar con una dirección que el ISP que da servicio a su domicilio le asignara temporalmente a la computadora portátil. Ya hemos visto en la Sección 4.3 que DHCP ya proporciona esta funcionalidad.

- *¿Qué infraestructura cableada de soporte hay disponible?* En todos los escenarios anteriores hemos supuesto implícitamente que existe una infraestructura fija a la que el usuario móvil puede conectarse: por ejemplo, la red del ISP que da servicio a su domicilio, la red de acceso inalámbrico de la oficina o las redes de acceso inalámbrico que atraviesan la autovía. ¿Qué sucede si no existe tal infraestructura? Si dos usuarios están próximos entre sí, desde el punto de vista de las comunicaciones, ¿pueden establecer una conexión de red en ausencia de cualquier otra infraestructura de la capa de red? Las redes ad hoc proporcionan precisamente estas capacidades. Este área en rápido desarrollo constituye la vanguardia de las investigaciones en redes móviles y cae fuera del alcance de este libro. En [Perkins 2000] y en las páginas web del Grupo de trabajo de redes móviles ad hoc (manet) del IETF [manet 2016] se proporciona un tratamiento bastante exhaustivo de esta materia.

Para ilustrar los problemas implicados en el hecho de permitir a un usuario móvil a mantener conexiones activas mientras se está desplazando entre varias redes, vamos a considerar una analogía humana. Un veinteañero que se va del domicilio familiar pasa a ser un adulto móvil, viviendo en una serie de habitaciones y/o apartamentos, cambiando a menudo de dirección. Si un viejo conocido quiere ponerse en contacto con él, ¿cómo puede ese conocido encontrar la dirección de su amigo móvil? Una forma común sería contactar con la familia, ya que un adulto móvil a menudo registrará ante la familia (le comunicará) su dirección actual (aunque solo sea para que sus padres puedan enviarle dinero para pagarle el alquiler). El domicilio familiar, con su dirección permanente, se convertirá en ese lugar al que otros pueden acudir en primer lugar para poder comunicarse con ese adulto móvil. Las comunicaciones posteriores de ese viejo conocido pueden ser indirectas (por ejemplo, ese conocido puede enviar correo al domicilio de los padres y estos reenviar el correo a nuestro adulto móvil) o directas (por ejemplo, cuando ese viejo conocido utiliza la dirección que le han facilitado los padres para enviar el correo directamente a su amigo móvil).

En un entorno de red, el domicilio permanente de un nodo móvil (como por ejemplo una computadora portátil o un smartphone) se conoce con el nombre de **red propia** (*home network*) y la entidad dentro de la red propia que se encarga de llevar a cabo las funciones de gestión de la movilidad, de las que hablaremos más adelante, por cuenta del nodo móvil se conoce como **agente propio** (*home agent*). La red en la que reside actualmente el nodo móvil se conoce con el nombre de **red ajena o visitada** (*foreign or visited network*) y la entidad dentro de la red ajena que ayuda al nodo móvil con las funciones de gestión de la movilidad, que veremos más adelante, se conoce con el nombre de **agente ajeno** (*foreign agent*). Para los profesionales móviles, lo más probable es que su red propia sea la red de su empresa, mientras que la red ajena podría ser la de un colega al que estén visitando. Un **corresponsal** (*correspondent*) es la entidad que se quiere comunicar con el nodo móvil. La Figura 7.23 ilustra estos conceptos, así como los conceptos de direccionamiento considerados más adelante. En la Figura 7.23, observe que los agentes se muestran como si estuvieran ubicados

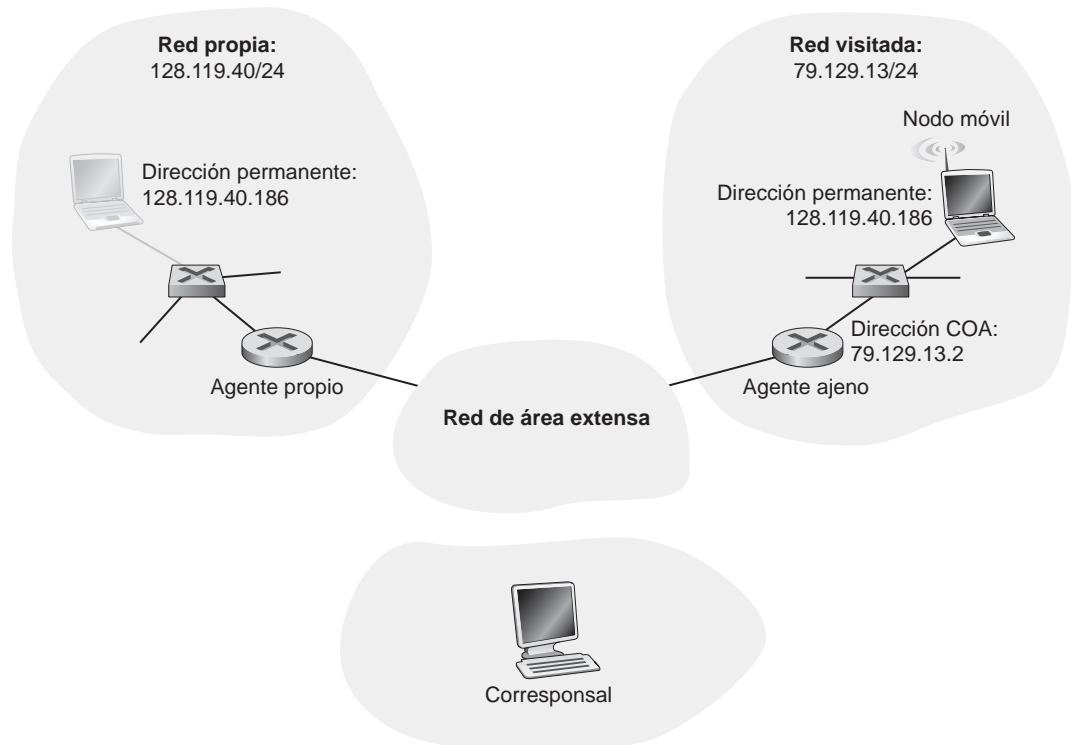


Figura 7.23 ♦ Elementos iniciales de una arquitectura de red móvil.

en routers (por ejemplo, como procesos ejecutándose en routers), pero también podrían ejecutarse en otros hosts o servidores de la red.

7.5.1 Direccionamiento

Hemos observado anteriormente que, para que la movilidad de los usuarios sea transparente a ojos de las aplicaciones de red, es deseable que los nodos móviles conserven su dirección mientras se mueven de una red a otra. Cuando un nodo móvil reside en una red ajena, todo el tráfico dirigido a la dirección permanente de dicho nodo ahora tendrá que ser enrutado hacia la red ajena. ¿Cómo puede hacerse esto? Una opción es que la red ajena anuncie a todas las demás redes que el nodo móvil está residiendo ahora en su red. Esto podría hacerse mediante el intercambio usual de información de enrutamiento entre dominios y dentro de los dominios y requeriría pocos cambios en la infraestructura de enrutamiento existente. La red ajena podría simplemente anunciar a sus vecinos que dispone de una ruta altamente específica hacia la dirección permanente del nodo móvil (es decir, básicamente consistiría en informar a otras redes de que dispone de la ruta correcta para enviar datagramas a la dirección permanente del nodo móvil; véase la Sección 4.3). Esos vecinos propagarían entonces esa información de enrutamiento por toda la red como parte del procedimiento normal de actualización de la información de enrutamiento y de las tablas de reenvío. Cuando el nodo móvil abandone una red ajena y se una a otra, la nueva red ajena anunciará una nueva ruta altamente específica hacia el nodo móvil y la antigua red ajena retirará su información de enrutamiento relativa a ese nodo móvil.

Esto resuelve dos problemas a la vez, y lo hace sin llevar a cabo cambios significativos en la infraestructura de la capa de red. Otras redes conocerán la ubicación del nodo móvil y será fácil enrutar los datagramas hacia él, ya que las tablas de reenvío dirigirán los datagramas a la red ajena. Sin embargo, una desventaja significativa es la de la escalabilidad. Si la gestión de la movilidad tuviera que ser responsabilidad de los routers de la red, los routers tendrían que mantener entradas en sus tablas de reenvío para un número potencialmente muy alto de millones de nodos móviles y actualizar dichas entradas a medida que los nodos se movieran. En los problemas incluidos al final del capítulo se exploran otras desventajas adicionales.

Un enfoque alternativo (que además se ha adoptado en la práctica) consiste en trasladar la funcionalidad de movilidad desde el núcleo de la red hasta la frontera de la misma, lo cual es un tema recurrente en nuestro estudio de la arquitectura de Internet. Una forma natural de hacer esto es mediante la red propia del nodo móvil. De la misma forma que los padres del veinteañero móvil se encargan de controlar la ubicación de su hijo, el agente propio situado en la red propia del nodo móvil puede controlar en qué red ajena reside el nodo móvil. Obviamente, será necesario que exista un protocolo entre el nodo móvil (o un agente ajeno que represente al nodo móvil) y el agente propio para poder actualizar la ubicación del nodo móvil.

Consideremos ahora el agente ajeno con un poco más de detalle. El enfoque conceptualmente más simple, mostrado en la Figura 7.23, consiste en ubicar los agentes ajenos en los routers de frontera de la red ajena. Un papel del agente ajeno consiste en crear la denominada **dirección cedida (COA, Care-Of Address)** para el nodo móvil, en la que la parte de red de la COA se corresponde con la de la red ajena. Por tanto, habrá dos direcciones asociadas con un nodo móvil, su **dirección permanente** (análoga a la dirección del domicilio familiar de nuestro veinteañero móvil) y su COA, a veces denominada **dirección ajena** (análoga a la dirección de la vivienda en la que está residiendo actualmente nuestro veinteañero móvil). En el ejemplo de la Figura 7.23, la dirección permanente del nodo móvil es 128.119.40.186. Cuando está visitando la red 79.129.13/24, el nodo móvil tiene una COA igual a 79.129.13.2. Un segundo papel del agente ajeno consiste en informar al agente propio de que el nodo móvil reside en su red (en la del agente ajeno) y tiene la COA indicada. Veremos enseguida que la COA se utiliza para “re-enrutar” datagramas hacia el nodo móvil a través de su agente ajeno.

Aunque hemos separado la funcionalidad del nodo móvil y del agente ajeno, merece la pena indicar que el nodo móvil también puede asumir las responsabilidades del agente ajeno. Por ejemplo,

el nodo móvil podría obtener una COA en la red ajena (por ejemplo, utilizando un protocolo como DHCP) e informar él mismo al agente propio de cuál es su COA.

7.5.2 Enrutamiento hacia un nodo móvil

Hemos visto cómo un nodo móvil puede obtener una COA y cómo puede informar a su agente propio de dicha dirección. Pero hacer que el agente propio conozca la COA solo resuelve parte del problema. ¿Cómo deberían direccionarse los datagramas y cómo deberían reenviarse hacia el nodo móvil? Puesto que solo el agente propio (y no los routers del resto de la red) conoce la ubicación del nodo móvil, ya no bastará con dirigir simplemente un datagrama a la dirección permanente del nodo móvil y enviarlo hacia la infraestructura de la capa de red, sino que hay que hacer algo más. Podemos identificar dos enfoques distintos, a los que denominaremos enrutamiento indirecto y enrutamiento directo.

Enrutamiento indirecto hacia un nodo móvil

Consideremos en primer lugar un corresponsal que desea enviar un datagrama a un nodo móvil. Con la técnica del **enrutamiento indirecto**, el corresponsal simplemente dirige el datagrama con la dirección permanente del nodo móvil y lo envía a la red, completamente ignorante de si el nodo móvil reside en su red propia o está visitando una red ajena; por tanto, la movilidad es completamente transparente para el corresponsal. Dichos datagramas se enrutan primero de la forma habitual hacia la red propia del nodo móvil. Esto se ilustra en el paso 1 de la Figura 7.24.

Volvamos ahora nuestra atención hacia el agente propio. Además de ser responsable de interactuar con un agente ajeno para saber en todo momento la COA del nodo móvil, el agente propio tiene otra función muy importante. Su segunda tarea consiste en estar atento para ver si llegan datagramas

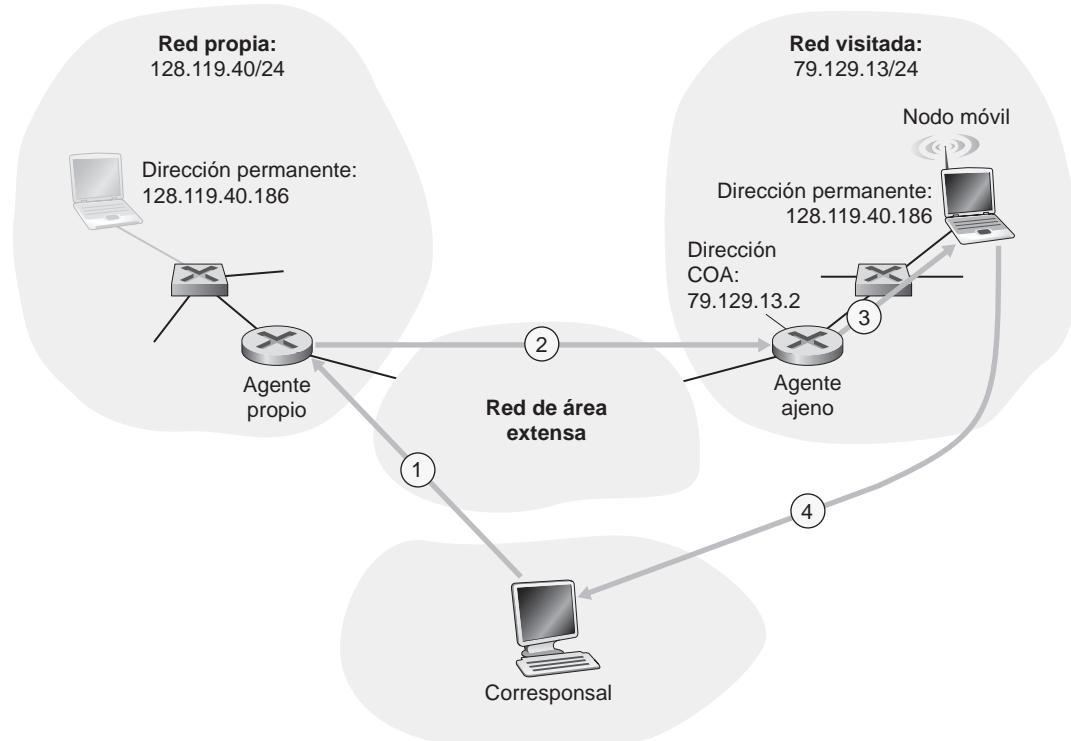


Figura 7.24 ♦ Enrutamiento indirecto hacia un nodo móvil.

dirigidos a nodos cuya red propia sea la de ese agente propio, pero que estén actualmente residiendo en una red ajena. El agente propio intercepta estos datagramas y luego los reenvía hacia un nodo móvil siguiendo un proceso de dos pasos. Primero, el datagrama es reenviado hacia el agente ajeno, utilizando la dirección COA del nodo móvil (paso 2 de la Figura 7.24) y luego es reenviado desde el agente ajeno hacia el nodo móvil (paso 3 de la Figura 7.24).

Es bastante instructivo analizar este re-enrutamiento con mayor detalle. El agente propio necesitará direccionar el datagrama utilizando la COA del nodo móvil, de modo que la capa de red enrute el datagrama hacia la red ajena. Por otro lado, es deseable dejar el datagrama del corresponsal intacto, ya que la aplicación que va a recibir el datagrama no debe ser consciente de que el datagrama ha sido reenviado a través del agente propio. Ambos objetivos pueden satisfacerse haciendo que el agente propio **encapsule** el datagrama completo original del corresponsal dentro de un nuevo datagrama más grande. Este datagrama de mayor tamaño se direcciona y se entrega a la COA del nodo móvil. El agente ajeno, que “posee” la COA, recibirá y desencapsulará el datagrama; es decir, extraerá el datagrama original del corresponsal de dentro del datagrama más grande que lo encapsula y reenviará (paso 3 de la Figura 7.24) el datagrama original hacia el nodo móvil. La Figura 7.25 muestra un datagrama original de un corresponsal que es enviado hacia la red propia, un datagrama encapsulado enviado hacia el agente ajeno y el datagrama original entregado al nodo móvil. El lector atento observará que la encapsulación/desencapsulación aquí descrita es idéntica al concepto de tunelización, explicado en la Sección 4.3 en el contexto de la multidifusión IP y de IPv6.

Consideremos ahora cómo un nodo móvil envía datagramas a un corresponsal. Esto es bastante simple, ya que el nodo móvil puede dirigir sus datagramas *directamente* al corresponsal (utilizando su propia dirección permanente como la dirección de origen y la dirección del corresponsal como dirección de destino). Puesto que el nodo móvil conoce la dirección del corresponsal, no hay ninguna necesidad de enrutar el datagrama a través del agente propio. Esto se muestra en el paso 4 de la Figura 7.24.

Resumamos nuestras explicaciones acerca del enrutamiento indirecto indicando la nueva funcionalidad requerida en la capa de red para dar soporte a la movilidad.

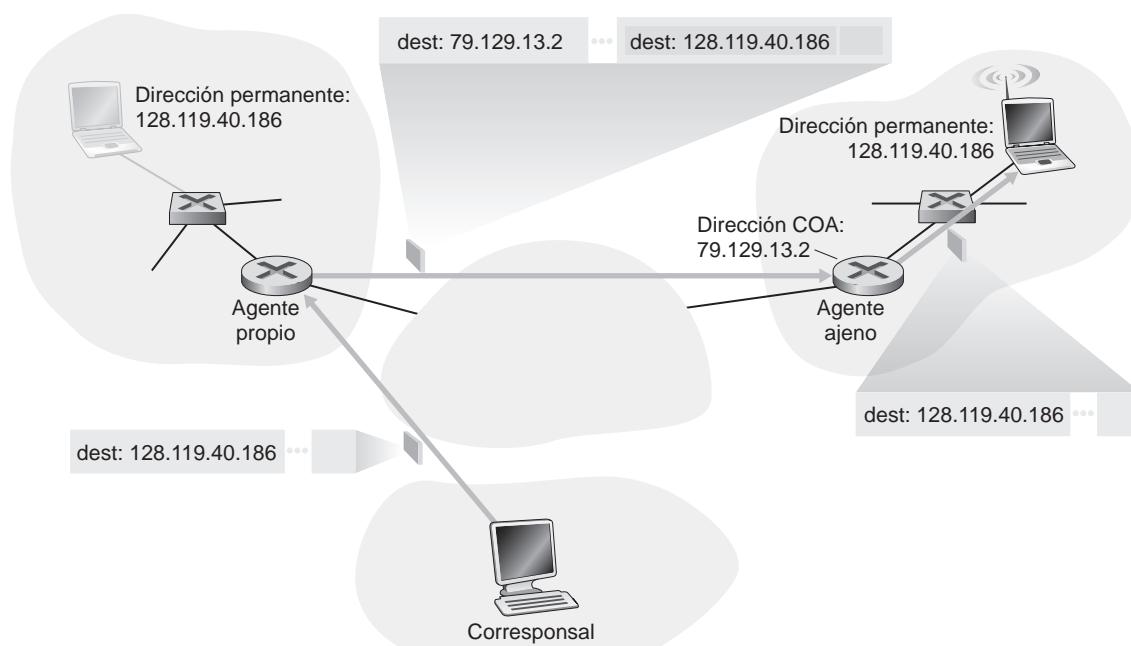


Figura 7.25 ♦ Encapsulación y desencapsulación.

- *Un protocolo entre el nodo móvil y el agente ajeno.* El nodo móvil se registrará ante el agente ajeno cuando se conecte a la red ajena. De forma similar, el nodo móvil se desegistrará ante el agente ajeno cuando abandone la red ajena.
- *Un protocolo de registro entre el agente ajeno y el agente propio.* El agente ajeno registrará la COA del nodo móvil ante el agente propio. El agente ajeno no necesita desegistrar explícitamente una COA cuando un nodo móvil abandona su red, porque el subsiguiente registro de una nueva COA, cuando el nodo móvil se desplace a otra red, se encargará de ello.
- *Un protocolo de encapsulación de datagramas para el agente propio.* Este protocolo se encargará de la encapsulación y del reenvío del datagrama original del corresponsal dentro de un datagrama dirigido a la COA.
- *Un protocolo de desencapsulación para el agente ajeno.* Este protocolo se encargará de la extracción del datagrama original del corresponsal a partir del datagrama encapsulante y del reenvío del datagrama original al nodo móvil.

La exposición anterior proporciona todos los elementos (agentes ajenos, el agente propio y reenvío indirecto) necesarios para que un nodo móvil pueda mantener una conexión activa mientras se desplaza de una red a otra. Como ejemplo del modo en que estos elementos encajan, suponga que el nodo móvil está conectado a la red ajena A, que ha registrado una COA en la red A ante su agente propio y que está recibiendo datagramas que se están enruteando indirectamente a través de su agente propio. El nodo móvil se desplaza ahora a la red ajena B y se registra ante el agente ajeno de esa red, el cual informa al agente propio de la nueva COA del nodo móvil. A partir de ese momento, el agente propio re-enruteará los datagramas hacia la red ajena B. En lo que a un corresponsal respecta, la movilidad es transparente, ya que los datagramas se enrutan a través del mismo agente propio, tanto antes como después de que el nodo móvil se haya desplazado. En lo que respecta al agente propio, no existe ninguna interrupción en el flujo de datagramas, ya que los datagramas que vayan llegando se reenvían primero hacia la red ajena A y, después del cambio de COA, se reenvían a la red ajena B. ¿Pero experimentará el nodo móvil una interrupción en el flujo de datagramas mientras se desplaza de una red a la otra? Mientras que el tiempo que transcurre entre la desconexión del nodo móvil de la red A (en cuyo momento ya no puede recibir datagramas a través de A) y la conexión a la red B (en cuyo momento registrará una nueva COA ante su agente propio) sea pequeño, solo se perderán unos pocos datagramas. Recuerde del Capítulo 3 que las conexiones terminal a terminal pueden sufrir pérdidas de datagramas debidas a la congestión de la red. Por tanto, la pérdida ocasional de datagramas en una conexión cuando un nodo se desplaza de una red a otra no constituye en modo alguno un problema catastrófico. Si hiciera falta una comunicación libre de pérdidas, los mecanismos de la capa superior podrían recuperarse de la pérdida de datagramas, independientemente de si dicha pérdida es el resultado de una congestión en la red o de la movilidad del usuario.

En el estándar de IP móvil [RFC 5944], del que hablaremos en la Sección 7.6, se utiliza una técnica de enrutamiento indirecto.

Enrutamiento directo hacia un nodo móvil

La técnica basada en el enrutamiento indirecto ilustrada en la Figura 7.24 se ve aquejada de una inefficiencia conocida como el **problema del enrutamiento triangular**: los datagramas dirigidos al nodo móvil deben enrutararse en primer lugar hacia el agente propio y luego hacia la red ajena, aún cuando exista una ruta mucho más eficiente entre el corresponsal y el nodo móvil. En el caso peor, imagine un usuario móvil que está visitando la red ajena de un colega. Los dos están sentados uno al lado del otro e intercambiando datos a través de la red. Los datagramas del corresponsal (en este caso, el colega del visitante) se enrutarán hacia el agente propio del usuario móvil y luego volverán a la red ajena.

El **enrutamiento directo** elimina la inefficiencia del enrutamiento triangular, pero el precio que hay que pagar es una mayor complejidad. Con la técnica del enrutamiento directo, un **agente**

corresponsal situado en la red del corresponsal determina primero la COA del nodo móvil. Esto puede conseguirse haciendo que el agente corresponsal consulte al agente propio, suponiendo que (como en el caso del enrutamiento indirecto) el nodo móvil tiene registrado un valor actualizado de su COA ante el agente propio. También es posible que el propio corresponsal lleve a cabo la función del agente corresponsal, al igual que el nodo móvil puede realizar la función del agente ajeno. Esto se muestra en los pasos 1 y 2 de la Figura 7.26. Entonces, el agente corresponsal tuneliza los datagramas (pasos 3 y 4 de la Figura 7.26) directamente hacia la COA del nodo móvil, de forma análoga a la tunelización que lleva a cabo el agente propio.

Aunque el enrutamiento directo resuelve el problema del enrutamiento triangular, plantea dos problemas adicionales de importancia:

- Hace falta un **protocolo de localización de usuarios móviles** para que el agente corresponsal consulte al agente propio con el fin de obtener la COA del nodo móvil (pasos 1 y 2 de la Figura 7.26).
- Cuando el nodo móvil se desplaza de una red ajena a otra, ¿cómo podremos entonces reenviar los datos hacia la nueva red ajena? En el caso del enrutamiento indirecto, este problema se resolvía fácilmente actualizando la COA mantenida por el agente propio. Sin embargo, con el enrutamiento directo, el agente corresponsal solo consulta la COA una vez al agente propio, al inicio de la sesión. Por tanto, la actualización de la COA en el agente propio, aunque sigue siendo necesaria, no será suficiente para resolver el problema de cómo enrutar los datos hacia la nueva red ajena del nodo móvil.

Una solución sería crear un nuevo protocolo para notificar al corresponsal el cambio de la COA. Otra solución alternativa, que veremos adoptada en la práctica en las redes GSM, funciona

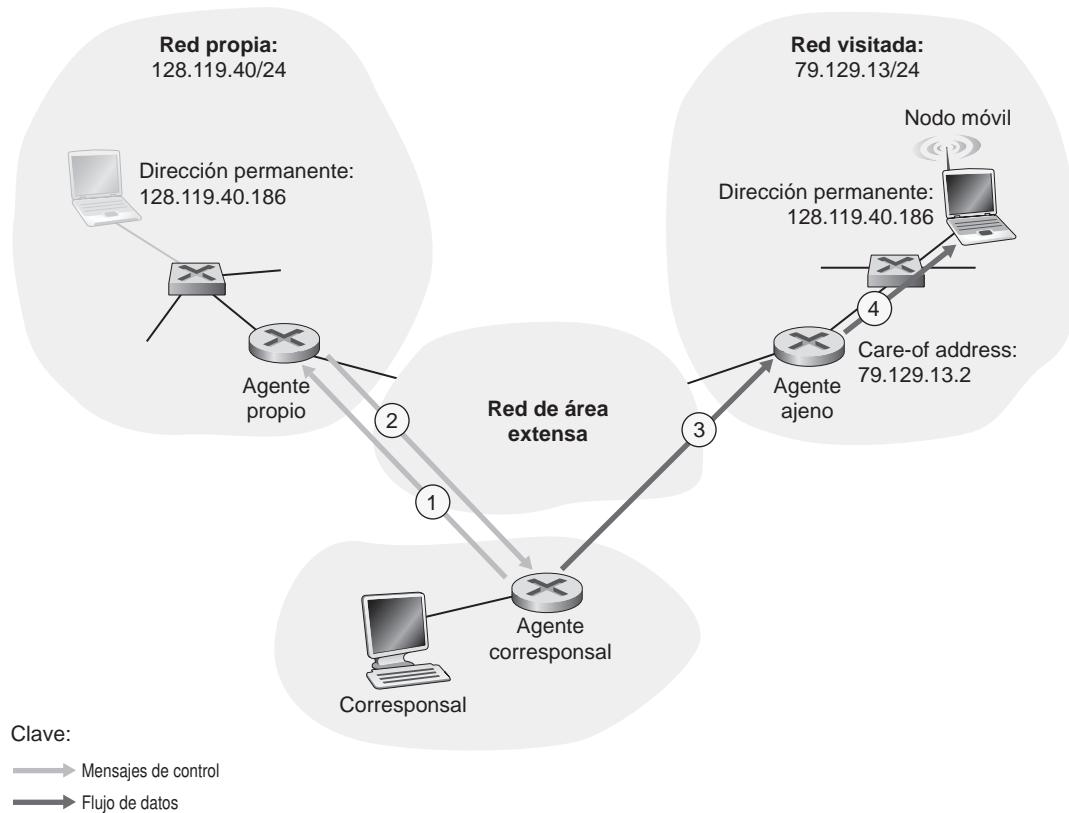


Figura 7.26 ♦ Enrutamiento directo hacia un usuario móvil.

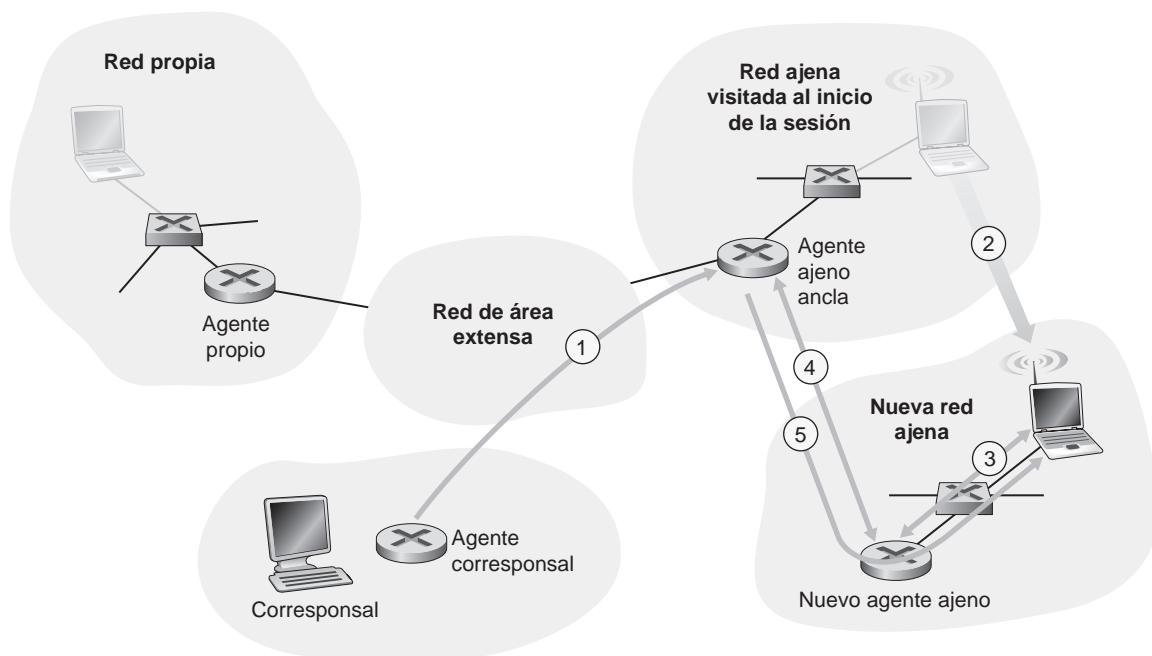


Figura 7.27 ♦ Transferencia de móviles entre redes con enruteamiento directo.

de la forma siguiente: suponga que los datos están siendo reenviados actualmente hacia el nodo móvil en la red ajena en la que dicho nodo estaba ubicado en el momento de comenzar la sesión (paso 1 de la Figura 7.27). Identificaremos al agente ajeno de dicha red ajena en la que el nodo móvil se encontraba inicialmente con el nombre de **agente ajeno ancla**. Cuando el nodo móvil se desplaza a una nueva red ajena (paso 2 de la Figura 7.27) se registra ante el nuevo agente ajeno (paso 3) y el nuevo agente ajeno proporciona al agente ajeno ancla la nueva COA del nodo móvil (paso 4). Cuando el agente ajeno ancla recibe un datagrama encapsulado para un nodo móvil que ya ha salido de su red, puede entonces reencapsular el datagrama y reenviarlo al nodo móvil (paso 5) utilizando la nueva COA. Si el nodo móvil se desplaza posteriormente otra vez a otra nueva red ajena, el agente ajeno de esa nueva red visitada contactará con el agente ajeno ancla para establecer el reenvío hacia esta nueva red ajena.

7.6 IP móvil

Los protocolos y la arquitectura de Internet necesarios para dar soporte a la movilidad, colectivamente conocidos con el nombre de IP móvil, están definidos principalmente en RFC 5944 para IPv4. IP móvil es un estándar flexible, que soporta muchos modos distintos de operación (por ejemplo, operación con o sin un agente ajeno), múltiples formas de que los agentes y los nodos móviles se descubran entre sí, utilización de direcciones COA únicas o múltiples y diversas formas de encapsulación. Como tal, IP móvil es un estándar complejo y haría falta un libro completo para describirlo en detalle; de hecho, uno de tales libros es [Perkins 1998b]. Nuestro objetivo aquí, mucho más modesto, es proporcionar una panorámica de los aspectos más importantes de IP móvil e ilustrar su uso en unos cuantos escenarios comunes.

La arquitectura de IP móvil contiene muchos de los elementos que hemos presentado anteriormente, incluyendo los conceptos de agente propio, agente ajeno, direcciones COA y encapsulación/desencapsulación. El estándar actual [RFC 5944] especifica el uso del enruteamiento indirecto hacia el nodo móvil.

El estándar IP móvil consta de tres elementos principales:

- *Descubrimiento de agentes.* IP móvil define los protocolos utilizados por un agente propio o ajeno para anunciar sus servicios a los nodos móviles, así como protocolos para que los nodos móviles soliciten los servicios de un agente ajeno o propio.
- *Registro ante el agente propio.* IP móvil define los protocolos utilizados por el nodo móvil y/o el agente ajeno para registrar y desregar direcciones COA ante el agente propio de un nodo móvil.
- *Enrutamiento indirecto de los datagramas.* El estándar también define la forma en que el agente propio reenvía los datagramas hacia los nodos móviles, incluyendo reglas para el reenvío de datagramas, reglas para la gestión de las condiciones de error y diversas formas de encapsulación [RFC 2003, RFC 2004].

Las consideraciones de seguridad tienen una gran importancia a lo largo de todo el estándar de IP móvil. Por ejemplo, es clara la necesidad de que un nodo móvil se autentique para garantizar que los usuarios maliciosos no puedan registrar una dirección COA falsa ante un agente propio que haría que todos los datagramas destinados a una dirección IP fueran redirigidos hacia el usuario malicioso. En IP móvil la seguridad se consigue utilizando muchos de los mecanismos que examinaremos en el Capítulo 8, por lo que en las explicaciones que siguen no vamos a tocar los problemas de seguridad.

Descubrimiento de agentes

Un nodo de IP móvil que llegue a una nueva red, independientemente de si está conectando a una red ajena o volviendo a la suya propia, debe averiguar la identidad del agente ajeno o propio correspondiente. De hecho, es el descubrimiento de un nuevo agente ajeno, con una nueva dirección de red, el que permite a la capa de red del nodo móvil averiguar que acaba de entrar en una nueva red ajena. Este proceso se conoce con el nombre de **descubrimiento de agentes**. El descubrimiento de agentes puede realizarse de una de dos formas: mediante los anuncios de los agentes o mediante las solicitudes de agente.

Con el **anuncio de agente**, cada agente ajeno o propio anuncia sus servicios utilizando una extensión del protocolo existente de descubrimiento de router [RFC 1256]. El agente difunde periódicamente un mensaje ICMP con un campo de tipo de valor 9 (descubrimiento de router) a través de todos los enlaces con los que esté conectado. El mensaje de descubrimiento de router contiene la dirección IP del router (es decir, el agente), permitiendo así que los nodos móviles conozcan la dirección IP del agente. El mensaje de descubrimiento de router también contiene una extensión de anuncio de agente de movilidad que contiene información adicional que el nodo móvil necesita. Entre los campos más importantes de esa extensión se encuentran los siguientes:

- *Bit de agente propio (H).* Indica que el agente es un agente propio para la red en la que reside.
- *Bit de agente ajeno (F).* Indica que el agente es un agente ajeno para la red en la que reside.
- *Bit de registro requerido (R).* Indica que los usuarios móviles en esta red *deberán* registrarse ante un agente ajeno. En particular, un usuario móvil no podrá obtener una dirección COA en la red ajena (por ejemplo, utilizando DHCP) y asumir él mismo la funcionalidad del agente ajeno sin antes registrarse ante el agente ajeno.
- *Bits de encapsulación M, G.* Indican si se utilizará algún tipo de encapsulación distinta de la encapsulación IP-en-IP.
- *Campos de dirección COA.* Es una lista de una o más direcciones COA proporcionada por el agente ajeno. En el ejemplo que indicamos más adelante, la dirección COA estará asociada con el agente ajeno, que recibirá los datagramas destinados a esa COA y luego los reenviará al nodo móvil apropiado. El usuario móvil seleccionará una de estas direcciones como dirección COA a la hora de registrarse ante su agente propio.

La Figura 7.28 ilustra algunos de los campos clave contenidos en el mensaje de anuncio de agente.

Con la **solicitud de agente**, un nodo móvil que quiera averiguar información acerca de los agentes sin esperar a recibir un anuncio de agente puede difundir un mensaje de solicitud de agente, que es simplemente un mensaje ICMP con un campo de tipo que contiene el valor 10. Un agente que reciba la solicitud enviará directamente al nodo móvil un anuncio de agente, mediante un mensaje de unidifusión, pudiendo entonces el nodo móvil proceder como si hubiera recibido un anuncio no solicitado.

Registro ante el agente propio

Una vez que un nodo IP móvil ha recibido una dirección COA, debe registrar dicha dirección ante su agente propio. Esto puede hacerse a través del agente ajeno (que se encargará de registrar la COA ante el agente propio) o lo puede hacer también directamente el propio nodo de IP móvil. Vamos a analizar el primero de los casos, que consta de cuatro pasos:

1. Despues de recibir el anuncio de un agente ajeno, el nodo móvil envía un mensaje de registro de IP móvil a ese agente ajeno. El mensaje de registro se transporta dentro de un datagrama UDP y se envía al puerto 434. El mensaje de registro incluye una dirección COA anunciada por el agente ajeno, además de la dirección del agente propio (HA), la dirección permanente del nodo móvil (MA), el tiempo de vida solicitado para el registro y un identificador de registro de 64 bits. El tiempo de vida solicitado para el registro es el número de segundos durante los cuales el registro tendrá validez. Si el registro no se renueva ante el agente propio dentro de ese tiempo de vida especificado el registro quedará invalidado. El identificador de registro actúa como un número de secuencia y sirve para establecer la correspondencia entre una respuesta de registro recibida y una solicitud de registro, como se indica más adelante.
2. El agente ajeno recibe el mensaje de registro y anota la dirección IP permanente del nodo móvil. El agente ajeno ahora sabe que debe buscar datagramas que contengan un datagrama encapsulado cuya dirección de destino coincida con esa dirección permanente del nodo móvil. El agente ajeno envía a continuación un mensaje de registro de IP móvil (de nuevo dentro de un datagrama UDP) al puerto 434 del agente propio. El mensaje contiene las direcciones COA,

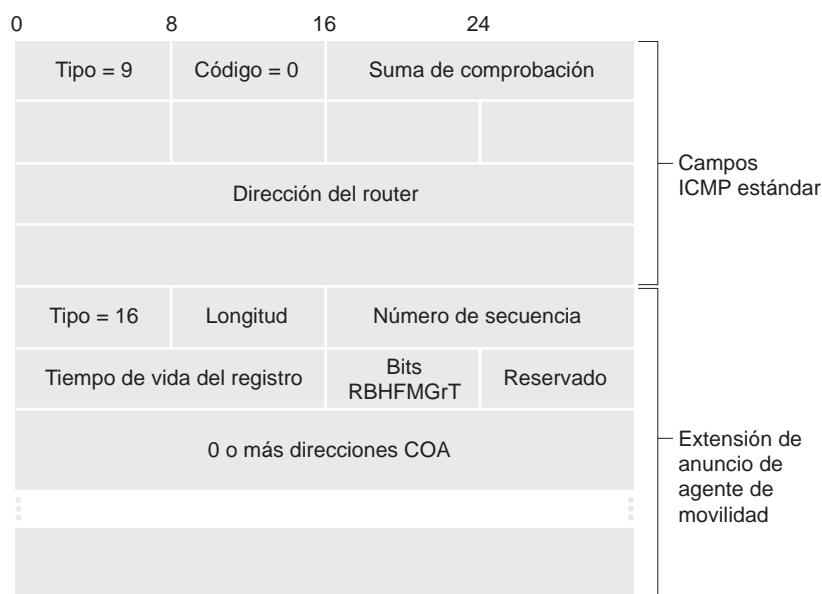


Figura 7.28 ♦ Mensaje de descubrimiento de router ICMP con la extensión de anuncio de agente de movilidad.

HA y MA, el formato de encapsulación solicitado, el tiempo de vida de registro solicitado y el identificador de registro.

3. El agente propio recibe la solicitud de registro y comprueba la autenticidad y la corrección de la misma. El agente propio establece una asociación entre la dirección IP permanente del nodo móvil y la dirección COA. De este modo, en el futuro, los datagramas que el agente propio reciba y que estén dirigidos al nodo móvil serán encapsulados y tunelizados hacia la dirección COA. El agente propio envía una respuesta de registro de IP móvil que contiene las direcciones HA y MA, el tiempo de vida real del registro y el identificador de registro correspondiente a la solicitud que se esté satisfaciendo con esta respuesta.
4. El agente ajeno recibe la respuesta de registro y a continuación la reenvía hacia el nodo móvil.

Llegados a este punto, el proceso de registro estará terminado y el nodo móvil podrá recibir los datagramas enviados hacia su dirección permanente. La Figura 7.29 ilustra estos pasos. Observe que el agente propio especifica un tiempo de vida más pequeño que el tiempo de vida solicitado por el nodo móvil.

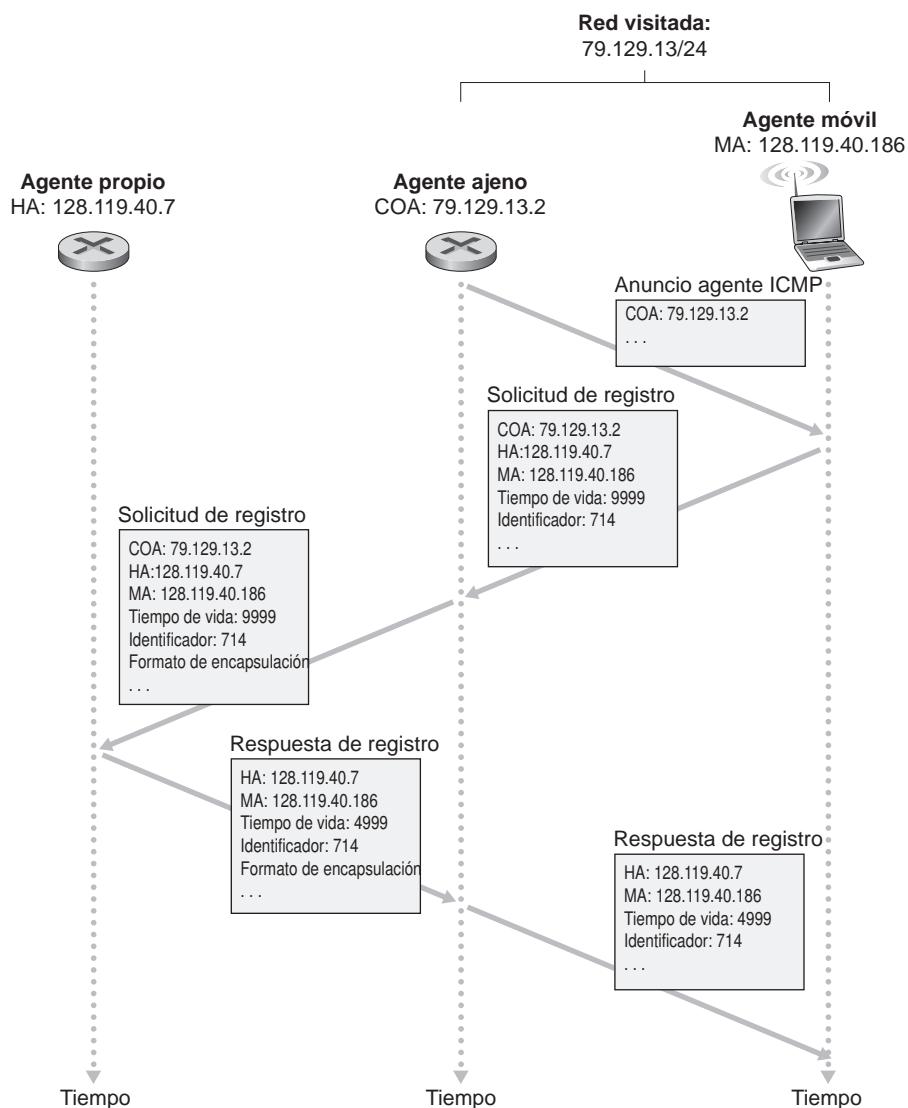


Figura 7.29 ♦ Anuncio de agente y registro de IP móvil.

Un agente ajeno no necesita des registrar explícitamente una dirección COA cuando un nodo móvil abandone su red. Ese proceso de cancelación de la dirección tendrá lugar automáticamente cuando el nodo móvil se desplace a una red nueva (independientemente de si es otra red ajena o si se trata de su red propia) y registre una nueva dirección COA.

El estándar de IP móvil permite muchas capacidades y escenarios adicionales a los que hemos descrito. El lector interesado puede consultar [Perkins 1998b; RFC 5944].

7.7 Gestión de la movilidad en redes celulares

Habiendo examinado cómo se gestiona la movilidad en las redes IP, volvamos ahora nuestra atención a otras redes que tienen una historia mucho más dilatada de soporte de la movilidad: las redes de telefonía celular. Mientras que en la Sección 7.4 nos hemos centrado en el enlace inalámbrico del primer salto dentro de las redes celulares, aquí vamos a centrarnos en la movilidad, utilizando como caso de estudio la arquitectura de las redes celulares GSM [Goodman 1997; Mouly 1992; Scourias 2012; Kaaranen 2001; Korhonen 2003; Turner 2012], dado que se trata de una tecnología madura y de amplia implantación. La movilidad en las redes 3G y 4G es similar en sus principios a la que emplea en las redes GSM. Como en el caso de IP móvil, veremos que la arquitectura de las redes GSM integra varios de los principios fundamentales que hemos identificado en la Sección 7.5.

Al igual que IP móvil, GSM adopta una técnica basada en el enrutamiento indirecto (véase la Sección 7.5.2), enrutando primero la llamada del correspondiente hacia la red propia del usuario móvil y de allí a la red visitada. En terminología GSM, la red propia del usuario móvil se denomina **red móvil terrestre pública propia (home PLMN, home Public Land Mobile Network)**. Dado que el acrónimo PLMN resulta un tanto complicado de pronunciar y dado que tratamos de evitar una sopa de letras de acrónimos, en lo sucesivo nos referiremos a la PLMN propia de GSM simplemente con el nombre de **red propia**. La red propia es el proveedor de telefonía celular con el que está abonado el usuario móvil (es decir, el proveedor que factura al usuario por los servicios mensuales de telefonía celular). La PLMN visitada, a la que denominaremos sencillamente **red visitada**, es la red en la que reside actualmente el usuario móvil.

Como en el caso de IP móvil, las responsabilidades de las redes propia y visitada son bastante distintas.

- La red propia mantiene una base de datos que se conoce con el nombre de **registro de ubicaciones propias (HLR, Home Location Register)**, que contiene el número de teléfono celular permanente y la información del perfil de abonado para cada uno de sus abonados. Es importante resaltar que la base de datos HLR también contiene información acerca de las ubicaciones actuales de estos abonados. Es decir, si un usuario móvil está actualmente en situación de itinerancia (*roaming*) dentro de la red celular de otro proveedor, la HLR contendrá suficiente información como para obtener (a través de un proceso que vamos a describir en breve) una dirección, dentro de la red visitada, hacia la que enrutar las llamadas dirigidas al usuario móvil. Como veremos, un conmutador especial dentro de la red propia, conocido como **centro de conmutación pasarela para servicios móviles (GMSC, Gateway Mobile services Switching Center)** es contactado por el correspondiente cada vez que realiza una llamada a un usuario móvil. De nuevo, para tratar de evitar el uso excesivo de acrónimos, en lo sucesivo nos vamos a referir al GMSC con el término más descriptivo de **MSC propio**.
- La red visitada mantiene una base de datos conocida con el nombre de **registro de ubicación de visitantes (VLR, Visitor Location Register)**. La base de datos VLR contiene una entrada para cada usuario móvil que se encuentra *actualmente* en la parte de la red a la que da servicio VLR. Las entradas de VLR aparecen y desaparecen, por tanto, a medida que los usuarios móvi-

les entran y salen de la red. La base de datos VLR normalmente está co-ubicada con el centro de conmutación móvil (MSC) que coordina el establecimiento de llamadas hacia y desde la red visitada.

En la práctica, la red celular de un proveedor sirve como red propia para sus abonados y como red visitada para los usuarios móviles que estén abonados a un proveedor diferente de telefonía celular.

7.7.1 Enrutamiento de llamadas hacia un usuario móvil

Ahora podemos describir cómo se realiza una llamada a un usuario móvil GSM que se encuentra en una red ajena. A continuación proporcionamos un ejemplo simple; el lector interesado puede encontrar otros escenarios más complejos en [Mouly 1992]. Los pasos, ilustrados en la Figura 7.30, son los siguientes:

1. El corresponsal marca el número telefónico del usuario móvil. Este número no hace referencia por sí mismo a ninguna línea telefónica o ubicación concreta (después de todo, el número telefónico es fijo, mientras que el usuario es móvil). Los primeros dígitos del número son suficientes para identificar globalmente la red propia a la que el móvil pertenece. La llamada será enrutada desde el corresponsal, a través de la red telefónica conmutada pública (PSTN, *Public Switched Telephone Network*), hasta el MSC propio de la red propia del móvil. Éste es el primer tramo de la llamada.
2. El MSC propio recibe la llamada e interroga a HLR para determinar la ubicación del usuario móvil. En el caso más simple, HLR devuelve el **número de itinerancia de la estación móvil (MSRN, Mobile Station Roaming Number)**, al que en lo sucesivo denominaremos simplemente **número de itinerancia**. Observe que este número es diferente del número telefónico permanente del móvil que está asociado con la red propia del móvil. El número de itinerancia es efímero: es asignado temporalmente al móvil cuando entra dentro de una red visitada. El número de itinerancia desempeña un papel similar al de la dirección COA en IP móvil y, al igual

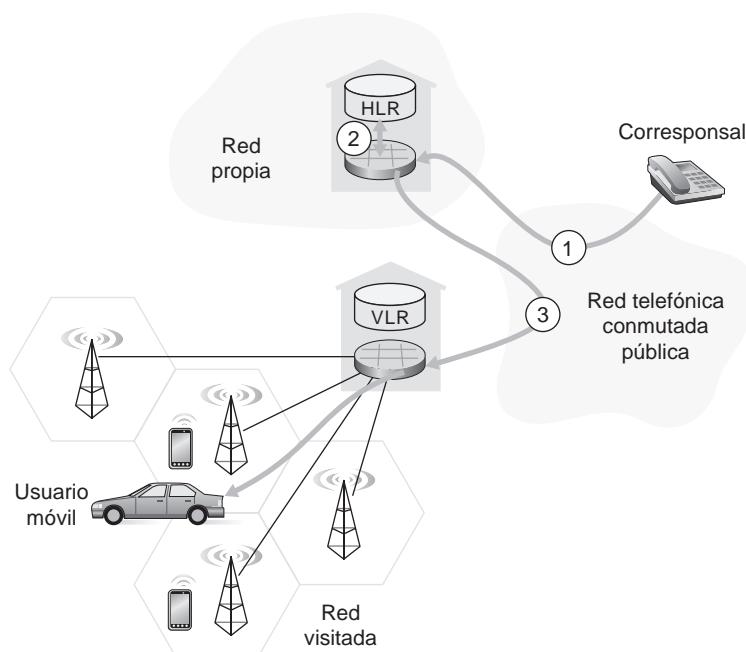


Figura 7.30 ◆ Establecimiento de una llamada hacia un usuario móvil: enrutamiento indirecto.

que la dirección COA, es invisible para el corresponsal y para el móvil. Si el registro HLR no tiene el número de itinerancia, devuelve la dirección del VLR en la red visitada. En este caso (no mostrado en la Figura 7.30), el MSC propio necesitará consultar a ese VLR para obtener el número de itinerancia del nodo móvil. Pero, ¿cómo obtiene el HLR el número de itinerancia o la dirección del VLR? ¿Qué sucede con estos valores cuando el usuario móvil se desplaza a otra red visitada? Analizaremos estas importantes cuestiones enseguida.

3. Conocido el número de itinerancia, el MSC propio establece el segundo tramo de la llamada a través de la red hasta el MSC de la red visitada. Con ello, la llamada se habrá completado, produciéndose el enrutamiento desde el corresponsal hasta el MSC propio, de este al MSC visitado y de ahí a la estación base que da servicio al usuario móvil.

Una cuestión no resuelta en el paso 2 es cómo el HLR obtiene la información acerca de la ubicación del usuario móvil. Cuando se enciende un teléfono móvil o este entra en una parte de una red visitada que está cubierta por un nuevo VLR, el móvil debe registrarse ante la red visitada. Esto se realiza intercambiando mensajes de señalización entre el móvil y el VLR. El VLR visitado, a su vez, envía un mensaje de solicitud de actualización de la ubicación al HLR del móvil. Este mensaje informa al HLR o bien del número de itinerancia a través del cual se puede contactar con el móvil, o de la dirección del VLR (al que se puede consultar más adelante para obtener el número del móvil). Como parte de este intercambio, el VLR obtiene también información de abonado de la base de datos HLR acerca del móvil y determina qué servicios (en su caso) tiene que proporcionar la red visitada al usuario móvil.

7.7.2 Transferencia de llamadas en GSM

Cuando una estación móvil cambia su asociación de una estación base a otra durante una llamada se produce lo que se denomina **transferencia o cesión de la llamada** (*handoff*). Como se muestra en la Figura 7.31, la llamada del móvil es enrutada inicialmente (antes de la transferencia) hacia el móvil a través de una estación base (a la que denominaremos estación base antigua) mientras que después de la transferencia se enruta hacia el móvil a través de otra estación base (a la que denominaremos nueva estación base). Observe que una transferencia de llamada entre las estaciones base da como resultado no solo cuando el móvil transmite/reciba hacia/desde una estación base nueva, sino también cuando la llamada activa se re-enrute desde un punto de comutación dentro de la red hacia la nueva estación base. Inicialmente, vamos a suponer que las estaciones base nueva y antigua comparten el mismo MSC y que dicho reenrutamiento tiene lugar en ese MSC.

Puede haber varias razones para que se produzca la transferencia de la llamada, entre las que podemos citar (1) que la señal entre la estación base actual y el móvil se puede haber deteriorado hasta tal punto que existe riesgo de que la llamada se interrumpa y (2) que una celda pueda haberse

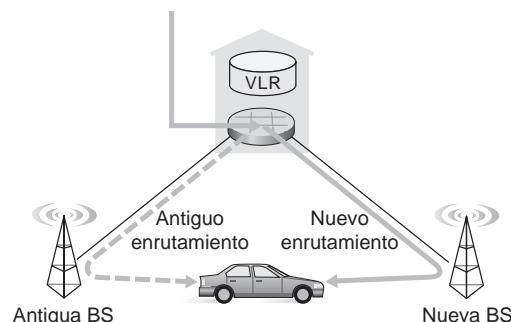


Figura 7.31 ♦ Escenario de transferencia de llamadas entre estaciones base con un MSC común.

sobrecargado debido a que está gestionando un gran número de llamadas. Esta congestión puede aliviarse transfiriendo usuarios móviles a otras celdas cercanas menos congestionadas.

Mientras que está asociado con una estación base, el móvil mide periódicamente la intensidad de una señal baliza recibida desde su estación base actual, así como señales baliza procedentes de estaciones base cercanas que el móvil pueda “escuchar”. Estas medidas son reenviadas una o dos veces por segundo hacia la estación base actual del móvil. Es la propia estación base antigua la que inicia la transferencia de una llamada en GSM basándose en estas medidas, en la carga actual de móviles existente en las celdas cercanas y en otros factores [Mouly 1992]. El estándar GSM no especifica el algoritmo concreto que una estación base debe utilizar a la hora de determinar si transferir o no una llamada.

La Figura 7.32 ilustra los pasos que se llevan a cabo cuando una estación base decide transferir un usuario móvil:

1. La estación base (BS) antigua informa al MSC visitado de que hay que realizar una transferencia, así como de la estación base (o del posible conjunto de estaciones base) a la que hay que transferir el móvil.
2. El MSC visitado inicia las operaciones de establecimiento de la ruta hacia la nueva estación base, asignando los recursos necesarios para transportar la llamada re-enrutada y señalizando a la nueva estación base que se va a producir una transferencia.
3. La nueva estación base asigna y activa un nuevo canal de radio para que lo utilice el móvil.
4. La nueva estación base envía de vuelta al MSC visitado y a la estación base antigua una señal indicativa de que se ha establecido la ruta entre el MSC visitado y la nueva estación base, y de que hay que informar al móvil de la transferencia que se va a producir. La nueva estación base proporciona toda la información que el móvil necesitará para asociarse con ella.
5. El móvil es informado de que debe realizar una transferencia. Observe que hasta este punto, el móvil ha estado completamente ignorante de que la red estaba realizando el trabajo necesario para llevar a cabo una transferencia (por ejemplo, asignar un canal en la nueva estación base y asignar una ruta entre el MSC visitado y la nueva estación base).
6. El móvil y la nueva estación base intercambian uno o más mensajes para activar completamente el nuevo canal en la estación base nueva.
7. El móvil envía a la nueva estación base un mensaje indicando que se ha completado la transferencia, el cual es reenviado hacia el MSC visitado. Entonces, el MSC visitado re-enruta la llamada activa hacia el móvil a través de la nueva estación base.
8. Por último, se liberan los recursos asignados en la ruta que llevaba hacia la antigua estación base.

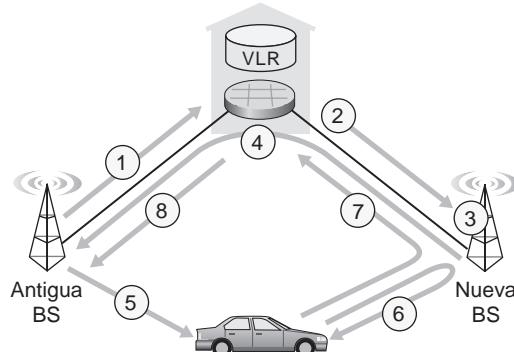


Figura 7.32 ♦ Pasos necesarios para realizar una transferencia entre estaciones base con un MSC común.

Vamos a concluir nuestro análisis de transferencia de llamadas considerando lo que ocurre cuando el móvil se desplaza a una estación base que está asociada con un MSC *diferente* al de la antigua estación base, y también lo que sucede cuando este tipo de transferencia entre dos MSC tiene lugar más de una vez. Como se muestra en la Figura 7.33, GSM define el concepto de **MSC ancla**. El MSC ancla es el MSC que estaba siendo visitado por el móvil cuando da comienzo una llamada; por tanto, el MSC ancla sigue siendo el mismo a todo lo largo de la llamada. Mientras dure la llamada e independientemente del número de transferencias inter-MSC realizadas por el móvil, la llamada es enrutada desde el MSC propio hasta el MSC ancla y después desde el MSC ancla al MSC visitado en el que el móvil esté situado actualmente. Cuando un móvil se desplaza desde el área de cobertura de un MSC a la de otro, la llamada activa se re-enruta desde el MSC ancla hasta el nuevo MSC visitado que contiene a la nueva estación base. Por tanto, en cada momento existen, como mucho, tres MSC (el MSC propio, el MSC ancla y el MSC visitado) entre el correspondiente y el móvil. La Figura 7.33 ilustra el enrutamiento de una llamada entre los MSC visitados por un usuario móvil.

En lugar de mantener un único salto de MSC desde el MSC ancla al MSC actual, una técnica alternativa habría sido encadenar simplemente los MSC visitados por el móvil, haciendo que cada MSC antiguo reenvíe la llamada activa al nuevo MSC cada vez que el móvil se desplace hasta un nuevo MSC. Dicho encadenamiento de centros MSC puede tener lugar en las redes celulares IS-41, con un paso opcional de minimización de la ruta para eliminar los MSC existentes entre el MSC ancla y el MSC visitado actual [Lin 2001].

Terminamos esta exposición acerca de la gestión de la movilidad en GSM con una comparación de la gestión de movilidad en GSM e IP móvil. La comparación en la Tabla 7.2 indica que aunque las redes celulares e IP son fundamentalmente distintos en muchos aspectos, comparten un sorprendente número de elementos funcionales comunes y de técnicas globales a la hora de gestionar la movilidad.

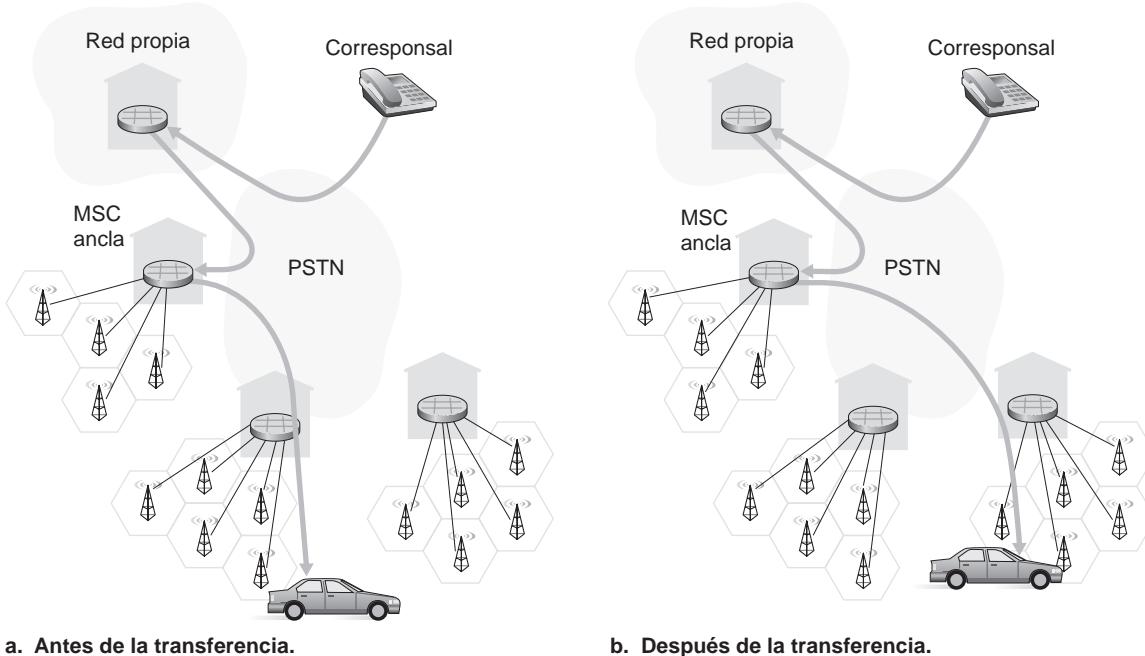


Figura 7.33 ♦ Re-enrutamiento a través de un MSC ancla.

Elemento GSM	Comentarios sobre el elemento GSM	Elemento IP móvil
Sistema propio	Red a la que pertenece el número de teléfono permanente del usuario móvil.	Red propia
Centro de conmutación móvil pasarela o simplemente MSC propio, Registro de ubicación propio (HLR)	MSC propio: punto de contacto para obtener la dirección enrutable del usuario móvil. HLR: base de datos en el sistema propio que contiene el número de teléfono permanente, la información de perfil, la ubicación actual del usuario móvil y la información de abono.	Agente propio
Sistema visitado	Red distinta del sistema propio en la que reside actualmente el usuario móvil.	Red visitada
Centro de conmutación de servicios móviles visitados, Registro de ubicación de visitantes (VLR)	MSC visitado: responsable de establecer las llamadas hacia/desde los nodos móviles en las celdas asociadas con el MSC. VLR: entrada temporal en la base de datos dentro del sistema visitado, que contiene información de suscripción para cada usuario móvil visitante.	Agente ajeno
Número de itinerancia de la estación móvil (MSRN) o simplemente número de itinerancia	Dirección enrutable para el segmento de llamada telefónica entre el MSC propio y el MSC visitado, que no es visible ni para el móvil ni para el correspondiente.	Dirección COA

Tabla 7.2 • Aspectos comunes entre IP móvil y GSM en lo que respecta a la movilidad.

7.8 Tecnología inalámbrica y movilidad: impacto sobre los protocolos de las capas superiores

En este capítulo hemos visto que las redes inalámbricas difieren significativamente de sus contrapartidas cableadas tanto en la capa de enlace (como resultado de características del canal inalámbrico tales como el desvanecimiento, el multicamino y los terminales ocultos) como en la capa de red (como resultado de la existencia de usuarios móviles que cambian sus puntos de asociación con la red). Pero, ¿existen diferencias importantes en las capas de transporte y de aplicación? Resulta tentador pensar que estas diferencias serán menores, dado que la capa de red proporciona a las capas superiores el modelo de servicio de entrega de mejor esfuerzo tanto en las redes cableadas como en las inalámbricas. De forma similar, si se utilizan protocolos como TCP o UDP para proporcionar los servicios de la capa de transporte a las aplicaciones tanto en redes inalámbricas como cableadas, entonces la capa de aplicación tampoco tendría por qué ser modificada. En un cierto sentido, nuestra intuición es correcta: TCP y UDP pueden (y de hecho lo hacen) operar en redes con enlaces inalámbricos. Por otra parte, los protocolos de transporte en general, y TCP en particular, pueden tener en ocasiones un rendimiento muy distinto en las redes cableadas e inalámbricas, y es aquí, en términos de rendimiento, que las diferencias se manifiestan. Veamos por qué.

Recuerde que TCP retransmite los segmentos perdidos o corruptos a lo largo de la ruta existente entre el emisor y el receptor. En el caso de usuarios móviles, las pérdidas pueden ser el resultado de la congestión de la red (desbordamiento de los buffers de los routers) o de la transferencia de llamadas (por ejemplo, debido a retardos en el re-enrutamiento de segmentos hacia el nuevo punto de asociación con la red de un móvil). En todos los casos, los mensajes ACK del receptor al emisor de TCP solo indican que un segmento no ha sido recibido intacto; el emisor no es consciente de si el segmento se ha perdido a causa de la congestión, durante la transferencia o porque se han detectado bits erróneos. En todos los casos, la respuesta del emisor es la misma: retransmitir el segmento. La respuesta del control de congestión de TCP es también siempre la misma en todos los casos: TCP disminuye el tamaño de su ventana de congestión como se explica en la Sección 3.7. Reduciendo incondicionalmente su ventana de congestión, TCP asume implícitamente que las pérdidas de segmento se deben a la congestión más que a la corrupción o a la transferencia de llamadas. Pero, como hemos visto en la Sección 7.2, los errores de bit son mucho más comunes

en las redes inalámbricas que en las redes cableadas. Cuando se producen tales errores de bits o cuando tienen lugar pérdidas debido al mecanismo de transferencia no existe realmente ninguna razón para que el emisor TCP reduzca su ventana de congestión (y por tanto su velocidad de envío). De hecho, podría darse perfectamente el caso de que los buffers de los routers estén vacíos y de que los paquetes estén fluyendo a lo largo de la ruta terminal a terminal sin que la congestión represente ningún obstáculo.

Los investigadores se dieron cuenta a principios y mediados de la década de 1990 de que dadas las tasas de errores de bit en los enlaces inalámbricos y la posibilidad de que se produzcan pérdidas en la transferencia de llamadas, la respuesta del control de congestión de TCP podría ser problemática en una configuración inalámbrica. Para resolver este problema se pueden usar tres clases genéricas de técnicas:

- *Recuperación local.* Los protocolos de recuperación local permiten recuperarse de los errores de bit en el lugar y en el momento en que esos errores se producen (por ejemplo, en el enlace inalámbrico); un caso sería el protocolo ARQ 802.11 que hemos estudiado en la Sección 7.3, o algunas técnicas más sofisticadas que utilizan tanto ARQ como FEC [Ayanoglu 1995].
- *Conocimiento por parte del emisor TCP de enlaces inalámbricos.* En las técnicas de recuperación local, el emisor TCP es afortunadamente inconsciente de que sus segmentos están atravesando un enlace inalámbrico. Una técnica alternativa es que es el emisor y el receptor TCP sean conscientes de la existencia de un enlace inalámbrico, para distinguir entre las pérdidas por congestión que tienen lugar en la red cableada y las pérdidas/corrupciones que se producen en el enlace inalámbrico, y para invocar los mecanismos de control de congestión solo en respuesta a las pérdidas debidas a que la red cableada está congestionada. [Balakrishnan 1997] investiga diversos tipos de TCP, bajo la suposición de que los sistemas terminales puedan llevar a cabo esta distinción. [Liu 2003] investiga técnicas para distinguir entre las pérdidas de los segmentos cableados e inalámbricos de una ruta extremo a extremo.
- *Técnicas de conexión dividida.* En la técnica de conexión dividida [Bakre 1995], la conexión extremo a extremo entre el usuario móvil y el otro extremo se divide en dos conexiones de la capa de transporte: una desde el host móvil hasta el punto de acceso inalámbrico y otra desde el punto de acceso inalámbrico hasta el otro extremo de la comunicación (que asumiremos aquí que se trata de un host cableado). La conexión extremo a extremo se forma, por tanto, mediante la concatenación de una parte inalámbrica y una parte cableada. La capa de transporte a través del segmento inalámbrico puede ser una conexión TCP estándar [Bakre 1995] o un protocolo de recuperación de errores especialmente adaptado ejecutándose sobre UDP. [Yavatkar 1994] investiga el uso de un protocolo de repetición selectiva de la capa de transporte a través de la conexión inalámbrica. Las medidas en [Wei 2006] indican que las conexiones TCP divididas se utilizan ampliamente en las redes de datos celulares y que pueden, de hecho, llevarse a cabo mejoras significativas utilizando ese tipo de conexiones divididas.

Nuestro tratamiento de TCP sobre los enlaces inalámbricos ha sido necesariamente breve. Puede encontrar estudios detallados de los retos y soluciones TCP para redes inalámbricas en [Hanabali 2005; Leung 2006]. Animamos al lector a consultar las referencias para conocer más detalles acerca de esta área activa de investigación.

Habiendo considerado los protocolos de la capa de transporte, veamos a continuación el efecto de la tecnología inalámbrica y de la movilidad sobre los protocolos de la capa de aplicación. Aquí, una consideración importante es que los enlaces inalámbricos suelen tener anchos de banda relativamente bajos, como vimos en la Figura 7.2. Como resultado, las aplicaciones que operan sobre enlaces inalámbricos, particularmente, sobre enlaces inalámbricos celulares, deben tratar el ancho de banda como un recurso escaso. Por ejemplo, un servidor web que esté enviando contenido a un navegador web que se ejecuta sobre un teléfono 4G probablemente no pueda proporcionar un contenido con la misma riqueza de imágenes que el que envía a un navegador que esté operando a través de una conexión cableada. Aunque los enlaces inalámbricos presentan desafíos en la capa

de aplicación, la movilidad que permiten también hace posible un amplio conjunto de aplicaciones conscientes de la ubicación y del contexto [Chen 2000; Baldauf 2007]. En términos más generales, las redes inalámbricas y móviles desempeñarán un papel crucial a la hora de conseguir disponer de los entornos de computación ubicua del futuro [Weiser 1991]. Dicho esto, terminemos dejando claro que esto es solo la punta del iceberg en lo que respecta al impacto de las redes inalámbricas y móviles sobre las aplicaciones en red y sus protocolos.

7.9 Resumen

Las redes inalámbricas y móviles han revolucionado la telefonía y están teniendo un impacto cada vez más profundo también en el mundo de las redes de computadoras. Con su acceso no restringido, en cualquier momento y en cualquier lugar, a la infraestructura global de red, no solo están haciendo que el acceso de red sea más ubicuo, sino que también permiten un conjunto muy excitante de nuevos servicios dependientes de la ubicación. Dada la creciente importancia de las redes inalámbricas y móviles, este capítulo se ha centrado en los principios, en las tecnologías de enlaces comunes y en las arquitecturas de red necesarias para dar soporte a las comunicaciones inalámbricas y móviles.

Hemos comenzado el capítulo con una introducción a las redes inalámbricas y móviles, estableciendo una importante distinción entre los desafíos planteados por la naturaleza *inalámbrica* de los enlaces de comunicaciones de tales redes y por la *movilidad* que estos enlaces inalámbricos hacen posible. Esto nos ha permitido aislar, identificar y dominar mejor los conceptos clave de cada área. Nos hemos centrado primero en la comunicación inalámbrica, considerando las características de un enlace inalámbrico en la Sección 7.2. En las Secciones 7.3 y 7.4 hemos examinado los aspectos del nivel de enlace del estándar para redes LAN inalámbricas 802.11 del IEEE (WiFi), de dos redes de área personal IEEE 802.15 (Bluetooth y Zigbee), y del acceso celular a Internet 3G y 4G. Después hemos vuelto nuestra atención hacia el problema de la movilidad. En la Sección 7.5 hemos identificado diversas formas de movilidad, viendo que los diversos puntos a lo largo de este espectro planteaban diferentes desafíos y admitían distintas soluciones. Hemos considerado los problemas de localizar un usuario móvil y efectuar el enrutamiento hasta él, así como las técnicas para transferir a los usuarios móviles que se desplazan dinámicamente desde un punto de asociación con la red hasta otro. En las Secciones 7.6 y 7.7, respectivamente, hemos examinado cómo se resolvían estas cuestiones en el estándar IP móvil y en GSM. Por último, hemos considerado el impacto de los enlaces inalámbricos y la movilidad en los protocolos de la capa de transporte y sobre las aplicaciones en red en la Sección 7.8.

Aunque hemos dedicado un capítulo completo al estudio de las redes inalámbricas y móviles, haría falta un libro completo (o más) para explorar totalmente este campo tan atractivo y en tan rápida expansión. Animamos al lector a profundizar más en este campo consultando las muchas referencias que hemos proporcionado a lo largo del capítulo.

Problemas y cuestiones de repaso

Capítulo 7 Cuestiones de repaso

SECCIÓN 7.1

- R1. ¿Qué quiere decir que una red inalámbrica está operando en “modo de infraestructura”? Si la red no se encuentra en modo de infraestructura, ¿en qué modo de operación se encuentra y cuál es la diferencia entre ese modo de operación y el modo de infraestructura?
- R2. ¿Cuáles son los cuatro tipos de redes inalámbricas identificados en nuestra taxonomía de la Sección 7.1? ¿Cuáles de estos tipos de redes inalámbricas ha usado usted?

SECCIÓN 7.2

- R3. ¿Cuáles son las diferencias entre los siguientes tipos de deficiencias de los canales inalámbricos: pérdida de la ruta, propagación multicamino, interferencia de otros orígenes?
- R4. A medida que un nodo móvil se va alejando de una estación base, ¿qué dos acciones puede llevar a cabo una estación base para garantizar que la probabilidad de pérdida de las tramas transmitidas no se incremente?

SECCIONES 7.3 Y 7.4

- R5. Describa el papel de las tramas baliza en 802.11.
- R6. Verdadero o falso: antes de que una estación 802.11 transmita una trama de datos, debe en primer lugar enviar una trama RTS y recibir la correspondiente trama CTS.
- R7. ¿Por qué se utilizan los reconocimientos en 802.11, pero no en una Ethernet cableada?
- R8. Verdadero o falso: Ethernet y 802.11 utilizan la misma estructura de trama.
- R9. Describa cómo funciona el umbral RTS.
- R10. Suponga que las tramas RTS y CTS de IEEE 802.11 fueran tan largas como las tramas DATA y ACK estándar. ¿Proporcionaría entonces alguna ventaja el uso de las tramas CTS y RTS? ¿Por qué?
- R11. En la Sección 7.3.4 se ha analizado la movilidad en los estándares 802.11, en la que una estación inalámbrica se desplaza desde una BSS a otra dentro de una misma subred. Cuando los puntos de acceso están interconectados mediante un switch, un punto de acceso puede necesitar enviar una trama con una dirección MAC suplantada para hacer que el conmutador reenvíe la trama adecuadamente. ¿Por qué?
- R12. ¿Cuáles son las diferencias entre un dispositivo maestro en una red Bluetooth y una estación base en una red 802.11?
- R13. ¿Qué es una supertrama en el estándar Zigbee 802.15.4?
- R14. ¿Cuál es la función del “núcleo de la red” en la arquitectura de datos celulares 3G?
- R15. ¿Cuál es la función del RNC en la arquitectura de datos celular 3G? ¿Qué función realiza el RNC en la red celular de voz?
- R16. ¿Cuál es la función de eNodeB, MME, P-GW y S-GW en la arquitectura 4G?
- R17. Enumere tres diferencias importantes entre las arquitecturas celulares 3G y 4G.

SECCIONES 7.5 Y 7.6

- R18. Si un nodo dispone de una conexión inalámbrica con Internet, ¿tiene que ser móvil ese nodo? Explique su respuesta. Suponga que un usuario con una computadora portátil va paseando por su domicilio y siempre accede a Internet a través del mismo punto de acceso. ¿Es este usuario móvil desde el punto de vista de la red? Explique su respuesta.
- R19. ¿Cuál es la diferencia entre una dirección permanente y una dirección cedida (COA)? ¿Quién asigna una dirección COA?
- R20. Considere una conexión TCP que pasa a través de IP móvil. Indique si es verdadera o falsa la siguiente afirmación: la fase de la conexión TCP entre el corresponsal y el host móvil pasa a través de la red propia del móvil, pero la fase de transferencia de datos se lleva a cabo directamente entre el corresponsal y el host móvil, sin pasar por la red propia.

SECCIÓN 7.7

- R21. ¿Cuáles son los objetivos de los registros HLR y VLR en las redes GSM? ¿Qué elementos de IP móvil son similares al HLR y al VLR?
- R22. ¿Cuál es el papel del MSC ancla en las redes GSM?

SECCIÓN 7.8

R23. ¿Qué tres técnicas pueden utilizarse para evitar que un único enlace inalámbrico degrada el rendimiento de una conexión TCP terminal a terminal de la capa de transporte?

Problemas

- P1. Considere el ejemplo de CDMA con un único emisor de la Figura 7.5. ¿Cuál sería la salida del emisor (para los 2 bits de datos mostrados) si el código CDMA del emisor fuera $(1, -1, 1, -1, 1, -1, 1)$?
 - P2. Considere el emisor 2 en la Figura 7.6. ¿Cuál es la salida del emisor hacia el canal (antes de que se sume con la señal procedente del emisor 1), $Z_{i,m}^2$?
 - P3. Suponga que el receptor de la Figura 7.6 desea recibir los datos que están siendo enviados por el emisor 2. Demuestre (mediante los cálculos necesarios) que el receptor puede efectivamente recuperar los datos del emisor 2 a partir de la señal agregada del canal utilizando el código correspondiente al emisor 2.
 - P4. Para el ejemplo de dos emisores y dos receptores, proporcione un ejemplo de dos códigos CDMA que contengan valores 1 y -1 y que no permitan a los dos receptores extraer los bits originales transmitidos por los dos emisores CDMA.
 - P5. Suponga que hay dos ISP que proporcionan acceso WiFi en una determinada cafetería, operando cada uno de esos ISP con su propio punto de acceso y disponiendo de su propio bloque de direcciones IP.
 - a. Suponga además que, por accidente, cada ISP ha configurado su punto de acceso para operar con el canal 11. ¿Dejaría completamente de funcionar el protocolo 802.11 en esta situación? Explique lo que sucede cuando dos estaciones, cada una de ellas asociada con un ISP diferente, tratan de transmitir al mismo tiempo.
 - b. Suponga ahora que uno de los puntos de acceso opera a través del canal 1 y que el otro opera a través del canal 11. ¿Cómo modifica esto sus respuestas anteriores?
 - P6. En el paso 4 del protocolo CSMA/CA, una estación que transmite con éxito una trama inicia el protocolo CSMA/CA para transmitir una segunda trama en el paso 2 en lugar de en el paso 1. ¿Qué razones pueden haber tenido en mente los diseñadores de CSMA/CA para hacer que dicha estación no transmita la segunda trama de forma inmediata (si se detecta que el canal está inactivo)?
 - P7. Suponga que configuramos una estación 802.11b para reservar siempre el canal con la secuencia RTS/CTS. Suponga también que esta estación desea de repente transmitir 1.000 bytes de datos y que todas las demás estaciones están inactivas en ese momento. Calcule el tiempo requerido para transmitir la trama y recibir el mensaje de reconocimiento en función de SIFS y DIFS, e ignorando el retardo de propagación y suponiendo que no se produce ningún error de bit.
 - P8. Considere el escenario mostrado en la Figura 7.34, en el que hay cuatro nodos inalámbricos, A, B, C y D. El radio de cobertura de los cuatro nodos se muestra mediante los óvalos sombreados; todos los nodos comparten la misma frecuencia. Cuando A transmite, solo puede ser escuchada/recibida por B; cuando B transmite, tanto A como C pueden escuchar/recibir desde B; cuando C transmite, tanto B como D pueden escuchar/recibir desde C; cuando D transmite, solo C puede escuchar/recibir desde D.
- Suponga ahora que cada nodo tiene un suministro infinito de mensajes que quiere enviar a cada uno de los otros nodos. Si el destino de un mensaje no es un vecino inmediato del nodo, entonces los mensajes deben ser reenviados. Por ejemplo, si A desea enviar a D, el mensaje debe enviarse primero a B, el cual lo envía a C y este lo envía a D. El tiempo está partitionado y el tiempo de transmisión de cada mensaje es exactamente igual a una partición de tiempo,

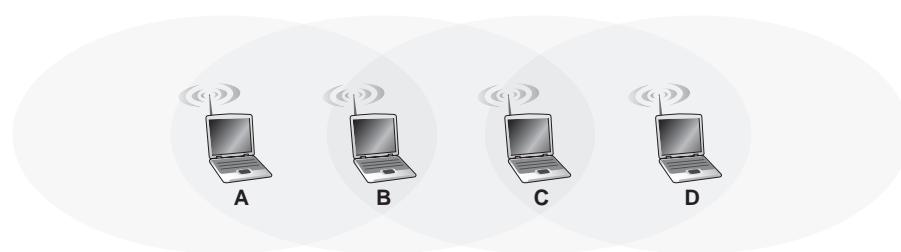


Figura 7.34 ♦ Escenario para el problema P8.

como sucede por ejemplo en el protocolo Aloha con particiones. Durante una partición de tiempo, un nodo puede hacer una de las cosas siguientes: (i) enviar un mensaje, (ii) recibir un mensaje (si se está enviando exactamente un mensaje a ese nodo), (iii) permanecer en silencio. Como siempre, si un nodo escucha dos o más transmisiones simultáneas se produce una colisión y ninguno de los mensajes transmitidos será recibido correctamente. Puede suponer aquí que no existen errores de nivel de bit y que, por tanto, si se está enviando exactamente un mensaje, éste será recibido correctamente por aquellos nodos que se encuentren dentro del radio de transmisión del emisor.

- Suponga ahora que un controlador omnisciente (es decir, un controlador que conoce el estado de todos los nodos de la red) puede ordenar a cada nodo que haga lo que el controlador omnisciente quiere, es decir, enviar un mensaje, recibir un mensaje o permanecer en silencio. Dado este controlador omnisciente, ¿cuál es la velocidad máxima a la que puede transmitirse un mensaje desde C a A, supuesto que no haya ningún otro mensaje entre ningún otro par de nodos origen/destino?
 - Suponga ahora que A envía mensajes a B y que D envía mensajes a C. ¿Cuál es la tasa máxima combinada a la que pueden fluir mensajes de datos desde A hasta B y desde D hasta C?
 - Suponga ahora que A envía mensajes a B y que C envía mensajes a D. ¿Cuál es la tasa máxima combinada a la que pueden fluir mensajes de datos desde A hasta B y desde C hasta D?
 - Suponga ahora que sustituimos los enlaces inalámbricos por enlaces cableados. Repita los apartados (a) hasta (c) en este escenario cableado.
 - Ahora suponga que volvemos al escenario inalámbrico y que para todos los mensajes de datos enviados desde el origen al destino, el destino responde con un mensaje ACK dirigido al origen (por ejemplo, como en TCP). Suponga también que cada mensaje ACK ocupa una partición. Repita los apartados (a) – (c) anteriores para este escenario.
- P9. Describa el formato de la trama Bluetooth 802.15.1. Tendrá que leer algunas referencias, además de este libro, para encontrar la información correspondiente. ¿Hay algo en el formato de trama que limite de manera inherente el número de nodos activos de una red 802.15.1 a ocho nodos? Explique su respuesta.
- P10. Considere el siguiente escenario LTE idealizado. El canal de bajada (véase la Figura 7.21) está particionado en el tiempo, entre F frecuencias. En el canal de bajada hay cuatro nodos, A, B, C y D, alcanzables desde la estación base a velocidades de 10 Mbps, 5 Mbps, 2,5 Mbps y 1 Mbps, respectivamente. Estas velocidades asumen que la estación base utiliza todas las particiones temporales disponibles en todas las F frecuencias para enviar a una única estación. La estación base tiene una cantidad infinita de datos que enviar a cada uno de los nodos y puede transmitir a cualquiera de estos cuatro nodos utilizando cualquiera de las F frecuencias durante cualquier partición de tiempo en la subtrama de bajada.
- ¿Cuál es la velocidad máxima a la que la estación base puede enviar a los nodos, suponiendo que puede enviar a cualquier nodo que elija durante cada partición de tiempo? ¿Es

- equitativa su solución? Explique su respuesta y defina qué quiere decir con el término “equitativa”.
- b. Si imponemos el requisito de que la comunicación sea equitativa, en el sentido de que cada nodo deba recibir la misma cantidad de datos durante cada intervalo de un segundo, ¿cuál es la velocidad media de transmisión de la estación de base (a todos los nodos) durante la subtrama de bajada? Explique cómo ha llegado a obtener su respuesta.
- c. Suponga que el criterio de comunicación equitativa es que cada nodo puede recibir como mucho el doble de datos que cualquier otro nodo durante la subtrama. ¿Cuál es la velocidad media de transmisión de la estación base (a todos los nodos) durante la subtrama? Explique cómo ha obtenido su respuesta.
- P11. En la Sección 7.5, una solución propuesta para permitir que los usuarios móviles mantengan sus direcciones IP a medida que se desplazan entre redes ajena era hacer que una red ajena anunciara una ruta altamente específica hacia el usuario móvil y usar la infraestructura de enrutamiento existente para propagar esta información a través de la red. Ya hemos identificado la escalabilidad como un problema. Suponga que cuando un usuario móvil se desplaza de una red a otra, la nueva red ajena anuncia una ruta específica hacia el usuario móvil y que la red ajena antigua retira su ruta. Considere cómo se propagaría la información de enrutamiento con un algoritmo de vector de distancia (particularmente para el caso de enrutamiento entre dominios, entre redes que abarquen todo el globo terráqueo).
- ¿Podrían otros routers ser capaces de enrutar datagramas inmediatamente hacia la nueva red ajena, en cuanto la red ajena comience a anunciar su ruta?
 - ¿Es posible que los diferentes routers crean que el usuario móvil se encuentra en diferentes redes ajena?
 - Analice la escala temporal sobre la que otros routers de la red terminarán por aprender la ruta hacia los usuarios móviles.
- P12. Suponga que el corresponsal de la Figura 7.23 fuera móvil. Dibuje la infraestructura adicional de la capa de red que sería necesaria para enrutar el datagrama desde el usuario móvil original hasta el (ahora móvil) corresponsal. Muestre la estructura del datagrama (o de los datagramas) entre el usuario móvil original y el (ahora móvil) corresponsal, como en la Figura 7.24.
- P13. En IP móvil, ¿qué efecto tendrá la movilidad sobre los retardos extremo a extremo de los datagramas enviados desde un origen a un destino?
- P14. Considere el ejemplo de encadenamiento analizado al final de la Sección 7.7.2. Suponga que un usuario móvil visita las redes ajena A, B y C y que un corresponsal inicia una conexión con el usuario móvil cuando este reside en la red ajena A. Enumere la secuencia de mensajes entre los agentes ajenos y entre esos agentes ajenos y el agente propio, a medida que el usuario móvil se desplaza desde la red A a la red B y a la red C. A continuación, suponga que no realizamos el encadenamiento y que hay que notificar explícitamente al corresponsal (así como al agente propio) los cambios en la dirección COA del usuario móvil. Enumere la secuencia de mensajes que habría que intercambiar en este segundo escenario.
- P15. Considere dos nodos móviles en una red ajena que dispone de un agente ajeno. ¿Es posible que los dos nodos móviles usen la misma dirección COA en IP móvil? Explique su respuesta.
- P16. En nuestro análisis de cómo el registro VLR actualizaba el HLR con información acerca de la ubicación actual del móvil, ¿cuáles son las ventajas y desventajas de proporcionar al HLR el MSRN en lugar de la dirección del VLR?

Práctica de laboratorio con Wireshark

En el sitio web del libro, www.pearsonhighered.com/cs-resources, encontrará una práctica de laboratorio con Wireshark para este capítulo en la que se capturan y analizan las tramas 802.11 intercambiadas entre una computadora portátil inalámbrica y un punto de acceso.

Deborah Estrin

Deborah Estrin es profesora de Ciencias de la Computación en Cornell Tech, en Nueva York, así como profesora de Salud Pública en el Weill Cornell Medical College. Ha sido fundadora del **Health Tech Hub** en Cornell Tech y cofundadora de la empresa sin ánimo de lucro **Open mHealth**. Se doctoró (1985) en Ciencias de la Computación en el M.I.T. y se licenció (1980) en la Universidad de California en Berkeley. Las primeras investigaciones de Estrin se centraron en el diseño de protocolos de red, incluyendo el enrutamiento de multidifusión e inter-dominios. En 2002 fundó el centro científico y tecnológico para sensores empotrados en red (CENS, Center for Embedded Networked Sensing, <http://cens.ucla.edu>) de UCLA, financiado por la NSF. El CENS puso en marcha nuevas áreas de investigación multidisciplinar en sistemas de computación, desde redes de sensores para monitorización medioambiental hasta proyectos de medida participativos para ciencia ciudadana. Actualmente está centrada en los temas de salud móvil y datos de pequeña envergadura, tratando de aprovechar la ubicuidad de los dispositivos móviles y de las interacciones digitales para mejorar la salud y la calidad de vida, como describió en una conferencia TEDMED en 2013. La profesora Estrin es miembro electo de la Academia Americana de Artes y Ciencias (2007) y de la Academia Nacional de Ingeniería (2009). Es miembro del IEEE, la ACM y la AAAS. Fue seleccionada como primera Profesora Athena de la ACM-W (2006), ha recibido el premio a la innovación Mujeres con Visión del Instituto Anita Borg (2007), ha sido incluida en el salón de la fama de WITI (2008) y ha sido nombrada *Doctor Honoris Causa* por la EPFL (2008) y la Universidad de Uppsala (2011).



**¿Podría describirnos algunos de los proyectos más excitantes en los que haya trabajado durante su carrera?
¿Cuáles fueron los principales desafíos?**

A mediados de los noventa, en USC e ISI, tuve la gran suerte de trabajar con gente como Steve Deering, Mark Handley y Van Jacobson en el diseño de protocolos de enrutamiento multidifusión (en particular, PIM). Intenté aplicar muchas de las lecciones de diseño arquitectónico de multidifusión al diseño de matrices de monitorización ecológica, que fue donde por primera vez comencé a tomarme realmente en serio las aplicaciones y la investigación multidisciplinar. Es esa innovación conjunta en los campos social y tecnológico lo que más me interesa en mi más reciente área de investigación, la salud móvil. Los desafíos en estos proyectos eran tan variados como los propios dominios de problemas, pero lo que tenían en común era la necesidad de prestar atención a si teníamos bien definido el problema, a medida que iterábamos entre diseño e implantación, prototipo y piloto. Ninguno de esos problemas podía ser resuelto analíticamente, mediante simulación o ni siquiera en experimentos de laboratorio artificiales. Todos ponían a prueba nuestra capacidad de mantener las arquitecturas claras en presencia de problemas y contextos confusos, y todos exigían una amplia colaboración.

¿Qué cambios e innovaciones cree que experimentarán en el futuro las redes inalámbricas y la movilidad?

En una edición anterior de esta entrevista dije que nunca he confiado mucho en las predicciones de futuro, pero sí que especulé que podríamos ver el final de los teléfonos tradicionales (es decir, los que no son programables y solo se utilizan para voz y mensajes de texto), a medida que los teléfonos inteligentes se fueran haciendo cada vez más potentes y se convirtieran en el principal punto de acceso a Internet para muchas personas; ahora, no muchos años después, está claro que ha sucedido así. También predijo que veríamos la proliferación continuada de módulos de instrumentación científica empotrados que permitirían a todo tipo de dispositivos comunicarse a través de la red celular a una baja velocidad de bit. Aunque eso ha ocurrido, podemos ver muchos dispositivos y la “Internet de las cosas” que utilizan WiFi integrada y otras formas de conectividad de menor potencia y alcance más corto para conectarse a concentradores locales. Por aquel entonces no pude prever la aparición de un gran mercado de productos vestibles de consumo. Para cuando se publique la siguiente edición, preveo que habrán proliferado enormemente las aplicaciones personales que aprovechen los datos de la Internet de las cosas y otras trazas digitales.

¿Cuál cree que es el futuro de las redes y de Internet?

De nuevo, pienso que es útil mirar tanto hacia atrás como hacia adelante. Anteriormente dije que los esfuerzos relativos a datos nominados y a redes definidas por software terminarían por crear una infraestructura mejor gestionable, mejor actualizable y más rica, que en términos generales representa desplazar el papel de la arquitectura hacia arriba de la pila de protocolos. En los comienzos de Internet, la arquitectura eran la capa 4 e inferiores, con aplicaciones más aisladas/monolíticas descansando sobre ella. Ahora, los datos y la analítica dominan el transporte. La adopción de SDN (que me alegra ver que se cubre en esta edición del libro) ha sobrepasado con mucho mis expectativas. Sin embargo, si miramos más arriba en la pila, nuestras aplicaciones dominantes viven cada vez más en jardines amurallados, ya se trate de aplicaciones móviles o de grandes plataformas de consumo como Facebook. A medida que se desarrollen la Ciencia de Datos y las técnicas de Datos a gran escala (*Big Data*), puede que contribuyan a extraer estas aplicaciones de sus silos, debido al valor inherente a la conexión con otras aplicaciones y plataformas.

¿Qué personas le han servido de inspiración profesionalmente?

Hay tres personas que me vienen a la mente. En primer lugar, Dave Clark, el ingrediente secreto y el héroe desconocido de la comunidad Internet. Tuve la suerte de poder verle actuar, en los primeros tiempos, como “principio organizador” del IAB y del gobierno de Internet; era el sumo sacerdote del consenso y de la ejecución de código. En segundo lugar, Scott Shenker, por su brillantez intelectual, su integridad y su persistencia. Yo intento conseguir, pero raras veces la alcanzo, su claridad a la hora de definir los problemas y las soluciones. En tercer lugar, mi hermana Judy Estrin, que tuvo la creatividad y el coraje de dedicar su carrera a llevar al mercado nuevas ideas y conceptos. Sin las Judys de este mundo, las tecnologías Internet nunca habrían conseguido transformar nuestras vidas.

¿Qué le recomendaría a los estudiantes que quieran desarrollar su carrera en el campo de la Ciencia de la Computación y las Redes?

En primer lugar, que desarrollen una sólida base en su trabajo académico, equilibrada con toda la experiencia de trabajo en el mundo real que sean capaces de adquirir. A la hora de buscar un entorno laboral, que intenten buscar oportunidades trabajando en problemas que realmente les importen y con equipos humanos inteligentes de los que puedan aprender.

Seguridad en las redes de computadoras

En la Sección 1.6 hemos descrito algunos de los ataques más dañinos y predominantes en Internet, incluyendo los ataques de software malicioso, de denegación de servicio, de husmeadores, de emmascaramiento de orígenes y borrado y modificación de mensajes. Aunque ya hemos estudiado un montón de cosas acerca de las redes de computadoras, no hemos examinado todavía cómo dotar de seguridad a las redes para defenderte de estos ataques. Equipados con nuestros conocimientos recién adquiridos sobre las redes de computadoras y los protocolos de Internet, ahora vamos a estudiar en profundidad las comunicaciones seguras y, en concreto, cómo las redes de computadoras pueden defenderse de los malos.

Presentemos a Alicia y Benito, dos personas que desean comunicarse y desean hacerlo “de forma segura”. Puesto que este es un texto sobre redes, debemos resaltar que Alicia y Benito podrían ser dos routers que desean intercambiar sus tablas de enrutamiento de forma segura, o un cliente y un servidor que desean establecer una conexión de transporte segura, o dos aplicaciones de correo electrónico que quieren intercambiar mensajes de correo seguros (casos que consideraremos más adelante en el capítulo). Alicia y Benito son entidades bien conocidas en la comunidad de la seguridad, quizás porque sus nombres son más divertidos que una entidad genérica “A” que desea comunicarse de forma segura con una entidad genérica “B”. Los asuntos amorosos ilícitos, las comunicaciones en tiempo de guerra y las transacciones de negocios son las necesidades humanas de comunicación segura habituales; prefiriendo la primera de estas necesidades a las dos últimas, nos hace felices utilizar a Alicia y Benito como nuestro emisor y nuestro receptor e imaginarlos en este primer escenario.

Hemos dicho que Alicia y Benito desean comunicarse “de forma segura”, pero, ¿qué quiere decir esto exactamente? Como veremos, la seguridad (como el amor) es algo que tiene muchos matices; es decir, la seguridad tiene muchas caras. Realmente, Alicia y Benito desean que el contenido de sus comunicaciones sea secreto y que nadie pueda conocerlo. Probablemente, también desearán estar seguros de que cuando ellos están manteniendo una comunicación, realmente estén comunicándose el uno con el otro y que si la comunicación es alterada por un tercero, esa alteración pueda ser detectada. En la primera parte de este capítulo vamos a abordar las técnicas fundamentales de criptografía que permiten cifrar las comunicaciones, autenticar al interlocutor con el que se establece la comunicación y garantizar la integridad del mensaje.

En la segunda parte del capítulo examinaremos cómo pueden utilizarse los principios fundamentales de la criptografía para crear protocolos de red seguros. De nuevo, siguiendo el método de arriba-abajo, examinaremos los protocolos seguros en cada una de las cuatro capas superiores, comenzando por la capa de aplicación. Veremos cómo dotar de seguridad al correo electrónico, cómo dotar de seguridad a una conexión TCP, cómo proporcionar seguridad a la capa de red y cómo dotar de seguridad a una red LAN inalámbrica. En la tercera parte del capítulo consideraremos la seguridad operacional, la cual se ocupa de la protección de las redes institucionales frente a los ataques. En particular, estudiaremos cómo los cortafuegos y los sistemas de detección de intrusiones pueden mejorar la seguridad de la red de una organización.

8.1 ¿Qué es la seguridad de red?

Comencemos nuestro estudio sobre la seguridad de las redes volviendo a nuestros amantes Alicia y Benito que desean comunicarse “de forma segura”. ¿Qué significa esto exactamente? En realidad, Alicia quiere que solo Benito sea capaz de comprender los mensajes que ella le envía, incluso aunque estén comunicándose a través de un medio no seguro en el que un intruso (Gertrudis, por ejemplo) pueda interceptar lo que Alicia transmite a Benito. Benito también quiere estar seguro de que el mensaje que él recibe de Alicia fue realmente enviado por Alicia, y Alicia quiere estar segura de que la persona que se está comunicando con ella es realmente Benito. Alicia y Benito también quieren estar seguros de que el contenido de sus mensajes no ha sido alterado en el camino. Además, quieren estar seguros de que siempre podrán comunicarse (es decir, que nadie les puede denegar el acceso a los recursos necesarios para comunicarse). Teniendo en cuenta estas consideraciones, podemos identificar las siguientes propiedades deseables en una **comunicación segura**.

- *Confidencialidad.* Solo el emisor y el receptor deseado deberán comprender el contenido de los mensajes transmitidos. Puesto que los curiosos pueden interceptar los mensajes, es absolutamente necesario que los mensajes sean **cifrados** de alguna manera, de modo que un mensaje interceptado no pueda ser comprendido por el que lo ha interceptado. Este aspecto de la confidencialidad es probablemente el concepto más comúnmente percibido del término *comunicación segura*. En la Sección 8.2 estudiaremos las técnicas criptográficas para el cifrado y descifrado de datos.
- *Integridad de los mensajes.* Alicia y Benito quieren estar seguros de que el contenido de sus comunicaciones no se ve alterado durante la transmisión ni maliciosamente ni por accidente. Se pueden emplear extensiones a las técnicas de suma de comprobación que hemos visto en los protocolos de enlace de datos y de transporte fiable para proporcionar integridad a los mensajes. Estudiaremos la integridad de los mensajes en la Sección 8.3.
- *Autenticación del punto terminal.* Tanto el emisor como el receptor deberán poder confirmar la identidad del otro en el proceso de comunicación (confirmar que el otro es de hecho quien dice ser). La comunicación humana frente a frente resuelve este problema fácilmente gracias al reconocimiento visual. Cuando las entidades se comunican a través de un medio en el que no es posible ver al otro, la autenticación no es tan sencilla. Por ejemplo, cuando un usuario accede a su bandeja de entrada ¿cómo verifica el servidor de correo que el usuario es la persona que dice ser? En la Sección 8.4 veremos cómo se realiza la autenticación del punto terminal.
- *Seguridad operacional.* Casi todas las organizaciones (empresas, universidades, etc.) actuales disponen de redes que están conectadas a la red pública Internet. Estas redes pueden, potencialmente, verse comprometidas. Los atacantes pueden intentar depositar gusanos en los hosts de la red, conseguir secretos corporativos, realizar un mapa de las configuraciones internas de la red y ejecutar ataques DoS. En la Sección 8.9 veremos que se emplean dispositivos operacionales, como los cortafuegos y los sistemas de detección de intrusiones, para responder a los ataques efectuados contra la red de una organización. Un cortafuegos se coloca entre la red de la orga-

nización y la red pública, controlando el acceso de paquetes procedentes de la red. Un sistema de detección de intrusiones realiza una “inspección profunda de los paquetes”, alertando a los administradores de la red cuando detecta cualquier actividad sospechosa.

Una vez establecido lo que queremos decir al hablar de la seguridad de la red, vamos a ver exactamente a qué información puede tener acceso un intruso y qué acciones puede llevar a cabo dicho intruso. La Figura 8.1 ilustra este escenario. Alicia es el emisor y desea enviar datos a Benito, que es el receptor. Para intercambiar datos de forma segura y poder cumplir los requisitos de confidencialidad, autenticación del punto terminal e integridad de los mensajes, Alicia y Benito intercambiarán mensajes de control y mensajes de datos (de forma similar a como los emisores y receptores TCP intercambian segmentos de control y segmentos de datos). Normalmente, todos o algunos de estos mensajes serán cifrados. Como hemos visto en la Sección 1.6, potencialmente un intruso puede:

- *Curiosear* (husmear y registrar los mensajes de control y de datos que se transmiten por el canal).
- *Modificar, insertar o borrar* mensajes o el contenido de los mismos.

Como veremos, a menos que se tomen las contramedidas adecuadas, estas capacidades permitirán a un intruso montar una amplia variedad de ataques contra la seguridad de la red: escuchando las comunicaciones (posiblemente robando las contraseñas y los datos), suplantando a otra entidad, pirateando una sesión activa, denegando el servicio a los usuarios legítimos de la red por sobrecarga de los recursos del sistema, etc. Puede ver un resumen de los ataques conocidos en el Centro de Coordinación CERT [CERT 2016].

Una vez que ha quedado claro el hecho de que existen amenazas reales en Internet, ¿cuáles son los equivalentes en Internet de Alicia y Benito, nuestros amigos que necesitan comunicarse de forma segura? Realmente, Benito y Alicia pueden ser personas situadas en dos sistemas terminales, por ejemplo, una Alicia real y un Benito real que desean intercambiar mensajes de correo electrónico de forma segura. Asimismo, también pueden desear participar en una transacción de comercio electrónico; por ejemplo, Benito puede tener que transferir el número de su tarjeta de crédito de forma segura a un servidor web para comprar cierto producto en línea. De forma similar, una Alicia real puede querer interactuar con su banco en línea. Los participantes que necesitan comunicaciones seguras pueden ser ellos mismos parte de la infraestructura de red. Recuerde que el sistema de nombres de dominio (DNS, véase la Sección 2.4) o los demonios de enrutamiento que intercambian información de enrutamiento (véase el Capítulo 5) requieren una comunicación segura entre ambas partes. Esto también es así en el caso de las aplicaciones de gestión de red (un tema que hemos examinado en el Capítulo 5). Un intruso que pudiera interferir de forma activa en las búsquedas DNS (como hemos visto en la Sección 2.4), los cálculos de enrutamiento [RFC 4272] o las funciones de gestión de red [RFC 3414] podría causar estragos en Internet.

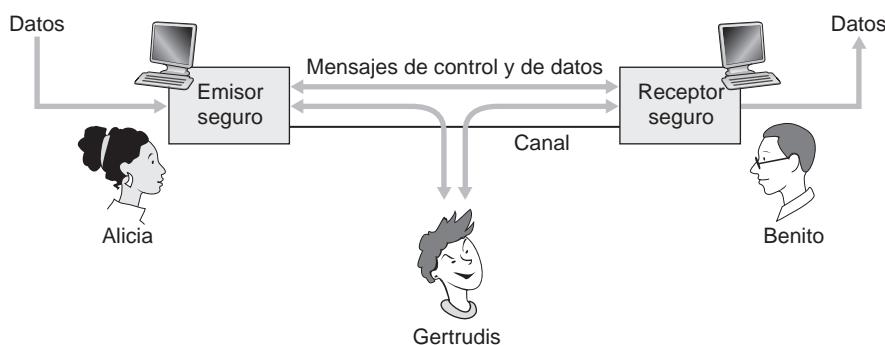


Figura 8.1 ♦ Emisor, receptor e intruso (Alicia, Benito y Gertrudis).

Una vez que hemos establecido el marco de trabajo, algunas de las definiciones más importantes y la necesidad de dotar de seguridad a la red, vamos a profundizar a continuación en la criptografía. Aunque el uso de la criptografía para proporcionar confidencialidad es evidente, veremos también que es fundamental para la autenticación del punto terminal y la verificación de la integridad de los mensajes, lo que hace de ella una piedra angular de los mecanismos de seguridad de la red.

8.2 Principios de la criptografía

Aunque la criptografía tiene una larga historia que se remonta hasta la época de Julio César, las técnicas criptográficas modernas, incluyendo muchas de las utilizadas en Internet, están basadas en los avances realizados en los últimos 30 años. El libro de Kahn, *The Codebreakers* [Kahn 1967], y el libro de Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* [Singh 1999], proporcionan una fascinante visión de la larga historia de la criptografía. Una exposición completa sobre criptografía requiere un libro completo [Kaufman 1995; Schneier 1995] y, por tanto, aquí solo vamos a abordar los aspectos esenciales, en particular aquellos que están en práctica en Internet. Debemos comentar también que aunque en esta sección vamos a centrarnos en el uso de la criptografía para conseguir confidencialidad, veremos después que las técnicas criptográficas están inextricablemente unidas a la autenticación, la integridad de los mensajes, el no repudio y otras muchas cuestiones.

Las técnicas criptográficas permiten a un emisor ocultar los datos de modo que los intrusos no puedan obtener ninguna información a partir de los datos interceptados. El receptor, por supuesto, deberá ser capaz de recuperar los datos originales a partir de los datos ocultados. La Figura 8.2 ilustra parte de la terminología más importante.

Suponga ahora que Alicia quiere enviar un mensaje a Benito. El mensaje de Alicia en su forma original (por ejemplo, "Benito, te quiero. Alicia") se conoce con el nombre de **texto en claro** o **texto plano** (*cleartext* o *plaintext*). Alicia cifra su mensaje de texto en claro utilizando un **algoritmo de cifrado** de modo que el mensaje cifrado, que se conoce con el nombre de **texto cifrado** (*ciphertext*), es ininteligible para cualquier intruso. Es interesante observar que, en muchos sistemas criptográficos modernos, incluyendo los utilizados en Internet, la propia técnica de cifrado es *conocida*, en el sentido de que es pública, está estandarizada y está disponible para todo el mundo (por ejemplo, [RFC 1321; RFC 3447; RFC 2420; NIST 2001]), incluso para los potenciales intrusos. Evidentemente, si todo el mundo conoce el método utilizado para codificar los datos,

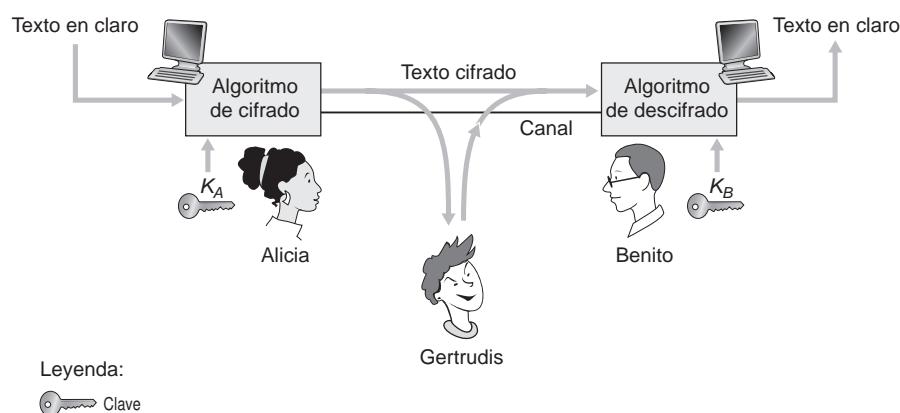


Figura 8.2 ♦ Componentes criptográficos.

entonces deberá existir algún tipo de información secreta que impida a un intruso descifrar los datos transmitidos: aquí es donde entran en acción las claves.

En la Figura 8.2 Alicia proporciona una **clave**, K_A , una cadena de números o caracteres como entrada para el algoritmo de cifrado. El algoritmo de cifrado toma la clave y el mensaje de texto en claro, m , como entrada y genera el texto cifrado como salida. La notación $K_A(m)$ hace referencia al formato en texto cifrado (cifrado utilizando la clave K_A) correspondiente al mensaje de texto en claro, m . El algoritmo de cifrado real con el que se vaya a utilizar la clave K_A resultará evidente dentro del contexto. De forma similar, Benito proporcionará una clave, K_B , al **algoritmo de descifrado**, que toma el texto cifrado y la clave de Benito como entrada y genera como salida el texto en claro original. En otras palabras, si Benito recibe un mensaje cifrado $K_A(m)$, lo descifra realizando el cálculo $K_B(K_A(m)) = m$. En los **sistemas de clave simétrica**, las claves de Alicia y de Benito son idénticas y deben mantenerse en secreto. En los **sistemas de clave pública**, se emplea una pareja de claves. Una de las claves es conocida tanto por Benito como por Alicia (de hecho es conocida por todo el mundo). La otra clave solo es conocida por Benito o por Alicia, pero no por ambos. En las dos siguientes subsecciones vamos a estudiar los sistemas de clave simétrica y de clave pública más detalladamente.

8.2.1 Criptografía de clave simétrica

Todos los algoritmos criptográficos implican sustituir una cosa por otra; por ejemplo, se toma un fragmento de texto en claro y luego se calcula y sustituye por el texto cifrado apropiado para crear el mensaje cifrado. Antes de estudiar un sistema criptográfico moderno basado en claves, entremos primero en materia estudiando un algoritmo de clave simétrica muy simple y muy antiguo atribuido a Julio César, y que se conoce con el nombre de **cifrado de César** (la palabra cifrado se emplea a menudo para designar a un método que permite encriptar los datos).

Para un texto en español, el cifrado de César funcionaría tomando cada letra del mensaje en claro y sustituyéndola por la letra que está k posiciones por detrás en el alfabeto (volviendo al principio una vez que se llega al final; es decir, haciendo que a la letra z la siga la letra a). Por ejemplo, si $k = 3$, entonces la letra a del texto en claro se convertirá en la letra d en el texto cifrado; la letra b de un texto en claro se convertirá en la letra e en el texto cifrado, y así sucesivamente. Aquí, el propio valor de k sirve como clave. Por ejemplo, el mensaje de texto en claro “benito, te quiero. alicia” se transformaría en “ehqlwr, wh txlhur. dolfld” en el texto cifrado. Aunque el texto cifrado parece una sucesión de letras sin sentido, en realidad no se tardaría mucho en romper el código si se sabe que se está utilizando el cifrado de César, ya que solo hay 25 posibles valores de clave.

Una mejora del cifrado de César sería el **cifrado monoalfabético**, que también sustituye una letra del alfabeto por otra. Sin embargo, en lugar de efectuar esas sustituciones según una patrón regular (por ejemplo, utilizando un desplazamiento k igual para todas las letras), cualquier letra puede sustituirse por cualquier otra, siempre que cada una tenga una única letra y viceversa. La regla de sustitución de la Figura 8.3 muestra una posible codificación para el texto en claro.

El mensaje de texto en claro “benito, te quiero. alicia” se convierte en “ncjsuk, uc pyscok. mgsbsm”. De nuevo, como en el caso del cifrado de César, esto parece una serie de letras sin sentido. El cifrado monoalfabético es desde luego mejor que el cifrado de César, en el sentido de que existen $26!$ (del orden de 10^{26}) posibles parejas de letras en lugar de las 25 posibles parejas que el cifrado de César proporciona. Un ataque por fuerza bruta que consistiera en probar todas las 10^{26} posibles parejas requeriría demasiado trabajo como para considerarlo una forma factible de romper el algoritmo de cifrado y decodificar el mensaje. Sin embargo, el análisis estadístico del idioma utilizado en el texto en claro, por ejemplo, saber que las letras e y a son las letras que más frecuentemente aparecen en un texto típico en español (representando aproximadamente el 14 y el 12 por ciento, respectivamente), y saber que existen determinadas combinaciones de dos y tres letras que aparecen muy frecuentemente juntas (por ejemplo, “se”, “es” “re”, “la” “al”, etc.) puede romper de forma relativamente fácil este código. Si el intruso tiene un conocimiento acerca del posible contenido del mensaje, entonces es todavía más fácil romper el código. Por ejemplo, si la

Letras texto en claro: a b c d e f g h i j k l m n o p q r s t u v w x y z
 Letras texto cifrado: m n b v c x z a s d f g h j k l p o i u y t r e w q

Figura 8.3 ♦ Cifrado monoalfabético.

intrusa Gertrudis es la mujer de Benito y sospecha que este tiene una relación sentimental con Alicia, entonces podría deducir que los nombres “benito” y “alicia” aparecerán en el texto. Si Gertrudis estuviera segura de que esos dos nombres aparecen en el texto cifrado y dispusiera de una copia del mensaje de texto cifrado de ejemplo que hemos proporcionado anteriormente, entonces podría determinar inmediatamente doce de las 26 parejas de letras, lo que implica reducir según un factor de 10^{15} menos posibilidades que habría que comprobar mediante un método de fuerza bruta. De hecho, si Gertrudis sospechara que Benito le está engañando, también esperaría encontrar algunas otras palabras seleccionadas dentro del mensaje.

A la hora de considerar lo fácil que puede ser que Gertrudis rompa el esquema de cifrado utilizado por Benito y Alicia, podemos distinguir tres escenarios distintos dependiendo de la información de la que disponga el intruso.

- *Ataque de solo texto cifrado.* En algunos casos, el intruso puede tener acceso únicamente al texto cifrado interceptado, sin disponer de información segura acerca del contenido del mensaje en claro. Ya hemos visto cómo el análisis estadístico puede ayudar a realizar un **ataque de solo texto cifrado** al esquema de cifrado.
- *Ataque de texto en claro conocido.* Anteriormente hemos visto que si Gertrudis estuviera seguro, de alguna manera, de que las palabras “benito” y “alicia” aparecen en el mensaje de texto cifrado, entonces podría haber determinado las parejas (texto en claro, texto cifrado) para las letras *a*, *l*, *i*, *c*, *b*, *e*, *n*, *t* y *o*. Gertrudis también podría haber sido lo suficientemente afortunada como para haber grabado todas las transmisiones de texto cifrado y luego encontrar en una hoja de papel la propia versión descifrada por Benito de una de las transmisiones cifradas. Cuando un intruso conoce alguna de las parejas (texto en claro, texto cifrado), decimos que se trata de un **ataque de texto en claro conocido** al esquema de cifrado.
- *Ataque de texto en claro seleccionado.* En un **ataque de texto en claro seleccionado**, el intruso tiene la posibilidad de elegir el mensaje de texto en claro y obtener su correspondiente texto cifrado. Para los algoritmos de cifrado simples que hemos visto ahora, si Gertrudis pudiera hacer que Alicia enviara el mensaje “Es extraño mojar queso en la cerveza o probar whisky de garrafa” podría romper completamente el esquema de cifrado. Pronto veremos que, para técnicas de cifrado más sofisticadas, un ataque de texto en claro seleccionado no implica necesariamente que la técnica de cifrado pueda ser rota.

Hace quinientos años se inventó una técnica denominada **cifrado polialfabético**, que permitía mejorar el cifrado monoalfabético. La idea subyacente al cifrado polialfabético es utilizar varios cifrados monoalfabéticos, utilizando un cifrado monoalfabético específico para codificar cada letra situada en una posición específica dentro del mensaje de texto en claro. De ese modo, una misma letra que aparezca en diferentes posiciones dentro del mensaje en claro se podría codificar de forma distinta cada vez. En la Figura 8.4 se muestra un ejemplo de un esquema de cifrado polialfabético. Está compuesto por dos cifrados de César (con $k = 5$ y $k = 19$), mostrados en sendas filas. Podríamos decidir utilizar estos dos cifrados de César, C_1 y C_2 , según el patrón repetitivo C_1, C_1, C_2, C_1, C_2 . De este modo, la primera y la segunda letras del texto en claro se codificarían utilizando C_1 , la tercera aplicando C_2 , la cuarta utilizando C_1 y la quinta utilizando C_2 . A continuación el patrón se repite, lo que haría que la sexta letra se codificara utilizando C_1 , la séptima con C_1 , y así sucesivamente. Así, el mensaje de texto en claro “benito, te quiero.” tendría el equivalente de texto cifrado “g jgnmt, yx vnnjkt.” Observe que la primera *e* del mensaje de texto en claro se cifra utilizando

Letras texto en claro: a b c d e f g h i j k l m n o p q r s t u v w x y z
 $C_1(k=5)$: f g h i j k l m n o p q r s t u v w x y z a b c d e
 $C_2(k=19)$: t u v w x y z a b c d e f g h i j k l m n o p q r s

Figura 8.4 ♦ Cifrado polialfabético utilizando dos cifrados de César.

C_1 , mientras que la segunda e se cifra utilizando C_2 . En este ejemplo, la “clave” de cifrado y de descifrado es el propio conocimiento de los dos cifrados de César ($k = 5, k = 19$) y del patrón C_1, C_1, C_2, C_1, C_2 .

Cifrados de bloque

Avancemos ahora hacia tiempos más modernos y veamos cómo se lleva a cabo hoy día el cifrado de clave simétrica. Existen dos clases generales de técnicas de cifrado simétrico: **cifrados de flujo** y **cifrados de bloque**. Examinaremos brevemente los cifrados de flujo en la Sección 8.7 cuando investiguemos la seguridad en las redes LAN inalámbricas. En esta sección, nos centraremos en los cifrados de bloque que se emplean en muchos protocolos seguros de Internet, incluyendo PGP (para correo electrónico seguro), SSL (para dotar de seguridad a las conexiones TCP) e IPsec (para dotar de seguridad al transporte de la capa de red).

En un cifrado de bloque, el mensaje que hay que cifrar se procesa en bloques de k bits. Por ejemplo, si $k = 64$, entonces el mensaje se descompone en bloques de 64 bits y cada bloque se cifra de forma independiente. Para codificar un bloque, el sistema de cifrado asigna una correspondencia uno-a-uno, con el fin de asignar el bloque de k bits de texto en claro a un bloque de k bits de texto cifrado. Veamos un ejemplo. Suponga que $k = 3$, de modo que el cifrado de bloque asigna a cada entrada de 3 bits (texto en claro) una salida de 3 bits (texto cifrado). En la Tabla 8.1 se proporciona una posible de estas correspondencias. Observe que se trata de una aplicación uno-a-uno, es decir, existe una salida diferente para cada entrada. Este cifrado de bloque descompone el mensaje en bloques de 3 bits y cifra cada bloque de acuerdo con la correspondencia anterior. Puede verificar que si se cifra el mensaje 010110001111 se obtiene como resultado 101000111001.

Continuando con este ejemplo de bloque de 3 bits, observe que la correspondencia de la Tabla 8.1 es simplemente una asignación de entre las muchas posibles. ¿Cuántas posibles correspondencias existen? Para responder a esta pregunta simplemente observe que una correspondencia no es otra cosa que una permutación de todas las posibles entradas. Existen $2^3 (= 8)$ posibles entradas (enumeradas bajo las columnas etiquetadas como Entrada). Estas ocho entradas pueden permutarse en $8! = 40.320$ formas distintas. Dado que cada una de estas permutaciones especifica una correspondencia, habrá 40.320 posibles correspondencias. Podemos interpretar cada una de estas correspondencias como una clave: si Alicia y Benito conocen simultáneamente la correspondencia (la clave), podrán cifrar y descifrar los mensajes que se intercambien.

El ataque por fuerza bruta a este sistema de cifrado consistiría en tratar de descifrar un texto utilizando todas las correspondencias posibles. Puesto que solo existen 40.320 correspondencias

Entrada	Salida	Entrada	Salida
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

Tabla 8.1 ♦ Un cifrado de bloque específico de 3 bits.

(cuando $k = 3$), esto puede llevarse a cabo rápidamente con cualquier PC de escritorio. Para evitar los ataques por fuerza bruta, los sistemas de cifrado de bloque normalmente utilizan bloques de mucho mayor tamaño, compuestos de $k = 64$ bits o incluso mayores. Observe que el número de correspondencias posibles para un cifrado cualquiera de bloques de k bits es de $2^k!$, que es un valor astronómico incluso para valores moderados de k (como por ejemplo $k = 64$).

Aunque los cifrados de bloque de tabla completa, como el que acabamos de describir, pueden producir esquemas de cifrado de clave simétrica robustos, incluso con valores moderados de k , lamentablemente resultan difíciles de implementar. Para $k = 64$ y una correspondencia determinada, Alicia y Benito necesitarían mantener una tabla con 2^{64} valores de entrada, lo que es una tarea imposible. Además, si Alicia y Benito quisieran cambiar de clave, ambos tendrían que volver a generar la tabla. Por tanto, la utilización de un cifrado de bloque de tabla completa que proporcione correspondencias predeterminadas entre todas las entradas y las salidas (como en el ejemplo anterior) resulta simplemente impracticable.

En lugar de ello, los sistemas de cifrado de bloque suelen utilizar funciones que simulan la generación de tablas permutadas aleatoriamente. En la Figura 8.5 se muestra un ejemplo (adaptado de [Kaufman 1995]) de una función de este tipo para $k = 64$. La función descompone en primer lugar el bloque de 64 bits en ocho fragmentos, estando cada fragmento compuesto por 8 bits. Cada fragmento de 8 bits se procesa mediante una tabla de 8 por 8 bits, que tiene un tamaño manejable. Por ejemplo, el primer fragmento se procesa mediante la tabla designada como T_1 . A continuación, los 8 fragmentos de salida se recomponen en un bloque de 64 bits. Las posiciones de los 64 bits del bloque a continuación se aleatorizan (permutan) para generar una salida de 64 bits. Esta salida se realimenta hacia la entrada de 64 bits, donde comienza un nuevo ciclo. Después de n ciclos como este, la función proporciona un bloque de 64 bits de texto cifrado. El propósito de los distintos ciclos es hacer que cada bit de entrada afecte a la mayoría (si no a todos) de los bits de salida finales (si solo se utilizara un ciclo, cada bit de entrada dado solo afectaría a 8 de los 64 bits de salida). La clave para este algoritmo de cifrado de bloque estaría compuesta por las ocho tablas de permutación (asumiendo que la función de aleatorización sea de dominio público).

Hoy día existen varios sistemas de cifrado de bloque populares, incluyendo DES (*Data Encryption Standard*, Estándar de cifrado de datos), 3DES y AES (*Advanced Encryption Standard*, Estándar avanzado de cifrado). Cada uno de estos estándares utiliza funciones en lugar de tablas predeterminadas, según las ideas expuestas en la Figura 8.5 (aunque son más complicadas y específicas de cada sistema de cifrado). Cada uno de estos algoritmos utiliza también una cadena

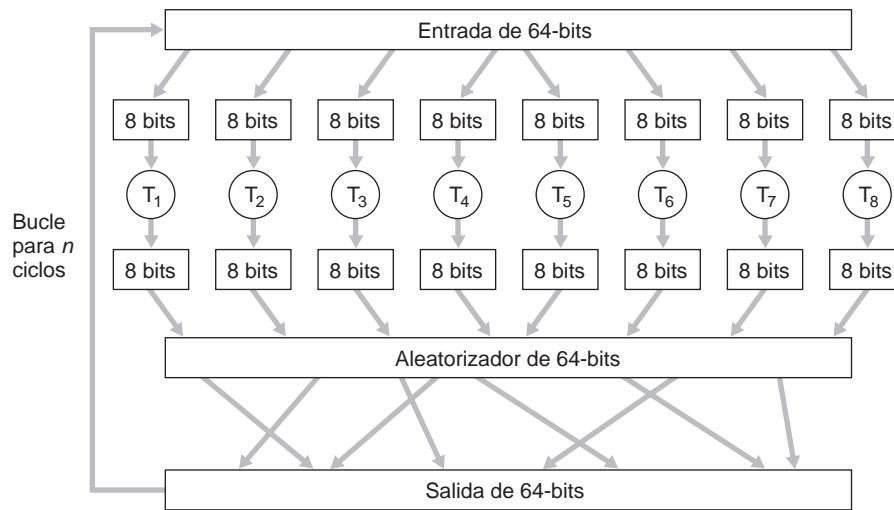


Figura 8.5 ♦ Ejemplo de un cifrado de bloque.

de bits como clave. Por ejemplo, DES emplea bloques de 64 bits con una clave de 56 bits. AES utiliza bloques de 128 bits y puede operar con claves de 128, 192 y 256 bits de longitud. La clave de un algoritmo determina las “mini-tablas” específicas de asignación y permutación dentro del algoritmo. El ataque por fuerza bruta a cada uno de estos sistemas de cifrado consistiría en ir aplicando sucesivamente todas las claves, usando el algoritmo de descifrado de cada una. Observe que con una longitud de clave igual a n , existen 2^n claves posibles. NIST [NIST 2001] estima que una máquina que pudiera romper el algoritmo DES de 56 bits en un segundo (es decir, que pudiera probar en un segundo las 2^{56} claves) necesitaría aproximadamente 149 trillones de años para romper una clave AES de 128 bits.

Encadenamiento de bloques cifrados

En las aplicaciones de redes de computadoras, normalmente es necesario cifrar mensajes de gran tamaño (o largos flujos de datos). Si aplicamos un cifrado de bloques como el descrito, descomponiendo simplemente el mensaje en bloques de k bits y cifrando independientemente cada bloque, aparece un problema bastante sutil pero de gran importancia. Para ver en qué consiste, observe que dos o más de los bloques del texto en claro podrían ser idénticos. Por ejemplo, el texto en claro en dos o más bloques podría ser “HTTP/1.1”. Para estos bloques idénticos, el cifrado de bloque produciría, por supuesto, el mismo texto cifrado. De ese modo, un atacante podría posiblemente adivinar el texto en claro cuando viera bloques de texto cifrado idénticos y podría incluso ser capaz de descifrar el mensaje completo identificando bloques de texto cifrado idénticos y utilizando el conocimiento acerca de la estructura de protocolos subyacente [Kaufman 1995].

Para resolver este problema, podemos introducir cierta aleatoriedad en el texto cifrado, de modo que idénticos bloques de texto en claro produzcan bloques de texto cifrado diferentes. Para explicar esta idea, sea $m(i)$ el i -ésimo bloque de texto en claro, sea $c(i)$ el i -ésimo bloque de texto cifrado y sea $a \oplus b$ la operación OR-exclusiva (XOR) de dos cadenas de bits, a y b . (Recuerde que $0 \oplus 0 = 1 \oplus 1 = 0$ y que $0 \oplus 1 = 1 \oplus 0 = 1$, y que la operación XOR de dos cadenas de bits se realiza bit por bit. Así, por ejemplo, $10101010 \oplus 11110000 = 01011010$.) Asimismo, designaremos mediante K_S al algoritmo de encriptación de bloque cifrado con clave S . La idea básica es la siguiente: el emisor genera un número aleatorio de k bits $r(i)$ para el i -ésimo bloque y calcula $c(i) = K_S(m(i) \oplus r(i))$. Observe que se selecciona un nuevo número aleatorio de k bits para cada bloque. El emisor envía entonces $c(1), r(1), c(2), r(2), c(3), r(3)$, etc. Puesto que el receptor recibe $c(i)$ y $r(i)$ puede recuperar cada bloque del texto en claro calculando $m(i) = K_S^{-1}(c(i)) \oplus r(i)$. Es importante observar que, aunque $r(i)$ se envíe sin cifrar y por tanto podría ser husmeado por Gertrudis, esta no podrá obtener el texto en claro $m(i)$, ya que no conoce la clave K_S . Observe también que si dos bloques de texto en claro $m(i)$ y $m(j)$ son iguales, los correspondientes bloques en texto cifrado $c(i)$ y $c(j)$ serán diferentes (siempre y cuando los números aleatorios $r(i)$ y $r(j)$ sean distintos, lo que ocurre con una muy alta probabilidad).

Por ejemplo, considere el sistema de cifrado de bloques de 3 bits de la Tabla 8.1. Suponga que el texto en claro es 010010010. Si Alicia cifra esta secuencia directamente sin incluir ninguna aleatoriedad, el texto cifrado resultante será 101101101. Si Gertrudis captura el texto cifrado, dado que cada uno de los tres bloques de cifrado es igual, podrá suponer correctamente que cada uno de los tres bloques de texto en claro son también coincidentes. Ahora suponga que en lugar de ello Alicia genera los bloques aleatorios $r(1) = 001$, $r(2) = 111$ y $r(3) = 100$ y aplica la técnica anteriormente explicada para generar el texto cifrado $c(1) = 100$, $c(2) = 010$ y $c(3) = 000$. Observe que ahora los tres bloques de texto cifrado son distintos incluso aunque los bloques de texto en claro son iguales. Alicia envía entonces $c(1), r(1), c(2) y r(2)$. El lector puede verificar que Benito podrá obtener el texto en claro original utilizando la clave compartida K_S .

Algún lector más astuto habrá deducido que el introducir la aleatoriedad resuelve un problema, pero crea otro: en concreto, Alicia tiene que transmitir el doble de bits que antes. De hecho, por cada bit de cifrado, Alicia debe ahora enviar también un bit aleatorio, duplicando así el ancho de banda requerido. Si lo que queremos es estar en misa y repicando a la vez, los sistemas de cifrado de bloque

suelen utilizar una técnica denominada **encadenamiento de bloques cifrados (CBC, Cipher Block Chaining)**. La idea básica consiste en enviar solo *un valor aleatorio junto con el primer mensaje, y hacer que el emisor y el receptor utilicen los bloques codificados calculados, en lugar de los subsiguientes números aleatorios*. Específicamente, CBC opera como sigue:

1. Antes de cifrar el mensaje (o el flujo de datos), el emisor genera una cadena aleatoria de k bits, denominada **vector de inicialización (IV, Initialization Vector)**. Denotaremos a este vector de inicialización mediante $c(0)$. El emisor envía el vector IV al receptor *sin cifrar*.
2. Para el primer bloque, el emisor calcula $m(1) \oplus c(0)$, es decir, calcula la operación OR exclusiva del primer bloque de texto en claro con IV. A continuación, introduce el resultado en el algoritmo de cifrado de bloque, para obtener el correspondiente bloque de texto cifrado; es decir, $c(1) = K_S(m(1) \oplus c(0))$. El emisor envía después el bloque cifrado $c(1)$ al receptor.
3. Para el i -ésimo bloque, el emisor genera el i -ésimo bloque de texto cifrado utilizando la fórmula $c(i) = K_S(m(i) \oplus c(i - 1))$.

Examinemos ahora algunas de las consecuencias de este método. En primer lugar, el receptor continuará pudiendo recuperar el mensaje original. De hecho, cuando el receptor reciba $c(i)$, descifrárá el mensaje con K_S para obtener $s(i) = m(i) \oplus c(i - 1)$; puesto que el receptor también conoce $c(i - 1)$, puede entonces obtener el bloque de texto en claro a partir de la fórmula $m(i) = s(i) \oplus c(i - 1)$. En segundo lugar, incluso si dos bloques de texto en claro son idénticos, los textos cifrados correspondientes serán (casi siempre) diferentes. En tercer lugar, aunque el emisor envíe el vector IV sin cifrar, ningún intruso podrá descifrar los bloques de texto cifrado dado que no conocen la clave secreta, S . Por último, el emisor solo envía un bloque de sobrecarga (el vector IV), con lo que el uso de ancho de banda solo se incrementa de una forma prácticamente despreciable, asumiendo que estemos utilizando mensajes de gran longitud (compuestos de centenares de bloques).

Como ejemplo, determinemos ahora el texto cifrado para el sistema de cifrado de bloque de 3 bits de la Tabla 8.1, utilizando como texto en claro 010010010 y como vector IV = $c(0) = 001$. En primer lugar, el emisor utiliza el vector IV para calcular $c(1) = K_S(m(1) \oplus c(0)) = 100$ y a continuación calcula $c(2) = K_S(m(2) \oplus c(1)) = K_S(010 \oplus 100) = 000$ y $c(3) = K_S(m(3) \oplus c(2)) = K_S(010 \oplus 000) = 101$. El lector puede verificar que el receptor, conociendo el vector IV y la clave K_S , puede recuperar el texto en claro original.

La técnica CBC tiene una importante consecuencia a la hora de diseñar protocolos de red seguros: necesitaremos proporcionar un mecanismo dentro del protocolo para transferir el vector IV desde el emisor al receptor. Posteriormente en el capítulo veremos cómo se hace esto en diversos protocolos.

8.2.2 Cifrado de clave pública

Durante más de 2.000 años (desde la época del cifrado de César hasta la década de 1970), la comunicación cifrada requería que los dos interlocutores que se estaban comunicando compartieran una clave secreta: la clave simétrica utilizada para el cifrado y el descifrado. Una dificultad con esta técnica es que ambas partes deben acordar de alguna manera cuál es esa clave compartida, ¡pero el hacer eso requiere que se comuniquen (presumiblemente de forma segura)! Quizá ambas partes podrían primero reunirse y acordar en persona cuál es esa clave (por ejemplo, dos de los centuriones de César podrían reunirse en una terma romana) y en lo sucesivo comunicarse mediante un método de cifrado. Sin embargo, en un mundo conectado en red, los interlocutores que se están comunicando pueden no llegar a encontrarse nunca de forma física, y puede que incluso no lleguen a conversar excepto a través de la red. ¿Es posible que dos partes se comuniquen de forma cifrada sin conocer de antemano una clave secreta compartida? En 1976, Diffie y Hellman [Diffie 1976] inventaron un algoritmo (que ahora se conoce como algoritmo de intercambio de claves de Diffie-Hellman) para hacer precisamente eso; se trata de un enfoque radicalmente distinto y maravillosamente elegante para las comunicaciones seguras y que ha conducido al desarrollo de los sistemas de cifrado actua-

les de clave pública. Como veremos en breve, los sistemas criptográficos de clave pública disfrutan también de diversas propiedades muy atractivas que los hacen útiles no solo para el cifrado, sino también para la autenticación y las firmas digitales. Es interesante observar que recientemente ha salido a la luz que unas ideas similares a las de [Diffie 1976] y [RSA 1978] habían sido desarrolladas independientemente a principios de la década de 1970 en una serie de informes secretos elaborados por investigadores del Grupo de Seguridad de Electrónica y Comunicaciones en el Reino Unido [Ellis 1987]. Como a menudo suele suceder, las grandes ideas pueden surgir de forma independiente en muchos lugares distintos; afortunadamente, los avances relativos a los sistemas de clave pública no solo tuvieron lugar en privado, sino también a la vista del público.

Conceptualmente, la utilización de un sistema de criptografía de clave pública es muy simple. Suponga que Alicia quiere comunicarse con Benito. Como se muestra en la Figura 8.6, en lugar de que Benito y Alicia comparten una única clave secreta (como es el caso en los sistemas de clave simétrica), Benito (el receptor de los mensajes de Alicia) dispone en su lugar de dos claves: una **clave pública** que está disponible para *todo el mundo* (incluyendo a Gertrudis la intrusa) y una **clave privada** que solo Benito conoce. Utilizaremos la notación K_B^+ y K_B^- para hacer referencia a las claves pública y privada de Benito, respectivamente. Para poder comunicarse con Benito, Alicia consulta primero la clave pública de Benito y luego cifra su mensaje, m , destinado a Benito utilizando esa clave pública de Benito y un algoritmo de cifrado conocido (por ejemplo, un algoritmo estandarizado); es decir, Alicia calcula $K_B^+(m)$. Benito recibe el mensaje cifrado de Alicia y utiliza su clave privada y un algoritmo de descifrado conocido (por ejemplo, un algoritmo estandarizado) para descifrar el mensaje cifrado de Alicia. Es decir, Benito calcula $K_B^-(K_B^+(m))$. Como veremos más adelante, existen técnicas y algoritmos de cifrado/descifrado para seleccionar claves públicas y privadas tales que $K_B^-(K_B^+(m)) = m$; es decir, tales que al aplicar la clave pública de Benito, K_B^+ , a un mensaje, m (para obtener $K_B^+(m)$), y aplicar luego la clave privada de Benito, K_B^- , a la versión cifrada de m (es decir, calcular $K_B^-(K_B^+(m))$) se vuelve a obtener m . Es un resultado realmente maravilloso. De esta manera, Alicia puede utilizar la clave de Benito que está públicamente disponible con el fin de enviar un mensaje secreto a Benito, sin que ninguno de los dos tenga que distribuir ninguna clave secreta. Como veremos enseguida, podemos intercambiar el papel de la clave pública y la clave privada y obtener el mismo resultado; es decir, $K_B^-(K_B^+(m)) = K_B^+(K_B^-(m)) = m$.

La utilización de la criptografía de clave pública es por tanto conceptualmente muy simple. Pero puede que al lector le surjan inmediatamente dos preguntas. Un primer posible problema es que, aunque un intruso que intercepte el mensaje de cifrado de Alicia solo obtendrá datos sin sentido, ese intruso conoce tanto la clave (la clave pública de Benito, que está disponible para que todo el mundo la vea), como el algoritmo que Alicia ha utilizado para el cifrado. Gertrudis podría entonces montar un ataque de texto claro conocido, utilizando ese algoritmo de cifrado

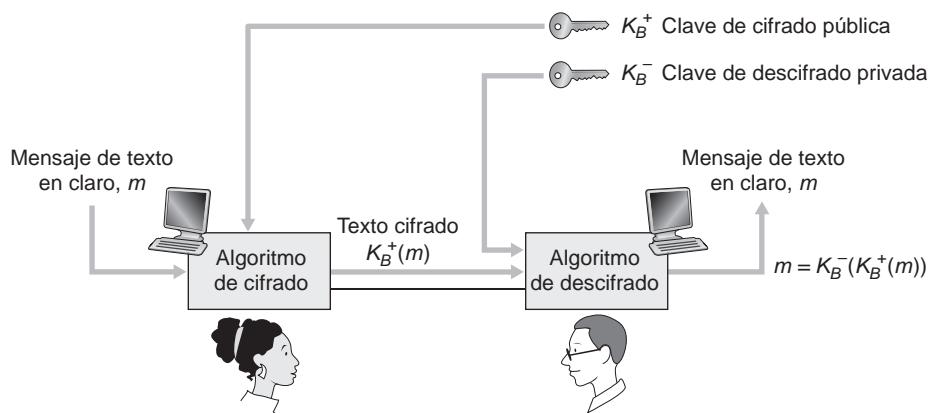


Figura 8.6 ♦ Criptografía de clave pública.

estandarizado y la clave de cifrado de Benito, públicamente disponible, para codificar cualquier mensaje que deseé. Gertrudis también podría intentar, por ejemplo, codificar mensajes, o partes de mensajes, que piense que Alicia podría enviar. Obviamente, para que la criptografía de clave pública pueda funcionar, la selección de claves y el cifrado/descifrado deben hacerse de forma tal que sea imposible (o al menos lo suficientemente difícil como para ser prácticamente imposible) para un intruso determinar la clave privada de Benito o descifrar o adivinar de alguna otra manera el mensaje que Alicia le ha enviado a Benito. El segundo problema potencial es que dado que la clave de cifrado de Benito es pública, cualquiera puede enviar un mensaje cifrado a Benito, incluyendo Alicia o alguien que se haga pasar por Alicia. En el caso de una única clave secreta compartida, el hecho de que emisor conozca la clave secreta identifica implícitamente al emisor ante el receptor. Sin embargo, en el caso de la criptografía de clave pública, este ya no será el caso puesto que nadie puede enviar un mensaje cifrado a Benito utilizando la clave públicamente disponible del mismo. Es necesaria una firma digital, tema que estudiaremos en la Sección 8.3, para vincular a un emisor con un mensaje.

RSA

Aunque pueden existir muchos algoritmos que se correspondan con la descripción realizada, el **algoritmo RSA** (llamado así por sus inventores, Ron Rivest, Adi Shamir y Leonard Adleman) se ha convertido casi en sinónimo de la criptografía de clave pública. En primer lugar vamos a ver cómo funciona RSA y luego examinaremos por qué funciona.

RSA hace un extenso uso de las operaciones aritméticas módulo n . Por ello, vamos a repasar brevemente la aritmética modular. Recuerde que $x \bmod n$ simplemente indica el resto de dividir x entre n ; así, por ejemplo, $19 \bmod 5 = 4$. En la aritmética modular se realizan las operaciones usuales de suma, multiplicación y exponentiación. Sin embargo, el resultado de cada operación se sustituye por el resto entero que se obtiene al dividir el resultado entre n . La realización de las operaciones de suma y multiplicación en aritmética modular se facilita teniendo en cuenta las siguientes fórmulas de utilidad:

$$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$

$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

Se deduce de la tercera fórmula que $(a \bmod n)^d \bmod n = a^d \bmod n$, que es una identidad que, como pronto veremos, resulta muy útil.

Supongamos ahora que Alicia desea enviar a Benito un mensaje cifrado con RSA, como se muestra en la Figura 8.6. En esta exposición sobre RSA hay que tener siempre en mente que un mensaje no es nada más que un patrón de bits y que cada patrón de bits puede representarse de manera única mediante un número entero (junto con la longitud del patrón de bits). Por ejemplo, suponga un mensaje igual al patrón de bits 1001; este mensaje se puede representar mediante el entero decimal 9. Por tanto, cifrar un mensaje con RSA es equivalente a cifrar el número entero único que representa a dicho mensaje.

En RSA existen dos componentes interrelacionados:

- La elección de las claves pública y privada.
- El algoritmo de cifrado y descifrado.

Para generar las claves RSA pública y privada, Benito lleva a cabo los pasos siguientes:

1. Elige dos números primos grandes, p y q . ¿Cómo de grandes tienen que ser p y q ? Cuanto más grandes sean estos valores, más difícil será romper el algoritmo RSA, pero también se tardará más en realizar la codificación y la decodificación. RSA Laboratories recomienda que

el producto de p por q sea del orden de 1.024 bits. Si desea obtener información acerca de cómo determinar números primos grandes, consulte [Caldwell 2012].

2. Calcula $n = pq$ y $z = (p - 1)(q - 1)$.
3. Elige un número, e , menor que n , que no tiene ningún factor común (distinto de 1) con z . (En este caso, se dice que e y z son números primos relativos.) Se emplea la letra e porque este valor se utilizará en el cifrado o encriptación.
4. Determina un número, d , tal que $ed - 1$ es divisible de forma exacta por z (es decir, su resto es igual a cero). La letra d se emplea porque este valor se utilizará en el descifrado. Dicho de otra forma, dado e , seleccionamos d tal que

$$ed \bmod z = 1$$

5. La clave pública que Benito pone a disposición de todo el mundo, K_B^+ , es la pareja de números (n, e) ; su clave privada, K_B^- , es la pareja de números (n, d) .

El cifrado por parte de Alicia y el descifrado que lleva a cabo Benito se realizan como sigue:

- Suponga que Alicia desea enviar a Benito un patrón de bits representado por el número entero m (con $m < n$). Para realizar la codificación, Alicia lleva a cabo la operación de exponenciación m^e , y luego calcula el resto entero que se obtiene al dividir m^e entre n . En otras palabras, el valor cifrado, c , del mensaje de texto en claro de Alicia, m , es

$$c = m^e \bmod n$$

El patrón de bits correspondiente a este texto cifrado c se envía a Benito.

- Para descifrar el mensaje de texto cifrado recibido, c , Benito hace el cálculo siguiente:

$$m = c^d \bmod n$$

que requiere el uso de su clave privada (n, d) .

Veamos un ejemplo simple de RSA. Suponga que Benito elige $p = 5$ y $q = 7$. (Evidentemente, estos valores son demasiado pequeños como para ser seguros.) Entonces, $n = 35$ y $z = 24$. Benito selecciona $e = 5$, ya que 5 y 24 no tienen factores comunes. Por último, Benito elige $d = 29$, dado que $5 \cdot 29 - 1$ (es decir, $ed - 1$) es divisible de forma exacta por 24. Benito hace públicos los dos valores, $n = 35$ y $e = 5$, y mantiene en secreto el valor $d = 29$. Observando estos dos valores públicos, supongamos ahora que Alicia quiere enviar a Benito las letras l, o, v y e . Interpretando cada letra como un número comprendido entre 1 y 26 (correspondiéndose la a con el 1 y la z con el 26), Alicia y Benito llevan a cabo las operaciones de cifrado y descifrado mostradas en las Tablas 8.2 y 8.3, respectivamente. Observe que, en este ejemplo, consideramos cada una de las cuatro letras como un mensaje distinto. Un ejemplo más realista sería convertir las cuatro letras en sus representaciones ASCII de 8 bits y luego cifrar el entero correspondiente al patrón de 32 bits resultante. (Tal ejemplo más realista genera números que son demasiado grandes como para imprimirlas aquí.)

Dado que el ejemplo simple de las Tablas 8.2 y 8.3 ha dado lugar a números extremadamente grandes y, puesto que como hemos visto anteriormente, p y q tendrán ambos longitudes de centenares de bits, esto nos lleva a plantearnos varias cuestiones prácticas relativas a RSA: ¿cómo se seleccionan los números primos grandes? ¿Cómo se eligen después e y d ? ¿Cómo se lleva a cabo la exponenciación con números grandes? Un análisis de estas importantes cuestiones queda fuera del alcance de este texto, pero puede consultar [Kaufman 1995] y las referencias que incluye para conocer más detalles.

Claves de sesión

Tenemos que resaltar aquí que la exponenciación requerida por RSA es un proceso que consume bastante tiempo. Por el contrario, DES es al menos 100 veces más rápido en software y entre 1.000 y

Letras texto en claro	m : representación numérica	m^e	Texto cifrado $c = m^e \text{ mod } n$
I	12	248832	17
O	15	759375	15
V	22	5153632	22
E	5	3125	10

Tabla 8.2 ♦ Cifrado RSA realizado por Alicia, $e = 5$, $n = 35$.

Texto cifrado c	c^d	$m = c^d \bmod n$	Letras texto en claro
17	4819685721067509150915091411825223071697	12	l
15	127834039403948858939111232757568359375	15	o
22	851643319086537701956194499721106030592	22	v
10	1000	5	e

Tabla 8.3 ♦ Descifrado RSA realizado por Benito, $d = 29$, $n = 35$.

10.000 veces más rápido en hardware [RSA Fast 2012]. Como resultado, a menudo RSA se utiliza en la práctica en combinación con algún otro mecanismo criptográfico de clave simétrica. Por ejemplo, si Alicia desea enviar a Benito una gran cantidad de datos cifrados podría hacer lo siguiente: primero elegiría una clave para codificar los propios datos; esta clave se conoce como **clave de sesión** y se designa mediante K_s . Alicia tiene que comunicar a Benito la clave de sesión, ya que se trata de la clave simétrica compartida que ambos utilizarán con un sistema de cifrado de clave simétrica (por ejemplo, con DES o AES). Alicia cifra la clave de sesión utilizando la clave pública de Benito, es decir, calcula $c = (K_s)^e \text{ mod } n$. Benito recibe la clave de sesión cifrada mediante RSA, c , y la descifra para obtener la clave de sesión, K_s . De este modo, ahora Benito conoce la clave de sesión que Alicia empleará para transferir los datos cifrados.

¿Por qué funciona RSA?

El cifrado/descifrado de RSA parece algo un poco mágico. ¿Por qué después de aplicar el algoritmo de cifrado y luego el algoritmo de descifrado se recupera el mensaje original? Para comprender por qué funciona RSA, volvamos a fijarnos en la ecuación $n = pq$, donde p y q son los números primos grandes utilizados por el algoritmo RSA.

Recuerde que, con el cifrado RSA, cada mensaje (representado únicamente mediante un entero), m , se eleva a la potencia e utilizando aritmética de módulo n , es decir,

$$c = m^e \bmod n$$

El descifrado se hace elevando este valor a la potencia d , utilizando de nuevo aritmética de módulo n . El resultado de un paso de cifrado seguido de un paso de descifrado será por tanto $(m^e \text{ mod } n)^d \text{ mod } n$. Veamos ahora qué podemos decir acerca de este valor. Como hemos dicho anteriormente, una propiedad importante de la aritmética modular es que $(a \text{ mod } n)^d \text{ mod } n = a^d \text{ mod } n$ para cualesquiera valores a , n y d . Luego utilizando $a = m^e$ en esta propiedad, tenemos

$$(m^e \bmod n)^d \bmod n \equiv m^{ed} \bmod n$$

Por tanto, queda demostrar que $m^{ed} \bmod n = m$. Aunque estamos intentando eliminar parte de la magia acerca de por qué funciona RSA, para demostrar esta igualdad necesitamos utilizar aquí un resultado que también parece un poco mágico extraído de la teoría de números. Específicamente, necesitamos un resultado que establece que si p y q son primos, y si $n = pq$ y si $z = (p - 1)(q - 1)$, entonces $x^y \bmod n$ es igual a $x^{(y \bmod z)} \bmod n$ [Kaufman 1995]. Aplicando este resultado con $x = m$ e $y = ed$ tenemos

$$m^{ed} \bmod n = m^{(ed \bmod z)} \bmod n$$

Pero recuerde que hemos seleccionado e y d tal que $ed \bmod z = 1$. Esto nos da

$$m^{ed} \bmod n = m^1 \bmod n = m$$

que es exactamente el resultado que estábamos buscando. Elevando en primer lugar a la potencia e (es decir, cifrando) y luego elevando a la potencia d (es decir, descifrando) obtenemos el valor original, m . Aunque más maravilloso es el hecho de que si primero elevamos a la potencia d y luego a la potencia e (es decir, si invertimos el orden del cifrado y el descifrado, realizando en primer lugar la operación de descifrado y aplicando luego la operación de cifrado) también obtenemos el valor original, m . Este estupendo resultado se concluye inmediatamente a partir de las reglas de la propia aritmética modular:

$$(m^d \bmod n)^e \bmod n = m^{de} \bmod n = m^{ed} \bmod n = (m^e \bmod n)^d \bmod n$$

La seguridad del algoritmo RSA se basa en el hecho de que no existen algoritmos conocidos para factorizar rápidamente un número, en este caso el valor público n , con el fin de obtener los números primos p y q . Si alguien conociera p y q , entonces podría calcular fácilmente a partir del valor público e la clave secreta d . Por otro lado, no se conoce con seguridad si *existen* o no algoritmos rápidos para factorizar un número, por lo que, en este sentido, la seguridad de RSA no está garantizada.

Otro algoritmo popular de cifrado de clave pública es el algoritmo de Diffie-Hellman, que analizaremos brevemente en los problemas de repaso. El algoritmo de Diffie-Hellman no es tan versátil como RSA, en el sentido de que no puede utilizarse para cifrar mensajes de longitud arbitraria; sin embargo, sí que puede emplearse para establecer una clave de sesión simétrica, que a su vez se utilizará para cifrar los mensajes.

8.3 Integridad de los mensajes y firmas digitales

En la sección anterior hemos visto cómo se puede utilizar el cifrado para proporcionar confidencialidad a dos entidades que desean comunicarse. En esta sección vamos a volver nuestra atención al tema criptográfico, igualmente importante, de proporcionar **integridad a los mensajes** (técnica también conocida como autenticación de mensajes). Junto con la integridad de los mensajes, analizaremos dos temas relacionados en esta sección: las firmas digitales y la autenticación de los puntos terminales.

Vamos a definir el problema de la integridad de los mensajes utilizando una vez más a Alicia y a Benito. Suponga que Benito recibe un mensaje (que puede estar cifrado o puede ser texto en claro) y que él cree que este mensaje fue enviado por Alicia. Para autenticar el mensaje, Benito tiene que verificar que:

1. El origen del mensaje es efectivamente Alicia.
2. El mensaje no ha sido alterado mientras viajaba hasta Benito.

Veremos en las Secciones 8.4 a 8.7 que este problema de la integridad de los mensajes es una preocupación crítica en prácticamente todos los protocolos de red seguros.

Como ejemplo específico, considere una red de computadoras en la que se está empleando un algoritmo de enrutamiento de estado del enlace (como por ejemplo OSPF) para determinar las rutas entre cada pareja de routers de la red (véase el Capítulo 5). En un algoritmo de estado del enlace, cada router necesita multidifundir un mensaje de estado del enlace a todos los restantes routers de la red. El mensaje de estado del enlace de un router incluye una lista de sus vecinos directamente conectados, junto con los costes directos a esos vecinos. Una vez que un router recibe mensajes de estado de enlace de todos los demás routers puede crear un mapa completo de la red, ejecutar su algoritmo de enrutamiento de coste mínimo y configurar su tabla de reenvío. Un ataque relativamente sencillo contra el algoritmo de enrutamiento consiste en que Gertrudis distribuya mensajes falsos de estado del enlace con información incorrecta acerca del estado de los enlaces. Debido a la necesidad de integridad de los mensajes, cuando el router B recibe un mensaje de estado del enlace procedente del router A debe verificar que efectivamente el router A ha creado dicho mensaje y, además, que nadie lo ha alterado mientras que el mensaje se encontraba en tránsito.

En esta sección vamos a describir una popular técnica de integridad de mensajes que se utiliza en muchos protocolos de red seguros. Pero, antes de eso, tenemos que tratar otro tema importante dentro del campo de la criptografía: las funciones hash criptográficas.

8.3.1 Funciones hash criptográficas

Como se muestra en la Figura 8.7, una función hash toma una entrada, m , y calcula una cadena de tamaño fijo $H(m)$ conocida con el nombre de hash. La suma de comprobación de Internet (Capítulo 3) y los códigos CRC (Capítulo 6) cumplen con esta definición. Además, una **función hash criptográfica** necesita exhibir la siguiente propiedad adicional:

- Es computacionalmente impracticable encontrar dos mensajes distintos x e y tales que $H(x) = H(y)$.

De una manera informal, podríamos decir que esta propiedad significa que es computacionalmente impracticable que un intruso sustituya un mensaje protegido mediante la función hash por otro mensaje diferente. Es decir, si $(m, H(m))$ son el mensaje y el valor hash de dicho mensaje creado por el emisor, entonces un intruso no puede generar el contenido de otro mensaje, y , que tenga el mismo valor de hash que el mensaje original.

Vamos a verificar que una suma de comprobación simple, como la suma de comprobación de Internet, nos proporcionaría una función hash criptográfica bastante poco segura. En lugar de realizar la aritmética en complemento a 1 (como en la suma de comprobación de Internet), calculemos una suma de comprobación tratando cada carácter como un byte y sumando los bytes en fragmentos

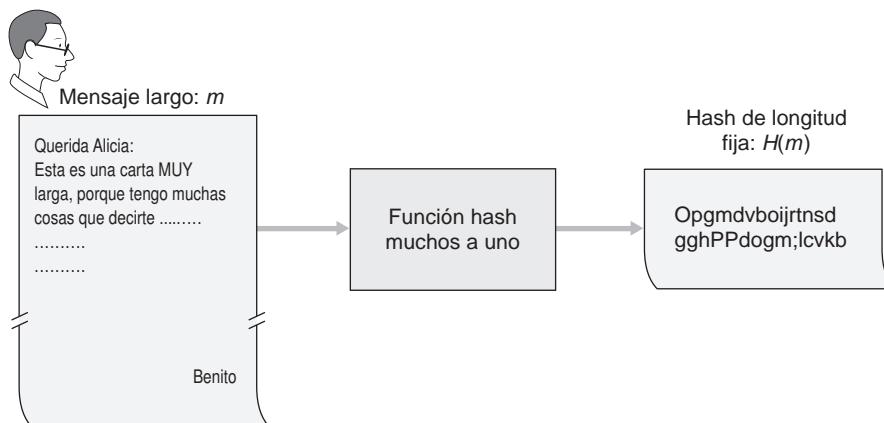


Figura 8.7 ♦ Funciones hash.

de 4 bytes cada vez. Suponga que Benito debe a Alicia 100,99 euros y que le envía un mensaje de confirmación que es la cadena de texto “DEBO100.99BENITO.” La representación ASCII (en notación hexadecimal) de estas letras sería 44,45,42,4F,31,30,30,2E,39,39,42,45,4E,49,54,4F.

La Figura 8.8 (parte superior) muestra que la suma de comprobación de 4 bytes para este mensaje es FC F8 09 11. Un mensaje ligeramente distinto (y mucho más costoso para Benito) es el que se muestra en la parte inferior de esta misma figura. Los mensajes “DEBO100.99BENITO” y “DEBO900.19BENITO” tienen la *misma* suma de comprobación. Por tanto, este sencillo algoritmo de suma de comprobación violaría el requisito que antes hemos mencionado. Dados los datos originales, es muy sencillo encontrar otro conjunto de datos con la misma suma de comprobación. Obviamente, para propósitos de seguridad necesitaremos una función hash bastante más potente que una mera suma de comprobación.

El algoritmo hash MD5 de Ron Rivest [RFC 1321] se utiliza ampliamente hoy día. Este algoritmo calcula un valor hash de 128 bits mediante un proceso en cuatro pasos, que consiste en un paso de relleno (añadir un uno seguido del número de ceros suficiente como para que la longitud del mensaje satisfaga ciertas condiciones), un paso de agregación (añadir una representación mediante 64 bits de la longitud del mensaje antes de la operación de relleno), una inicialización de un acumulador y un bucle final en el que se procesan los bloques de 16 palabras del mensaje, en cuatro pasadas sucesivas. Consulte [RFC 1321] para ver una descripción de MD5 (incluyendo una implementación con código fuente en C).

El segundo algoritmo principal de hash que se utiliza hoy día es el denominado algoritmo de hash seguro (SHA-1, *Secure Hash Algorithm*) [FIPS 1995]. Este algoritmo está basado en una serie de principios similares a los utilizados en el diseño de MD4 [RFC 1320], el predecesor de MD5. SHA-1, un estándar federal del gobierno de Estados Unidos, es obligatorio siempre que se requiera un algoritmo hash criptográfico para aplicaciones gubernamentales. Este algoritmo produce un resumen del mensaje (*message digest*) de 160 bits. Esa mayor longitud de salida hace que SHA-1 sea más seguro.

8.3.2 Código de autenticación del mensaje

Volvamos ahora al problema de la integridad de los mensajes. Ahora que sabemos qué son las funciones hash, veamos una primera aproximación de cómo podríamos garantizar la integridad de los mensajes:

Mensaje	Representación ASCII	
DEBO	44 45 42 4F	
100.	31 30 30 2E	
99BE	39 39 42 45	
NITO	<u>4E 49 54 4F</u>	
	FC F8 09 11	Suma de comprobación
Mensaje	Representación ASCII	
DEBO	44 45 42 4F	
900.	39 30 30 2E	
19BE	31 39 42 45	
NITO	<u>4E 49 54 4F</u>	
	FC F8 09 11	Suma de comprobación

Figura 8.8 ♦ El mensaje inicial y el mensaje fraudulento tienen la misma suma de comprobación.

1. Alicia crea el mensaje m y calcula el valor hash $H(m)$ (por ejemplo, con SHA-1).
2. Alicia añade a continuación $H(m)$ al mensaje m , creando un mensaje ampliado $(m, H(m))$, el cual envía a Benito.
3. Benito recibe el mensaje ampliado (m, h) y calcula $H(m)$. Si $H(m) = h$, Benito concluye que todo está correcto.

Este enfoque tiene un fallo fundamental. La intrusa Gertrudis puede crear un mensaje ficticio m' en el que dijera que es Alicia, calcular $H(m')$ y enviar a Benito $(m', H(m'))$. Cuando Benito recibiera el mensaje, todas las comprobaciones del paso 3 serían correctas, por lo que Benito no sería consciente de que se ha producido un engaño.

Para garantizar la integridad de los mensajes, además de utilizar funciones hash criptográficas Alicia y Benito necesitan un secreto compartido s . Este secreto compartido, que no es más que una cadena de bits, se denomina **clave de autenticación**. Utilizando este secreto compartido puede garantizarse de la forma siguiente la integridad de los mensajes:

1. Alicia crea el mensaje m , concatena s con m para crear $m + s$ y calcula el valor hash $H(m + s)$ (por ejemplo con SHA-1). $H(m + s)$ se denomina **código de autenticación de mensajes (MAC, Message Authentication Code)**.
2. Alicia añade entonces el código MAC al mensaje m , creando un mensaje ampliado $(m, H(m + s))$, y lo envía a Benito.
3. Benito recibe un mensaje ampliado (m, h) y, conociendo s , calcula el valor MAC $H(m + s)$. Si $H(m + s) = h$, Benito concluye que todo está correcto.

En la Figura 8.9 se muestra un resumen de este procedimiento. El lector debe fijarse en que las siglas MAC aquí (que corresponden a “*Message Authentication Code*”) no tienen nada que ver con las siglas MAC utilizadas en los protocolos de la capa de enlace (que corresponden a “*Medium Access Control*”).

Una característica muy conveniente de los valores MAC es que no se necesita ningún algoritmo de cifrado. De hecho, en muchas aplicaciones, incluyendo el algoritmo de enrutamiento de estado del enlace descrito anteriormente, las entidades que se están comunicando solo se preocupan de la integridad de los mensajes, mientras que la confidencialidad de los mismos no les importa. Utilizando un código MAC, las entidades pueden autenticar los mensajes que se intercambian sin tener que incluir complejos algoritmos de cifrado en el proceso de garantía de la integridad.

Como cabría esperar, a lo largo de los años se han propuesto diversos estándares para los valores MAC. El estándar más popular hoy día es **HMAC**, que puede utilizarse con MD5 o SHA-1. En la práctica, HMAC hace pasar los datos y la clave de autenticación a través de la función hash dos veces [Kaufman 1995; RFC 2104].

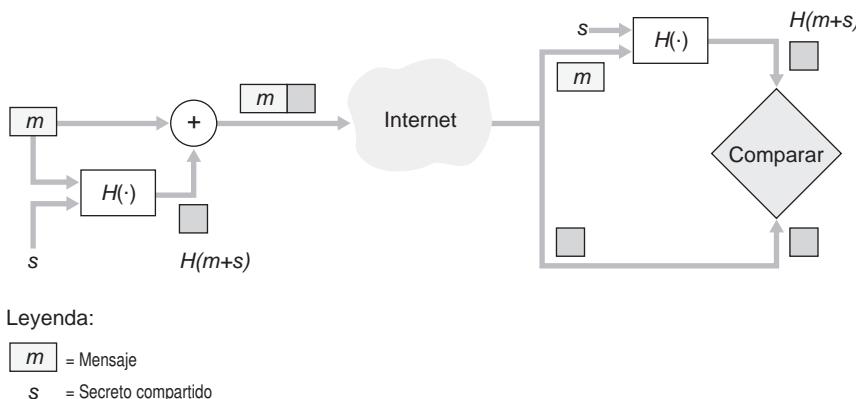


Figura 8.9 ♦ Código de autenticación de mensaje (MAC).

Sigue quedando pendiente una cuestión importante. ¿Cómo distribuimos la clave secreta de autenticación a las distintas entidades que tienen que comunicarse? Por ejemplo, en el algoritmo de enrutamiento de estado del enlace necesitaremos distribuir de alguna manera la clave de autenticación a cada uno de los routers del sistema autónomo. (Observe que todos los routers pueden utilizar la misma clave de autenticación.) Un administrador de red podría llevar a cabo esa distribución visitando físicamente cada uno de los routers. O bien, si el administrador de la red es demasiado perezoso y si cada router tiene su propia clave pública, el administrador puede distribuir la clave de autenticación a cualquiera de los routers cifrándola con la clave pública del router y luego enviando la clave cifrada hasta el router a través de la red.

8.3.3 Firmas digitales

Piense en el número de veces que ha firmado con su nombre en un papel durante la última semana. Todos estamos acostumbrados a firmar cheques, recibos de tarjetas de crédito, documentos legales y cartas. Nuestra firma atestigua el hecho de que nosotros (y no cualquier otra persona) hemos aceptado y/o acordado el contenido de un documento. En un mundo digital a menudo surge la necesidad de indicar el propietario o creador de un documento o de explicitar nuestro acuerdo con el contenido de un documento. Una **firma digital** es una técnica criptográfica que permite conseguir estos objetivos en el mundo digital.

Al igual que ocurre con las firmas manuscritas, las firmas digitales deben realizarse de forma que sean verificables y no falsificables. Es decir, debe ser posible demostrar que un documento firmado por una persona ha sido, de hecho, firmado por esa persona (la firma tiene que ser verificable) y que *solo* esa persona podría haber firmado el documento (la firma no puede ser falsificada).

Consideremos ahora cómo podríamos diseñar un esquema de firma digital. Observe que, cuando Benito firma un mensaje, debe poner algo en el mensaje que sea distintivo de él. Benito podría pensar en asociar un valor MAC para la firma, en cuyo caso crearía el valor MAC añadiendo su clave (que permite distinguirle del resto de las personas) al mensaje y luego aplicando la función hash. Pero, para que Alicia pudiera verificar esa firma, también debería disponer de una copia de la clave, en cuyo caso la clave dejaría de ser distintiva de Benito. Por tanto, los códigos MAC no nos permiten conseguir nuestro objetivo en este caso.

Recuerde que, con la criptografía de clave pública, Benito dispone de sendas claves pública y privada, siendo la pareja formada por esas claves distintiva de Benito. Por tanto, la criptografía de clave pública es un candidato excelente para poder proporcionar un mecanismo de firma digital. Veamos ahora cómo se estructura dicho mecanismo.

Suponga que Benito desea firmar digitalmente un documento, m . Podemos pensar en el documento como si fuera un archivo o un mensaje que Benito va a firmar y enviar. Como se muestra en la Figura 8.10, para firmar este documento Benito utiliza simplemente su clave privada, K_B^- , para calcular $K_B^-(m)$. De entrada, puede parecer extraño que Benito utilice su clave privada (que, como vimos en la Sección 8.2, se utiliza para descifrar un mensaje que haya sido cifrado con su clave pública) para firmar un documento. Pero recuerde que el cifrado y el descifrado no son otra cosa que operaciones matemáticas (consistentes en elevar a la potencia e o d en RSA; véase la Sección 8.2) y recuerde también que el objetivo de Benito no es cifrar u ocultar el contenido del documento, sino solo firmarlo de una manera que sea verificable y no falsificable. La firma digital del documento realizada por Benito es $K_B^-(m)$.

¿Cumple la firma digital $K_B^-(m)$ nuestros requisitos de que sea verificable y no falsificable? Suponga que Alicia dispone de m y $K_B^-(m)$. Ella quiere demostrar ante un tribunal (a consecuencia de algún tipo de litigio) que Benito es quien ha firmado el documento y que es la única persona que podría haberlo firmado. Alicia toma la clave pública de Benito, K_B^+ , y se la aplica a la firma digital, $K_B^-(m)$, asociada con el documento, m . Es decir, calcula $K_B^+(K_B^-(m))$, y *voilà*: con un gesto dramático obtiene m , que se corresponde exactamente con el documento original. Alicia argumenta a continuación que solo Benito podría haber firmado el documento por las siguientes razones:

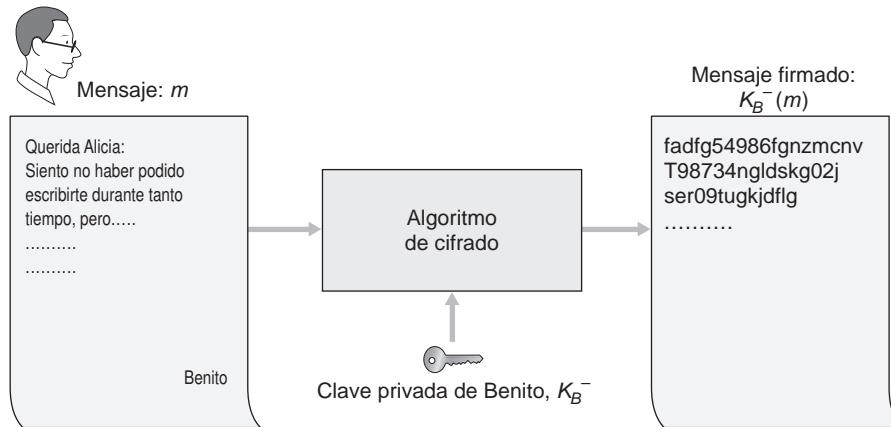


Figura 8.10 ♦ Creación de una firma digital para un documento.

- Quienquiera que haya firmado el documento tiene que haber utilizado la clave privada, K_B^- , para calcular la firma $K_B^-(m)$, tal que $K_B^+(K_B^-(m)) = m$.
- La única persona que puede conocer la clave privada, K_B^- , es el propio Benito. Recuerde, de nuestra exposición sobre RSA de la Sección 8.2, que el conocimiento de la clave pública, K_B^+ , no sirve de nada a la hora de determinar la clave privada, K_B^- . Por tanto, la única persona que podría conocer K_B^- es aquella que haya generado la pareja de claves, (K_B^+, K_B^-) , es decir, Benito. (Observe que aquí se está suponiendo, sin embargo, que Benito no ha proporcionado su clave privada K_B^- a nadie y que nadie le ha robado K_B^- .)

También es importante observar que si el documento original, m , se modificara de algún modo para obtener una forma alternativa, m' , la firma que Benito generara para m no sería válida para m' , ya que $K_B^+(K_B^-(m))$ no es igual a m' . Por tanto, podemos concluir que las firmas digitales también proporcionan un mecanismo de integridad de los mensajes, permitiendo al receptor verificar que el mensaje no ha sido alterado, además de verificar el origen del mismo.

Uno de los problemas con la firma de datos mediante mecanismos de cifrado es que el cifrado y el descifrado son computacionalmente muy caros. Dada la cantidad adicional de procesamiento que el cifrado y el descifrado exigen, el firmar los datos vía cifrándolos/descifrándolos completamente puede ser como matar moscas a cañonazos. Una técnica más eficiente consiste en introducir funciones hash en el mecanismo de firma digital. Recuerde de la Sección 8.3.2 que un algoritmo hash toma un mensaje, m , de longitud arbitraria y calcula una especie de “huella digital” de longitud fija para el mensaje, que designamos mediante $H(m)$. Utilizando una función hash, Benito firma el valor hash de un mensaje en lugar de firmar el propio mensaje, es decir, Benito calcula $K_B^-(H(m))$. Puesto que $H(m)$ es, generalmente, mucho más pequeño que el mensaje original m , la capacidad de proceso necesario para generar la firma digital se reduce sustancialmente.

En el contexto del envío de un mensaje a Alicia por parte de Benito, la Figura 8.11 proporciona un resumen del procedimiento operativo requerido para crear una firma digital. Benito hace pasar su mensaje original, de gran longitud, a través de una función hash. A continuación, firma digitalmente el valor hash resultante utilizando para ello su clave privada. Después, le envía a Alicia el mensaje original (como texto en claro) junto con el resumen del mensaje digitalmente firmado (al que a partir de ahora denominaremos firma digital). La Figura 8.12 proporciona un resumen del procedimiento operativo de firma. Alicia aplica la clave pública del emisor al mensaje para obtener un valor hash. Asimismo, aplica la función hash al mensaje recibido como texto en claro, para obtener un segundo valor hash. Si los dos valores coinciden, entonces Alicia puede estar segura acerca de la integridad y del autor del mensaje.

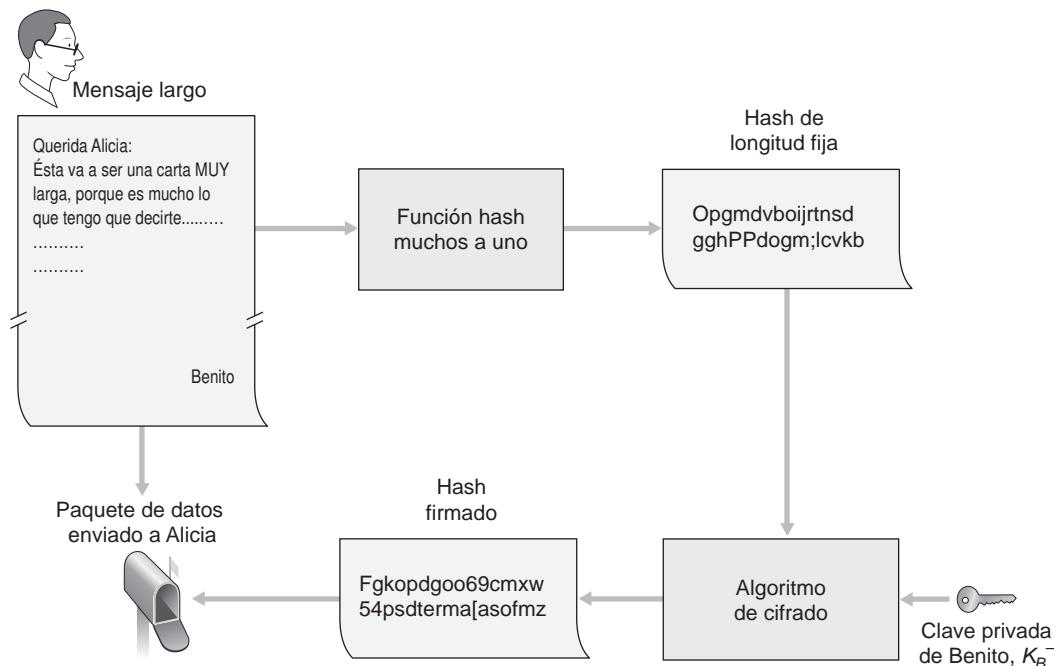


Figura 8.11 ♦ Transmisión de un mensaje firmado digitalmente.

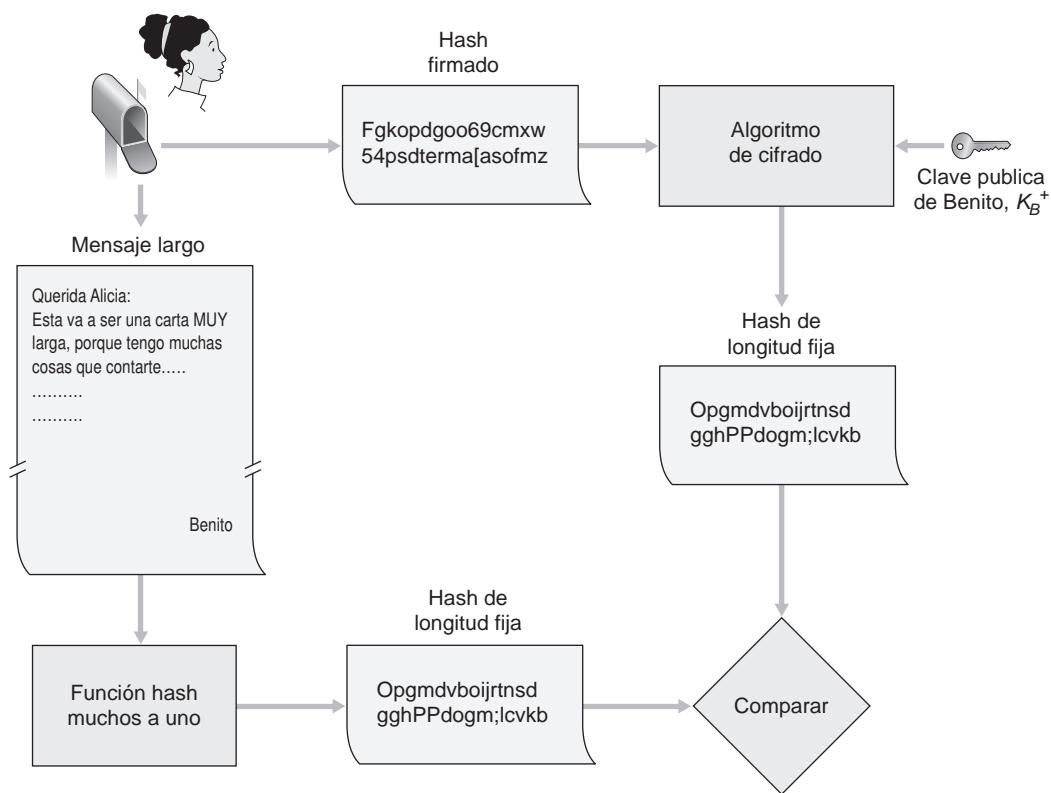


Figura 8.12 ♦ Verificación de un mensaje firmado.

Antes de seguir adelante, vamos a comparar brevemente las firmas digitales con los códigos MAC, dado que existen paralelismos entre ambos sistemas, pero también existen varias diferencias sutiles e importantes. Tanto las firmas digitales como los códigos MAC comienzan con un mensaje (o un documento). Para obtener un valor MAC de ese mensaje, añadimos al mensaje una clave de autenticación y luego calculamos el valor hash del resultado. Observe que, a la hora de crear el valor MAC, no hay involucrado ningún mecanismo de cifrado de clave pública ni de clave simétrica. Para crear una firma digital, primero calculamos el valor hash del mensaje y luego ciframos el mensaje con nuestra clave privada (utilizando criptografía de clave pública). Por tanto, la firma digital es una técnica “más pesada”, dado que requiere una infraestructura de clave pública (PKI, *Public Key Infrastructure*) subyacente, en la que existan autoridades de certificación, como las que más adelante se describen. En las Secciones 8.5 y 8.6 veremos que los valores MAC también se emplean en varios protocolos de seguridad populares de las capas de transporte y de red.

Certificación de clave pública

Una importante aplicación de las firmas digitales es la **certificación de clave pública**, es decir, certificar que una clave pública pertenece a una entidad específica. Las técnicas de certificación de clave pública se utilizan en muchos protocolos populares de seguridad para las comunicaciones de red, incluyendo IPsec y SSL.

Para tratar de comprender el problema, consideremos una versión de comercio electrónico por Internet del clásico servicio de pizza a domicilio. Alicia trabaja en el sector de la venta de pizza a domicilio y acepta pedidos a través de Internet. Benito, un amante de la pizza, envía a Alicia un mensaje de texto en claro que incluye su dirección particular y el tipo de pizza que desea. En ese mensaje, Benito también incluye una firma digital (es decir, un valor hash firmado del mensaje en claro original) para demostrar a Alicia que él es el verdadero origen del mensaje. Para verificar la firma, Alicia obtiene la clave de pública de Benito (quizá acudiendo a un servidor de clave pública o a partir del propio mensaje de correo electrónico) y comprueba la firma digital. De esta forma se asegura de que Benito, y no algún adolescente gracioso, ha realizado el pedido.

Este procedimiento parece correcto, hasta que nuestra intrusa Gertrudis entra en escena. Como se muestra en la Figura 8.13, Gertrudis está tratando de gastar una broma y envía un mensaje a Alicia en el que dice que es Benito, proporciona la dirección particular de Benito y pide una pizza. En este mensaje también incluye su clave pública (la de Gertrudis), aunque Alicia supone, naturalmente, que se trata de la clave pública de Benito. Gertrudis también adjunta una firma digital, que habrá creado con su propia clave privada. Después de recibir el mensaje, Alicia aplica la clave pública de Gertrudis (pensando que es la de Benito) a la firma digital y concluye que el mensaje de texto en claro ha sido creado por Benito. Benito se quedará muy sorprendido cuando el repartidor le lleve a su casa una pizza con pepperoni y anchoas.

Podemos ver a partir de este ejemplo que para que la criptografía de clave pública resulte útil necesitamos poder verificar que disponemos de la verdadera clave pública de la entidad (persona, router, navegador, etc.) con la que queremos comunicarnos. Por ejemplo, cuando Alicia quiere comunicarse con Benito empleando criptografía de clave pública necesita verificar que la clave pública que se supone que pertenece a Benito es verdaderamente de Benito.

La asociación entre una clave pública y una entidad concreta normalmente la realiza una **Autoridad de certificación (CA, Certification Authority)**, cuyo trabajo consiste en validar las identidades y emitir los certificados. Una autoridad de certificación desempeña las siguientes funciones:

1. Una CA verifica que una entidad (una persona, un router, etc.) es quien dice ser. No hay ningún procedimiento establecido para la forma en que se lleva a cabo esa certificación. A la hora de tratar con una CA, uno debe confiar en que esa autoridad de certificación habrá realizado una verificación de identidad adecuadamente rigurosa. Por ejemplo, si Gertrudis fuera capaz de entrar en una autoridad de certificación y anunciar simplemente “Yo soy Alicia” y recibir certificados asociados con la identidad de Alicia, entonces ninguno podríamos tener mucha fe

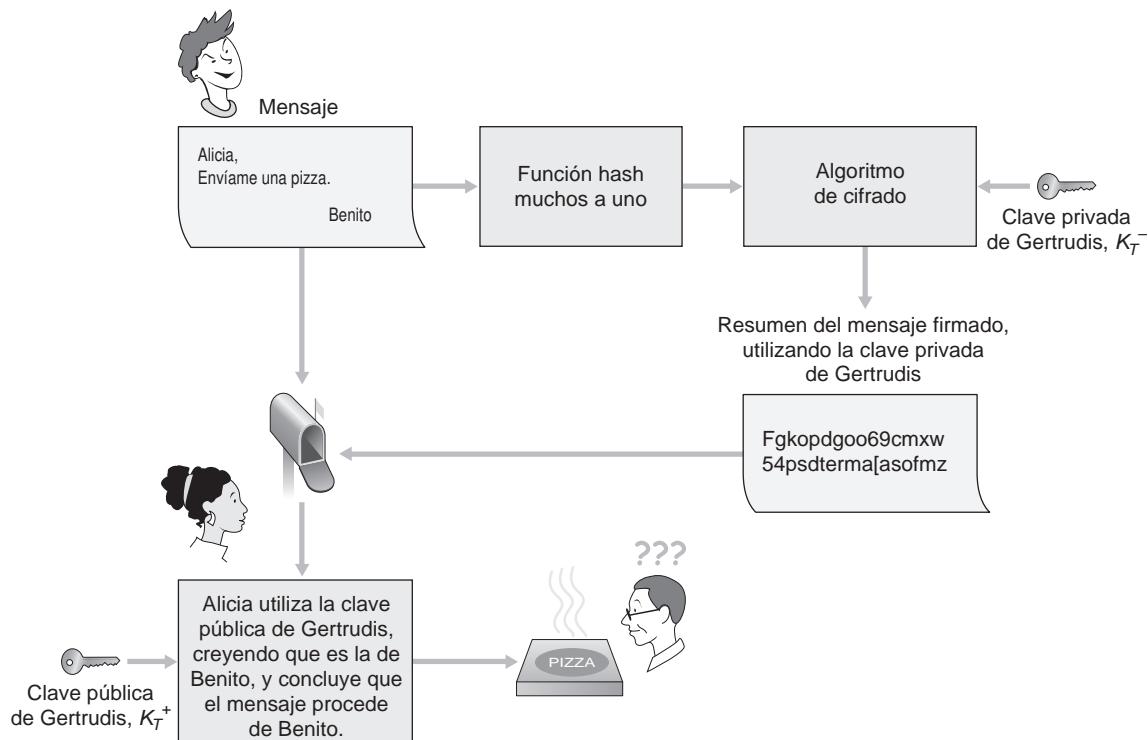


Figura 8.13 ♦ Gertrudis se hace pasar por Benito usando criptografía de clave pública.

en las claves públicas certificadas por esa autoridad de certificación concreta. Por otro lado, uno tendería (¡o quizás no!) a confiar más en una autoridad de certificación que dependa del gobierno. Solo podemos confiar en la identidad asociada con una clave pública en la medida en que podamos confiar en una autoridad de certificación y en sus técnicas de verificación de identidad. Como ve, lo que es necesario establecer es una auténtica red de confianza entre distintas entidades.

2. Una vez que la autoridad de certificación verifica la identidad de la entidad, genera un **certificado** que asocia con esa entidad su correspondiente clave pública. El certificado contiene la clave pública y la información de identificación globalmente distintiva acerca del propietario de la clave pública (por ejemplo, el nombre de una persona o una dirección IP). El certificado es firmado digitalmente por la autoridad de certificación. Estos pasos se muestran en la Figura 8.14.

Veamos ahora cómo pueden utilizarse los certificados para combatir a los bromistas de la pizza como Gertrudis y a otros sujetos indeseables. Cuando Benito hace su pedido, también envía su certificado firmado por la autoridad de certificación. Alicia utiliza la clave pública de la autoridad de certificación para comprobar la validez del certificado de Benito y extraer la clave pública de Benito.

Tanto la Unión Internacional de Telecomunicaciones (ITU, *International Telecommunication Union*) como el IETF han desarrollado estándares para las autoridades de certificación. ITU X.509 [ITU 2005a] especifica un servicio de autenticación, así como una sintaxis específica para los certificados. [RFC 1422] describe un mecanismo de administración de claves basado en autoridades de certificación para su utilización con el correo electrónico seguro a través de Internet. Es compatible con X.509, pero va más allá de dicho estándar, al establecer procedimientos y convenios para una arquitectura de gestión de claves. La Tabla 8.4 describe algunos de los campos más importantes de un certificado.

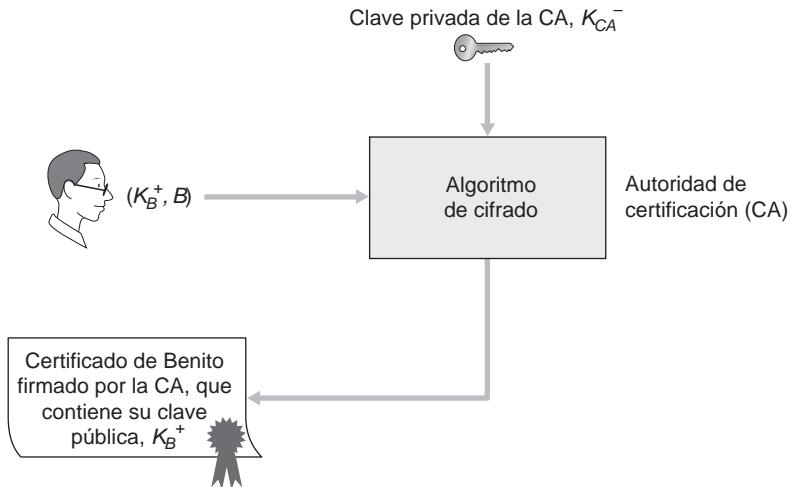


Figura 8.14 ♦ Benito hace que una autoridad de certificación (CA) certifique su clave pública.

Nombre del campo	Descripción
Versión	Número de versión de la especificación X.509.
Número de serie	Identificador único para un certificado emitido por una CA.
Firma	Especifica el algoritmo utilizado por la CA para firmar este certificado.
Nombre del emisor	Identidad de la CA que emite este certificado, en formato de nombre distintivo (DN, <i>Distinguished Name</i>) [RFC 4514].
Periodo de validez	Inicio y final del periodo de validez del certificado.
Nombre del sujeto	Identidad de la entidad cuya clave pública está asociada con este certificado, en formato DN.
Clave pública del sujeto	La clave pública del sujeto, así como una indicación del algoritmo de clave pública (y de los parámetros del algoritmo) con el que hay que usar esta clave.

Tabla 8.4 ♦ Campos seleccionados en una clave pública X.509 y RFC 1422.

8.4 Autenticación del punto terminal

La **autenticación del punto terminal** es el proceso de demostrar a alguien la propia identidad a través de una red de computadoras. Por ejemplo, un usuario demostrando su identidad a un servidor de correo electrónico. En la vida cotidiana las personas nos autenticamos mutuamente de muchas formas: reconocemos nuestras caras cuando nos encontramos, reconocemos nuestras voces a través del teléfono, nos autentica el oficial de aduanas que comprueba si nos parecemos a la fotografía de nuestro pasaporte.

En esta sección vamos a pensar en cómo podría una de las partes autenticar a su interlocutor cuando ambos se estén comunicando a través de la red. Nos vamos a centrar aquí en cómo autenticar a un interlocutor “en vivo”, en el preciso instante en que la comunicación está teniendo lugar. Un ejemplo concreto sería el de un usuario autenticándose ante un servidor de correo electrónico. Este problema difiere de manera sutil del hecho de demostrar que un mensaje recibido en el pasado proviene, efectivamente, del supuesto emisor, como hemos visto en la Sección 8.3.

A la hora de realizar la autenticación a través de la red los interlocutores no pueden utilizar información biométrica, como la apariencia visual o el patrón de voz. De hecho, veremos en nuestros casos de estudio posteriores que a menudo son elementos de la red, como routers y procesos cliente/servidor, los que deben autenticarse mutuamente. En esos casos, la autenticación debe realizarse basándose exclusivamente en los mensajes y datos intercambiados como parte de un **protocolo de autenticación**. Normalmente, el protocolo de autenticación se ejecutará *antes* de que los dos interlocutores ejecuten algún otro protocolo (por ejemplo, un protocolo de transferencia fiable de datos, un protocolo de intercambio de información de enrutamiento o un protocolo de correo electrónico). Primero, el protocolo de autenticación establece las identidades de los interlocutores a satisfacción de ambos; solo después de realizada la autenticación, los interlocutores acometerán la tarea que tengan entre manos.

Como en el caso de nuestro desarrollo de un protocolo de transferencia fiable de datos en el Capítulo 3, nos resultará instructivo aquí desarrollar varias versiones de un protocolo de autenticación, al que llamaremos **ap**, y ver los defectos de cada versión a medida que vayamos avanzando. (Si les gusta esta evolución paso a paso de un diseño, también disfrutarán con [Bryant 1988], que incluye una narración de una conversación ficticia entre diseñadores de un sistema abierto de autenticación de red y su descubrimiento de los múltiples y sutiles problemas implicados.)

Vamos a suponer que Alicia necesita autenticarse ante Benito.

8.4.1 Protocolo de autenticación *ap1.0*

Quizá el protocolo de autenticación más simple que podamos imaginar es uno en el que Alicia simplemente envíe un mensaje a Benito diciéndole que es Alicia. Este protocolo se ilustra en la Figura 8.15. El fallo aquí resulta obvio: no hay forma de que Benito compruebe que la persona que está enviando el mensaje “Soy Alicia” es, efectivamente, Alicia. Por ejemplo, Gertrudis (la intrusa) podría también enviar ese mensaje.

8.4.2 Protocolo de autenticación *ap2.0*

Si Alicia dispone de una dirección de red bien conocida (como por ejemplo una dirección IP) desde la que siempre se comunica, Benito podría tratar de autenticar a Alicia verificando que la dirección de origen del datagrama IP que transporta el mensaje de autenticación se corresponde con la dirección bien conocida de Alicia. Si es así, Alicia sería autenticada. Esto podría impedir que algún intruso sin demasiados conocimientos de redes se hiciera pasar por Alicia, ¡pero no detendría a los lectores de este libro, ni a muchos otros!

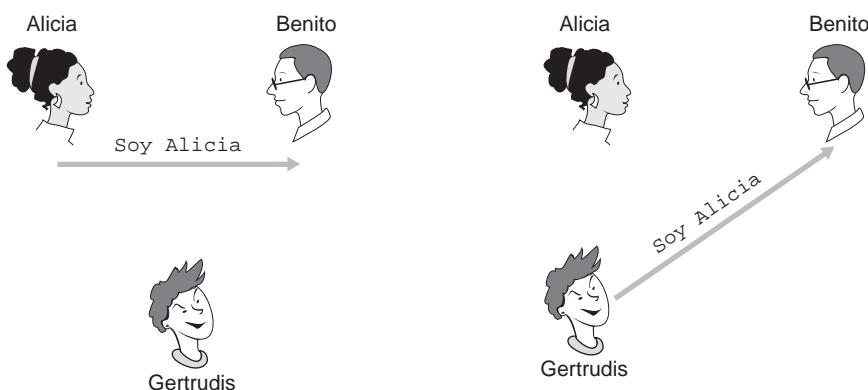


Figura 8.15 ♦ Protocolo *ap1.0* y un posible escenario de fallo.

De nuestro estudio de las capas de red y de enlace de datos, sabemos que no es difícil (por ejemplo, si uno tiene acceso al código del sistema operativo y puede construir su propio kernel del sistema operativo, como es el caso con Linux y varios otros sistemas operativos disponibles de forma gratuita) crear un datagrama IP, incluir en él cualquier dirección IP de origen que deseemos (por ejemplo, la dirección IP bien conocida de Alicia) y enviar el datagrama a través del protocolo de la capa de enlace hacia el router de primer salto. A partir de ahí, el datagrama con la dirección de origen incorrecta sería diligentemente reenviado hacia Benito. Esta técnica, ilustrada en la Figura 8.16, es una forma de suplantación IP. La suplantación IP puede evitarse si el router de primer salto de Gertrudis está configurado para reenviar solo aquellos datagramas que contengan la verdadera dirección IP de origen de Gertrudis [RFC 2827]. Sin embargo, esta capacidad no está universalmente implantada, ni tampoco se la impone de manera universal. Por ello, Benito sería muy ingenuo si asumiera que el administrador de la red de Gertrudis (que podría ser la propia Gertrudis) ha configurado el router de primer salto de Gertrudis para que solo reenvíe los datagramas que contengan direcciones auténticas.

8.4.3 Protocolo de autenticación *ap3.0*

Una solución clásica al problema de la autenticación consiste en usar una contraseña secreta. La contraseña es un secreto compartido por el autenticador y la persona que está siendo autenticada. Gmail, Facebook, Telnet, FTP y muchos otros servicios utilizan autenticación de contraseñas. En el protocolo *ap3.0* Alicia envía su contraseña secreta a Benito, como se ilustra en la Figura 8.17.

Dado lo mucho que se utilizan las contraseñas, podríamos tender a pensar que el protocolo *ap3.0* es suficientemente seguro. ¡Si pensáramos eso, nos equivocaríamos! El fallo de seguridad aquí está bastante claro: si Gertrudis espía las comunicaciones de Alicia, entonces puede averiguar cuál es su contraseña. Si piensa que esto es poco probable, considere el hecho de que, cuando se conecta con Telnet a otra máquina e inicia una sesión, la contraseña de inicio de sesión se envía sin cifrar al servidor Telnet. Cualquiera que esté conectado a la red LAN del cliente o del servidor Telnet podría interceptar (leer y almacenar) todos los paquetes transmitidos a través de la LAN y robar así la contraseña de inicio de sesión. De hecho, esta es una técnica bien conocida de robo de contraseñas (véase, por ejemplo, [Jimenez 1997]). Dicha amenaza es, obviamente, bastante real, por lo que está claro que *ap3.0* no nos sirve.

8.4.4 Protocolo de autenticación *ap3.1*

Nuestra siguiente idea para corregir *ap3.0* consiste, naturalmente, en cifrar la contraseña. Cifrando las contraseñas, podemos impedir que Gertrudis averigüe la contraseña de Alicia. Si asumimos que Alicia y Benito comparten una clave secreta simétrica, K_{A-B} , entonces Alicia puede cifrar la

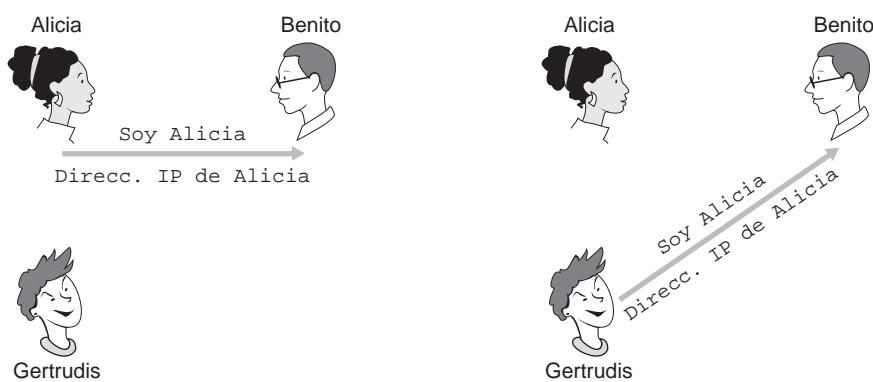


Figura 8.16 ♦ Protocolo *ap2.0* y un posible escenario de fallo.

contraseña y enviar a Benito su mensaje de identificación “Soy Alicia” junto con la contraseña cifrada. Benito descifra entonces la contraseña recibida y, suponiendo que la contraseña sea correcta, autentica a Alicia. Benito no tiene problema en autenticar a Alicia, ya que Alicia no solo conoce la contraseña, sino que también conoce el valor de la clave secreta compartida necesario para cifrar la contraseña. Vamos a llamar a este protocolo *ap3.1*.

Aunque es cierto que *ap3.1* impide a Gertrudis averiguar la contraseña de Alicia, el uso de la criptografía en este caso no resuelve el problema de la autenticación. Benito está sujeto a un posible **ataque por reproducción**: para hacerse pasar por Alicia, Gertrudis solo necesita interceptar las comunicaciones de Alicia, anotar la versión cifrada de la contraseña y mandar a Benito una copia de esa versión cifrada de la contraseña. El uso de una contraseña cifrada en *ap3.1* no cambia de forma significativa la situación con respecto a la del protocolo *ap3.0* de la Figura 8.17.

8.4.5 Protocolo de autenticación *ap4.0*

El escenario de fallo de la Figura 8.17 surgía del hecho de que Benito no podía distinguir entre la autenticación original de Alicia y la posterior reproducción de la autenticación original de Alicia. Es decir, Benito no podía determinar si Alicia se estaba comunicando “en vivo” (es decir, si estaba realmente al otro extremo de la comunicación en ese momento) o si los mensajes que estaba recibiendo eran la reproducción de una autenticación previa de Alicia, que hubiera sido grabada. El lector muy (*muy*) atento recordará que el protocolo de negociación en tres fases de TCP tenía que enfrentarse a este mismo problema: el lado del servidor de una conexión TCP no quería aceptar una conexión si el segmento SYN recibido era una copia antigua (retransmisión) de un segmento SYN de una conexión anterior. ¿Cómo resolvía el lado del servidor TCP el problema de determinar si el cliente estaba realmente comunicándose en vivo? Lo que hacía era elegir un número inicial de secuencia que no se hubiera utilizado durante muchísimo tiempo, enviar ese número al cliente y luego esperar a que el cliente respondiera con un segmento ACK que contuviera dicho número. Podemos adoptar aquí la misma idea de cara a la autenticación.

Un **número distintivo (nonce)** es un número que un protocolo solo utilizará una vez en la vida. Es decir, una vez que un protocolo emplea un número distintivo, nunca volverá a usar ese número. Nuestro protocolo *ap4.0* usa los números distintivos de la forma siguiente:

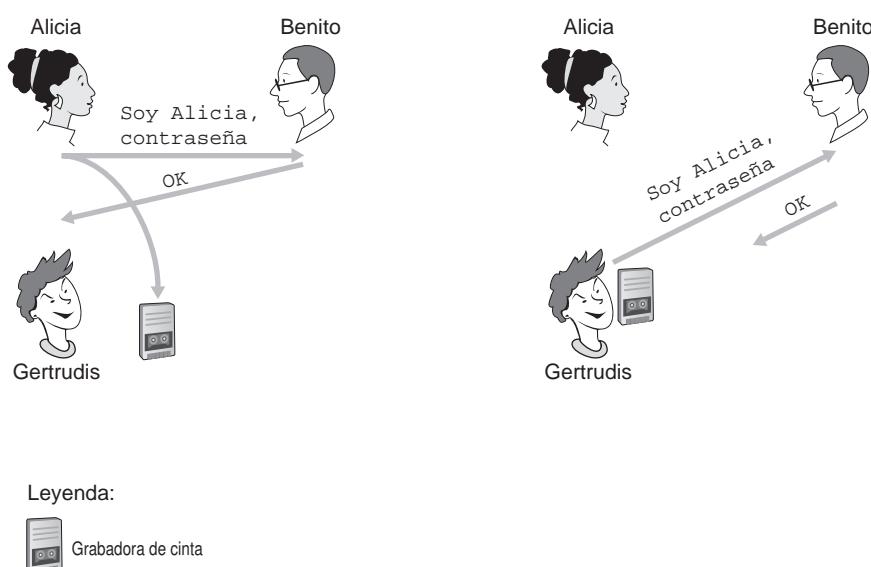


Figura 8.17 ♦ Protocolo *ap3.0* y un posible escenario de fallo.

1. Alicia envía a Benito el mensaje "Soy Alicia".
2. Benito selecciona un número distintivo, R , y se lo envía a Alicia.
3. Alicia cifra el número distintivo mediante la clave secreta simétrica que comparten Alicia y Benito, K_{A-B} , y devuelve el número distintivo cifrado $K_{A-B}(R)$ a Benito. Como en el protocolo *ap3.1*, el hecho de que Alicia conozca K_{A-B} y la use para cifrar un valor permite a Benito saber que el mensaje recibido ha sido generado por Alicia. El número distintivo se utiliza para cerciorarse de que Alicia se está comunicando en vivo.
4. Benito descifra el mensaje recibido. Si el número distintivo descifrado coincide con el que envió a Alicia, Alicia quedará autenticada.

El protocolo *ap4.0* se ilustra en la Figura 8.18. Utilizando el valor distintivo, R , y comprobando el valor devuelto, $K_{A-B}(R)$, Benito puede asegurarse de que Alicia es quien dice ser (ya que conoce el valor de la clave secreta necesaria para cifrar R) y está comunicándose en vivo (ya que ha cifrado el número distintivo, R , que Benito acaba de generar).

El uso de un número distintivo y de la criptografía de clave simétrica forman la base de *ap4.0*. Una pregunta natural es si podemos usar un número distintivo y criptografía de clave pública (en lugar de criptografía de clave simétrica) para resolver el problema de la autenticación. Esta cuestión se explora en los problemas del final del capítulo.

8.5 Asegurando el correo electrónico

En las secciones anteriores hemos examinado una serie de problemas fundamentales en el campo de la seguridad de red, incluyendo las técnicas de criptografía de clave simétrica y de clave pública, las de autenticación de punto terminal, los mecanismos de distribución de claves, el problema de la integridad de los mensajes y las técnicas de firma digital. Ahora vamos a examinar cómo se utilizan hoy en día dichas herramientas para proporcionar seguridad en Internet.

Es interesante observar que es posible proporcionar servicios de seguridad en cualquiera de las cuatro capas superiores de la pila de protocolos de Internet. Cuando se proporciona seguridad para un protocolo específico de la capa de aplicación, la aplicación que utiliza ese protocolo disfrutará de uno o más servicios de seguridad, como los de confidencialidad, autenticación o integridad. Cuando los mecanismos de seguridad se proporcionan para un protocolo de la capa de transporte, todas las aplicaciones que usan dicho protocolo disfrutarán de los servicios de seguridad del protocolo de transporte. Cuando la seguridad se proporciona en la capa de red en un esquema host a host, todos los segmentos de la capa de transporte (y por tanto todos los datos de la capa de aplicación) disfrutarán de los servicios de seguridad de la capa de red. Cuando se proporciona seguridad a nivel de enlace, entonces los datos de todas las tramas que viajan a través del enlace utilizarán los servicios de seguridad del enlace.

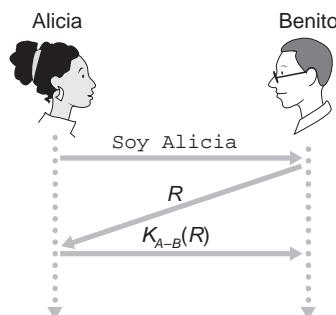


Figura 8.18 ♦ Protocolo *ap4.0*.

En las Secciones 8.5 a 8.8 vamos a examinar el modo en que se utilizan las técnicas de seguridad en las capas de aplicación, transporte, red y de enlace. Para ser coherentes con la estructura general del libro, comenzaremos por la parte superior de la pila de protocolos, abordando la seguridad en la capa de aplicación. Nuestro enfoque consiste en utilizar una aplicación específica, como el correo electrónico, como caso de estudio de las técnicas de seguridad de la capa de aplicación. Después descenderemos por la pila de protocolos y examinaremos el protocolo SSL (que proporciona seguridad a la capa de transporte), IPsec (que proporciona seguridad en la capa de red) y los mecanismos de seguridad del protocolo IEEE 802.11 para redes LAN inalámbricas.

El lector puede estarse preguntando por qué se proporciona la funcionalidad de seguridad en más de una capa dentro de Internet. ¿No bastaría simplemente con proporcionar la funcionalidad de seguridad en la capa de red? Existen dos respuestas a esta pregunta. En primer lugar, aunque la seguridad en la capa de red puede proporcionar la funcionalidad básica de cifrado de todos los datos contenidos en los datagramas (es decir, de todos los segmentos de la capa de transporte) y de autenticar todas las direcciones IP de origen, lo que no puede es ofrecer seguridad de nivel de usuario. Por ejemplo, un sitio web de comercio electrónico no puede confiar en la seguridad de la capa IP para autenticar a un cliente que esté comprando bienes o servicios en ese sitio. Por tanto, existe una necesidad de incorporar funcionalidad de seguridad en las capas superiores, además de la funcionalidad básica en las capas inferiores. En segundo lugar, es más fácil generalmente implantar servicios Internet nuevos, incluyendo los servicios de seguridad, en las capas superiores de la pila de protocolos. Mientras se espera a que un nuevo mecanismo de seguridad se implante de forma generalizada en la capa de red, lo que puede llegar a tardar años, muchos desarrolladores de aplicaciones simplemente se ponen manos a la obra e introducen la funcionalidad de seguridad en sus aplicaciones favoritas. Un ejemplo clásico es PGP (*Pretty Good Privacy*), que proporciona correo electrónico seguro (y que estudiaremos posteriormente en esta sección). Requiriendo únicamente código de aplicación de cliente y de servidor, PGP fue una de las primeras tecnologías de seguridad que se utilizó ampliamente en Internet.

8.5.1 Correo electrónico seguro

Ahora vamos a utilizar los principios criptográficos de las Secciones 8.2 a 8.3 para crear un sistema de correo electrónico seguro. Crearemos este diseño de alto nivel de una forma incremental, introduciendo en cada paso nuevos servicios de seguridad. A la hora de diseñar un sistema de correo electrónico seguro debemos tener presente el ejemplo visto en la Sección 8.1: la relación amorosa entre Alicia y Benito. Imagine que Alicia quiere enviar un mensaje de correo electrónico a Benito y que Gertrudis pretende inmiscuirse en la conversación.

Antes de seguir adelante y diseñar un sistema de correo electrónico seguro para Alicia y Benito, tenemos que considerar qué características de seguridad serían más deseables para ellos. La primera y principal es la *confidencialidad*. Como hemos explicado en la Sección 8.1, ni Alicia ni Benito quieren que Gertrudis lea el correo electrónico de Alicia. La segunda característica que Alicia y Benito probablemente deseen para su sistema de correo electrónico seguro es la de *autenticación del emisor*. En particular, cuando Benito recibe el mensaje "Ya no te quiero. No quiero verte nunca más. Olvídate, Alicia." naturalmente le gustaría estar seguro de que el mensaje procede de Alicia y no de Gertrudis. Otra característica que los dos amantes apreciarían sería la de *integridad de los mensajes*, es decir, una garantía de que los mensajes que Alicia envíe no sean modificados a lo largo del camino hasta llegar a Benito. Por último, el sistema de correo electrónico debe proporcionar una *autenticación del receptor*; es decir, Alicia querrá asegurarse de que está enviando su mensaje a Benito y no a alguna otra persona (por ejemplo, Gertrudis) que esté haciendo pasar por Benito.

Así que comenzemos analizando el primero de los objetivos, la confidencialidad. La forma más directa de proporcionar confidencialidad es que Alicia cifre el mensaje con una tecnología de clave simétrica (como DES o AES) y que Benito descifre el mensaje al recibirlo. Como se explica en la Sección 8.2, si la clave simétrica es lo suficiente larga y si únicamente Alicia y Benito tienen la clave, entonces es extremadamente difícil que ninguna otra persona (incluyendo a Gertrudis) lea el

mensaje. Aunque esta técnica es muy sencilla, presenta la dificultad fundamental de la que ya hemos hablado en la Sección 8.2: distribuir una clave simétrica de modo que solo Alicia y Benito tengan copia de la misma. Por tanto, vamos a considerar una solución alternativa: la criptografía de clave pública (utilizando, por ejemplo, RSA). En la técnica de clave pública, Benito pone a disposición de todo el mundo su clave pública (por ejemplo, en un servidor de claves públicas o en su página web personal). Alicia cifra su mensaje con la clave pública de Benito y envía el mensaje cifrado a la dirección de correo electrónico de Benito. Cuando este recibe el mensaje, simplemente lo descifra con su clave privada. Suponiendo que Alicia esté segura de que la clave pública utilizada es la de Benito, esta técnica constituye un método excelente para proporcionar la confidencialidad deseada. Sin embargo, un problema es que el cifrado de clave pública es relativamente poco eficiente, particularmente para mensajes de gran longitud.

Para solventar este problema de eficiencia, vamos a hacer uso de una clave de sesión (de lo que hemos hablado en la Sección 8.2.2). En particular, Alicia (1) selecciona una clave de sesión simétrica aleatoria, K_S , (2) cifra su mensaje, m , con la clave simétrica, (3) cifra la clave simétrica con la clave pública de Benito, K_B^+ , (4) concatena el mensaje cifrado y la clave simétrica cifrada para formar un “paquete” y (5) envía este paquete a la dirección de correo electrónico de Benito. Los pasos se ilustran en la Figura 8.19. (En esta y las siguientes figuras, el símbolo “+” encerrado en un círculo representa la operación de concatenación y el signo “−” encerrado en un círculo representa la operación de desconexión.) Cuando Benito recibe el paquete, (1) utiliza su clave privada, K_B^- , para obtener la clave simétrica, K_S , y (2) utiliza la clave simétrica K_S para descifrar el mensaje m .

Habiendo diseñado un sistema de correo electrónico seguro que proporciona un servicio de confidencialidad, vamos a diseñar ahora otro sistema que proporcione tanto autenticación del emisor como integridad de los mensajes. Vamos a suponer por el momento que a Alicia y Benito ya no les preocupa la confidencialidad (¡quieren compartir sus sentimientos con todo el mundo!) y que solo les preocupa la autenticación del emisor y la integridad de los mensajes. Para llevar a cabo esta tarea, utilizaremos firmas digitales y resúmenes de mensajes, como se describe en la Sección 8.3. Específicamente, Alicia (1) aplica una función hash H (por ejemplo, MD5) a su mensaje, m , para obtener un resumen del mensaje, (2) firma el resultado de la función hash con su clave privada, K_A^- , para crear una firma digital, (3) concatena el mensaje original (no cifrado) con la firma para crear un paquete y (4) envía el paquete a la dirección de correo electrónico de Benito. Cuando este recibe el paquete, (1) aplica la clave pública de Alicia, K_A^+ , al resumen del mensaje firmado y (2) compara el resultado de esta operación con su propio valor hash, H , del mensaje. Estos pasos se ilustran en la Figura 8.20. Como se ha estudiado en la Sección 8.3, si los dos resultados coinciden, Benito puede estar seguro de que el mensaje procede de Alicia y no ha sido alterado.

Vamos a considerar ahora el diseño de un sistema de correo electrónico que proporcione confidencialidad, autenticación del emisor e integridad de los mensajes. Esto puede hacerse combinando los procedimientos de las Figuras 8.19 y 8.20. Alicia crea primero un paquete preliminar, exactamente como en la Figura 8.20, que está compuesto por su mensaje original y un valor hash firmado digitalmente del mensaje. A continuación, trata este paquete preliminar como un mensaje en sí mismo y envía este nuevo mensaje a través de los pasos del emisor de la Figura 8.19, creando un nuevo paquete que es enviado a Benito. Los pasos aplicados por Alicia se muestran en la Figura 8.21. Cuando Benito recibe el paquete, primero aplica su lado correspondiente de la Figura 8.19 y luego su lado correspondiente de la Figura 8.20. Debería quedar claro para el lector que este diseño permite conseguir el objetivo de proporcionar confidencialidad, autenticación del emisor e integridad de los mensajes. Observe que, con este esquema, Alicia utiliza la criptografía de clave pública dos veces: una vez con su propia clave privada y otra con la clave pública de Benito. De forma similar, Benito también emplea la criptografía de clave pública dos veces: una con su clave privada y otra con la clave pública de Alicia.

El diseño de correo electrónico seguro esbozado en la Figura 8.21 probablemente proporciona una seguridad satisfactoria para la mayoría de los usuarios de correo electrónico, en la mayoría de las ocasiones. Pero continuamos teniendo un problema importante que hay que resolver. El diseño

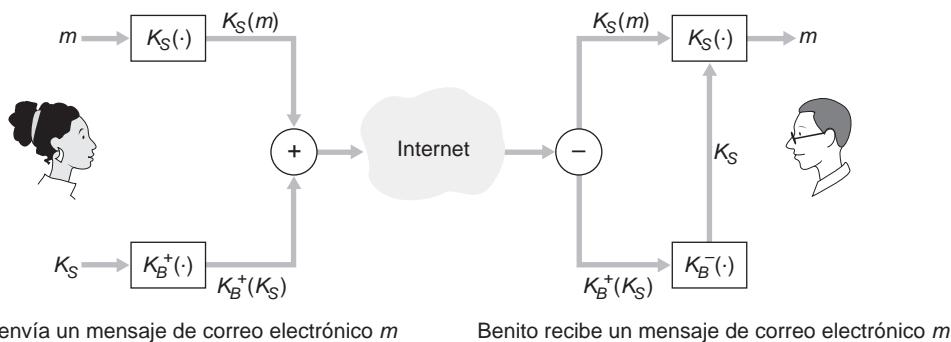


Figura 8.19 ♦ Alicia utiliza una clave de sesión simétrica, K_S , para enviar un mensaje secreto de correo electrónico a Benito.

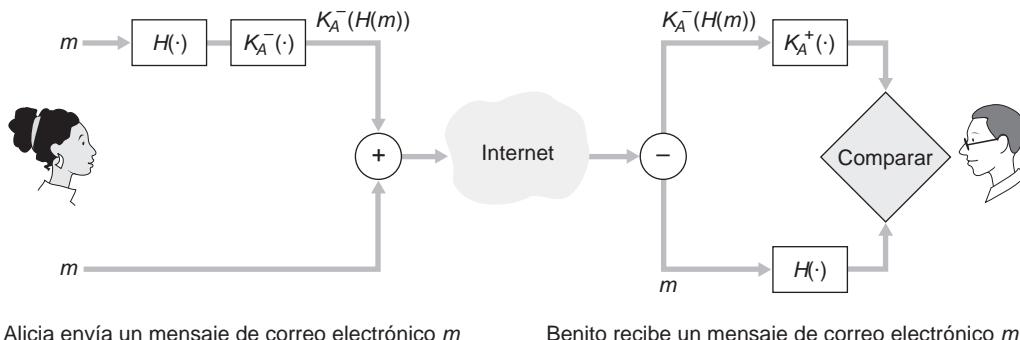


Figura 8.20 ♦ Utilización de funciones hash y firmas digitales para proporcionar autenticación del emisor e integridad de los mensajes.

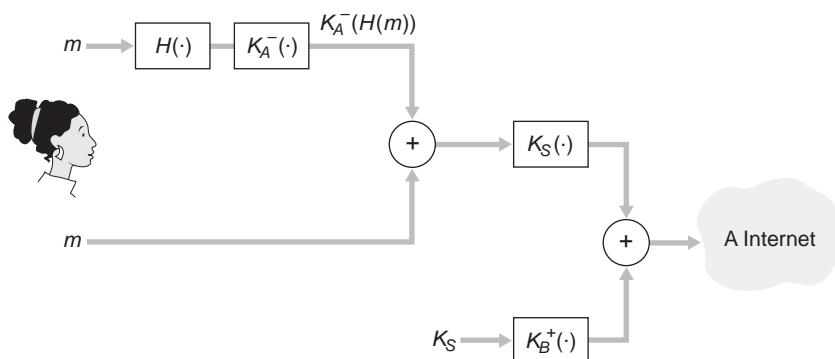


Figura 8.21 ♦ Alicia utiliza la criptografía de clave simétrica, la criptografía de clave pública, una función hash y una firma digital para obtener los servicios de confidencialidad, autenticación del emisor e integridad de los mensajes.

de la Figura 8.21 requiere que Alicia obtenga la clave pública de Benito y que Benito obtenga la clave pública de Alicia. La distribución de estas claves públicas no es un problema trivial. Por ejemplo, Gertrudis podría hacerse pasar por Benito y darle a Alicia su propia clave pública, mientras que le hace creer que se trata de la clave pública de Benito, lo que permitiría a Gertrudis recibir el

mensaje destinado a Benito. Como hemos visto en la Sección 8.3, una técnica popular para distribuir de forma segura las claves públicas consiste en *certificar* esas claves públicas utilizando una autoridad de certificación (CA).

8.5.2 PGP

Escrito por Phil Zimmermann en 1991, **Pretty Good Privacy (PGP)** es un buen ejemplo de esquema de cifrado de correo electrónico [PGPI 2016]. Hay versiones de PGP disponibles en el dominio público; por ejemplo, podemos encontrar el software PGP para nuestra plataforma favorita, así como una gran cantidad de artículos interesantes en la página web internacional de PGP [PGPI 2016]. Básicamente, el diseño de PGP es igual al mostrado en la Figura 8.21. Dependiendo de la versión, el software PGP utiliza MD5 o SHA para calcular el resumen del mensaje, CAST, triple-DES o IDEA para el cifrado de clave simétrica y RSA para el cifrado de clave pública.

Cuando se instala PGP, el software crea una pareja de clave pública y clave privada para el usuario. La clave pública puede darse a conocer a todo el mundo a través del sitio web del usuario o puede almacenarse en un servidor de clave pública. La clave privada está protegida mediante el uso de una contraseña. La contraseña debe introducirse cada vez que el usuario accede a la clave privada. PGP le da al usuario la opción de firmar digitalmente el mensaje, de cifrar el mensaje o de realizar tanto la operación de firma como la de cifrado. La Figura 8.22 muestra un mensaje firmado PGP. Este mensaje aparece después de la cabecera MIME. Los datos codificados en el mensaje son iguales a $K_A^-(H(m))$, es decir, son un resumen firmado digitalmente del mensaje. Como hemos comentado anteriormente, para que Benito verifique la integridad del mensaje necesitará tener acceso a la clave pública de Alicia.

La Figura 8.23 muestra un mensaje PGP secreto. Este mensaje también aparece después de la cabecera MIME. Por supuesto, el mensaje de texto en claro no está incluido en el mensaje secreto de correo electrónico. Cuando un emisor (como Alicia) quiere tanto confidencialidad como integridad, PGP contiene un mensaje como el de la Figura 8.23 dentro del mensaje de la Figura 8.22.

PGP también proporciona un mecanismo para la certificación de clave pública, pero el mecanismo es bastante distinto del sistema más convencional basado en autoridades de certificación. Las claves

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
Benito:
¿Puedo verte esta noche?
Apasionadamente tuya, Alicia
-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.0
Charset: noconv
yHJRHHGJGhgg/12EpJ+lo8gE4vB3mqJhFEvZP9t6n7G6m5Gw2
-----END PGP SIGNATURE-----
```

Figura 8.22 ♦ Un mensaje firmado mediante PGP.

```
-----BEGIN PGP MESSAGE-----
Version: PGP for Personal Privacy 5.0
u2R4d+/jKmn8Bc5+hgDsqAewsDfrGdszX68liKm5F6Gc4sDfcXyt
RfdS10juHgbcfDssWe7/K=1KhnMikLo0+1/BvcX4t==Ujk9PbcD4
Thdf2awQfgHbnmKlok8iy6gThlp
-----END PGP MESSAGE-----
```

Figura 8.23 ♦ Un mensaje PGP secreto.

públicas PGP están certificadas por una *red de confianza*. La propia Alicia puede certificar cualquier pareja de clave/nombre de usuario cuando crea que esa pareja es realmente correcta. Además, PGP permite que Alicia diga que confía en algún otro usuario a la hora de certificar la autenticidad de otras claves. Algunos otros usuarios PGP firman mutuamente sus claves manteniendo reuniones específicas de firma de claves. Los usuarios se reúnen físicamente, intercambian sus claves públicas y certifican las claves unos con otros firmándolas con sus claves privadas.

8.6 Asegurando las conexiones TCP: SSL

En la sección anterior hemos visto cómo las técnicas criptográficas pueden proporcionar confidencialidad, integridad de los datos y autenticación del punto terminal a una aplicación específica, que en nuestro caso concreto era el correo electrónico. En esta sección vamos a descender un nivel dentro de la pila de protocolos para examinar cómo se puede utilizar la criptografía para mejorar los servicios de seguridad de TCP, incluyendo la confidencialidad, la integridad de los datos y la autenticación del punto terminal. Esta versión mejorada de TCP se conoce comúnmente como **Capa de sockets seguros (SSL, Secure Sockets Layer)**. Una versión ligeramente modificada de SSL versión 3, denominada **Seguridad de la capa de transporte (TLS, Transport Layer Security)**, ha sido estandarizada por el IETF [RFC 4346].

El protocolo SSL fue diseñado originalmente por Netscape, pero las ideas básicas subyacentes para dotar de seguridad a TCP han terminado evolucionando a partir de los trabajos de Netscape (consulte, por ejemplo, Woo [Woo 1994]). Desde su invención, SSL ha disfrutado de una amplia implantación. SSL está soportado por todos los navegadores y servidores web más populares y es empleado por Gmail y la práctica totalidad de los sitios de comercio electrónico de Internet (incluyendo Amazon, eBay, y TaoBao). Anualmente se gastan cientos de miles de millones de euros a través de SSL. De hecho, si el lector ha realizado en alguna ocasión una compra por Internet con su tarjeta de crédito, la comunicación entre su navegador y el servidor para dicha compra tuvo lugar, casi con total seguridad, sobre SSL. (Puede verificar que su navegador está usando SSL viendo si el URL comienza por https: en lugar de por http:.)

Para entender la necesidad de SSL, analicemos un escenario de comercio electrónico típico por Internet. Imagine que Benito está navegando por la Web y que accede al sitio web de la empresa Alicia Incorporated, que se dedica a la venta de perfumes. El sitio Alicia Incorporated muestra un formulario en el que se supone que Benito debe indicar el tipo de perfume y la cantidad deseada, junto con su dirección y el número de su tarjeta de crédito. Benito introduce esta información, hace clic en el botón Enviar y espera recibir (por correo postal ordinario) los perfumes adquiridos; también espera recibir un cargo correspondiente a ese pedido en su siguiente extracto de la tarjeta de crédito. Todo esto suena muy bien, pero si no se adopta ninguna medida de seguridad Benito podría encontrarse con algunas sorpresas.

- Si no se utiliza ningún tipo de confidencialidad (cifrado), un intruso podría interceptar el pedido de Benito y obtener la información de su tarjeta de crédito. El intruso podría entonces realizar compras a expensas de Benito.
- Si no se utiliza ningún mecanismo que garantice la integridad de los datos, un intruso podría modificar el pedido de Benito, haciéndole comprar diez veces más frascos de perfume de los deseados.
- Por último, si no se utiliza ningún tipo de autenticación del servidor, cualquier servidor podría mostrar el famoso logotipo de Alicia Incorporated cuando en realidad nos encontráramos en un sitio mantenido por Gertrudis, que está haciendo pasar por Alicia Incorporated. Después de recibir el pedido de Benito, Gertrudis podría tomar el dinero de Benito y salir corriendo. O Gertrudis podría efectuar un robo de identidad, recopilando los datos relativos al nombre, la dirección y el número de tarjeta de crédito de Benito.

SSL solventa estos problemas mejorando TCP con servicios de confidencialidad, integridad de los datos, autenticación del servidor y autenticación del cliente.

SSL se emplea a menudo para proporcionar seguridad a las transacciones que tienen lugar a través de HTTP. Sin embargo, puesto que SSL dota de seguridad a TCP, puede ser empleado por cualquier aplicación que se ejecute sobre TCP. SSL proporciona una Interfaz de programación de aplicaciones (API, *Application Programmer Interface*) simple con sockets, que es similar y análoga a la API de TCP. Cuando una aplicación desea utilizar SSL, la aplicación incluye clases/bibliotecas SSL. Como se muestra en la Figura 8.24, aunque SSL reside técnicamente en la capa de aplicación, desde la perspectiva del desarrollador se trata de un protocolo de transporte que proporciona servicios TCP mejorados con servicios de seguridad.

8.6.1 Panorámica general

Comenzaremos describiendo una versión simplificada de SSL, una que nos va a permitir obtener una panorámica general del *por qué* y el *cómo* de SSL. Haremos referencia a esta versión simplificada de SSL como “casi-SSL”. Después de describir esta versión, en la siguiente subsección pasaremos a describir en detalle el protocolo SSL real. Tanto la versión casi-SSL como SSL se componen de tres fases: *acuerdo*, *deducción de la clave* y *transferencia de datos*. A continuación vamos a describir las tres fases de una sesión de comunicación entre un cliente (Benito) y un servidor (Alicia), teniendo Alicia una pareja de claves privada/pública y un certificado que asocia su identidad con su clave pública.

Fase de acuerdo

Durante la fase de acuerdo, Benito tiene que (a) establecer una conexión TCP con Alicia, (b) verificar que Alicia es *realmente* Alicia y (c) enviar a Alicia una clave secreta maestra, que ambos emplearán para generar todas las claves simétricas que necesiten para la sesión SSL. Estos tres pasos se muestran en la Figura 8.25. Observe que, una vez que se ha establecido la conexión TCP, Benito envía a Alicia un mensaje de saludo. A continuación, Alicia responde con su certificado, que contiene su clave pública. Como se ha explicado en la Sección 8.3, dado que el certificado ha sido emitido por una autoridad de certificación, Benito puede estar seguro de que la clave pública del certificado pertenece a Alicia. Benito genera entonces una clave maestra (MS, *Master Secret*), la cual solo utilizará para esta sesión SSL; cifra la MS con la clave pública de Alicia para crear la clave maestra cifrada (EMS, *Encrypted Master Secret*) y se la envía a Alicia. Alicia descifra la clave maestra cifrada con su clave privada para obtener la clave maestra (MS). Después de esta fase, tanto Benito como Alicia (y nadie más) conocen la clave maestra para esta sesión SSL.

Deducción de las claves

En principio, la MS, ahora compartida por Benito y Alicia, podría emplearse como la clave de sesión simétrica para todos los cífrados y comprobaciones de integridad de los datos subsiguientes. Sin

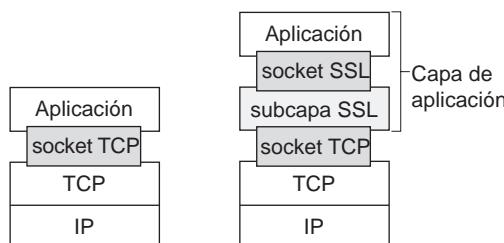


Figura 8.24 ♦ Aunque técnicamente SSL reside en la capa de aplicación, desde la perspectiva del desarrollador se trata de un protocolo de la capa de transporte.

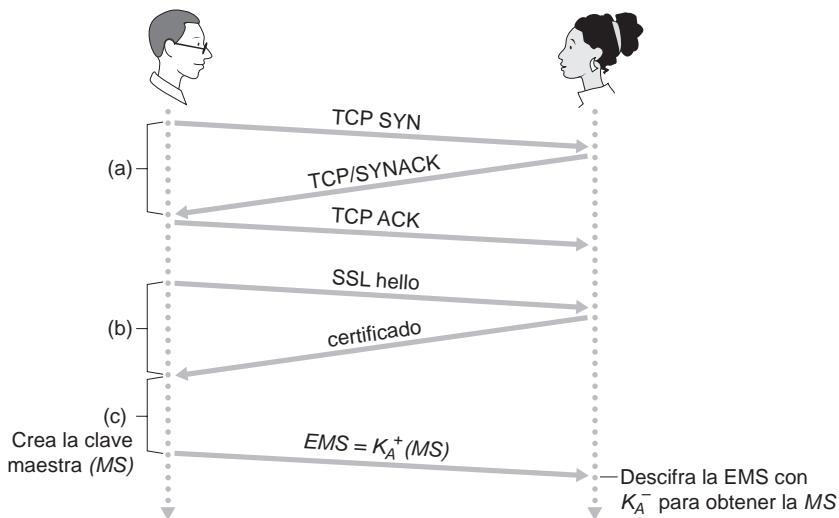


Figura 8.25 ♦ La fase de acuerdo de casi-SSL comienza con una conexión TCP.

embargo, generalmente, se considera más seguro que Alicia y Benito utilicen claves criptográficas distintas y también que empleen claves distintas para el cifrado y las comprobaciones de integridad. Por tanto, tanto Alicia como Benito utilizan la clave maestra para generar cuatro claves:

- E_B = clave de cifrado de sesión para los datos que Benito envía a Alicia.
- M_B = clave MAC de sesión para los datos que Benito envía a Alicia.
- E_A = clave de cifrado de sesión para los datos que Alicia envía a Benito.
- M_A = clave MAC de sesión para los datos que Alicia envía a Benito.

Tanto Alicia como Benito generan las cuatro claves a partir de la clave maestra (MS). Esto podría hacerse simplemente dividiendo la clave maestra en cuatro claves (pero, como veremos, esto es un poco más complicado en el protocolo SSL *real*). Al terminar la fase de deducción de claves, tanto Alicia como Benito disponen de las cuatro claves. Las dos claves de cifrado se utilizarán para cifrar los datos y las dos claves MAC se emplearán para verificar la integridad de los datos.

Transferencia de datos

Ahora que Alicia y Benito comparten las cuatro mismas claves de sesión (E_B , M_B , E_A y M_A) pueden comenzar a enviarse datos de forma segura a través de la conexión TCP. Puesto que TCP es un protocolo de flujos de bytes, una técnica natural sería que SSL cifrara los datos de aplicación sobre la marcha y luego pasara esos datos cifrados también sobre la marcha a TCP. Pero, si hacemos esto así, ¿dónde incluiríamos el valor MAC necesario para comprobar la integridad? Realmente, no es deseable tener que esperar a que termine la sesión TCP para verificar la integridad de todos los datos que Benito ha estado enviando a lo largo de la sesión completa. Para resolver este problema, SSL divide el flujo de datos en registros, añade un código MAC a cada registro para comprobar la integridad y luego cifra el registro junto con el código MAC. Para crear el valor MAC, Benito introduce los datos del registro y la clave M_B en una función hash, como hemos visto en la Sección 8.3. Para cifrar el paquete formado por el registro y el valor MAC, Benito utiliza su clave de cifrado de sesión E_B . Este paquete cifrado se pasa entonces a TCP para transportarlo a través de Internet.

Aunque este método permite resolver bastantes de los problemas, sigue sin ser perfecto en lo que se refiere a proporcionar integridad de los datos para todo el flujo de mensajes. En particular, suponga que Gertrudis es una intrusa que lleva a cabo un ataque por interposición y que tiene la

capacidad de insertar, borrar y sustituir segmentos en el flujo de segmentos TCP enviados entre Alicia y Benito. Gertrudis, por ejemplo, podría capturar dos segmentos enviados por Benito, invertir el orden de los mismos, ajustar los números de secuencia TCP (que no están cifrados) y luego enviar los dos segmentos en orden inverso a Alicia. Suponiendo que cada segmento TCP encapsula exactamente un registro, vamos a ver ahora cómo procesaría Alicia dichos segmentos.

1. El TCP que se ejecuta en Alicia pensará que todo es correcto y pasará los dos registros a la subcapa SSL.
2. SSL en Alicia descifrará los dos registros.
3. SSL en Alicia utilizaría la clave MAC en cada registro para verificar la integridad de los datos de los dos registros.
4. SSL pasaría a continuación los flujos de bytes descifrados de los dos registros a la capa de aplicación; pero el flujo de bytes completo recibido por Alicia no estaría en el orden correcto debido a la inversión del orden de los registros.

Animamos al lector a examinar escenarios similares para los casos en que Gertrudis elimine o reproduzca segmentos.

La solución a este problema, como probablemente ya habrá imaginado, consiste en utilizar números de secuencia. SSL hace esto de la forma siguiente: Benito mantiene un contador de número de secuencia, que se inicializa en cero y que se incrementa cada vez que envía un registro SSL. Benito no incluye realmente un número de secuencia en el propio registro, sino que cuando calcula el código MAC incluye el número secuencia en el cálculo del código MAC. Así, ahora el valor MAC es un hash de los datos más la clave MAC M_B más el número de secuencia actual. Alicia controla los números de secuencia de Benito, pudiendo verificar la integridad de los datos de un registro incluyendo el número de secuencia apropiado en el cálculo de MAC. Este uso de los números de secuencia SSL impide que Gertrudis lleve a cabo un ataque por interposición, tal como la reordenación o reproducción de segmentos. (¿Por qué?)

Registro SSL

En la Figura 8.26 se muestra el registro SSL (así como el registro casi-SSL). El registro consta de un campo de tipo, un campo de versión, un campo de longitud, un campo de datos y un campo MAC. Observe que los tres primeros campos no están cifrados. El campo de tipo indica si el registro es un mensaje de la fase de acuerdo o un mensaje que contiene datos de aplicación. También se utiliza para cerrar la conexión SSL, como explicamos más adelante. SSL en el terminal receptor utiliza el campo de longitud para extraer los registros SSL del flujo de bytes TCP entrante. El campo de versión se explica por sí mismo.

8.6.2 Una panorámica más completa

En la subsección anterior nos hemos ocupado del protocolo casi-SSL; esto ha servido para proporcionarnos un conocimiento básico acerca del por qué y del cómo de SSL. Ahora que ya tenemos una idea básica de SSL, podemos profundizar un poco y examinar los fundamentos del protocolo SSL real. En paralelo con la lectura de esta descripción del protocolo SSL, le animamos a que complete la práctica de laboratorio acerca de SSL con Wireshark, disponible en el sitio web del libro.

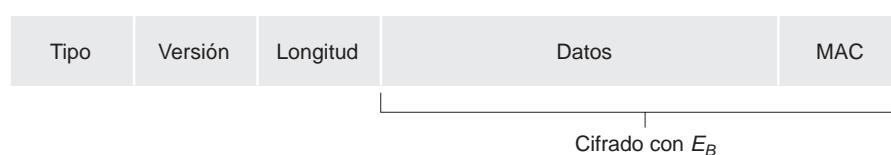


Figura 8.26 ♦ Formato de registro para SSL.

Fase de acuerdo de SSL

SSL no obliga a que Alicia y Benito utilicen un algoritmo de clave simétrica específico, un algoritmo de clave pública específico ni un código MAC específico. En su lugar, SSL permite que Alicia y Benito acuerden al principio de la sesión SSL, durante la fase de acuerdo, los algoritmos criptográficos que van a emplear. Además, durante la fase de acuerdo, Alicia y Benito se intercambian números distintivos, que se utilizan en la creación de las claves de sesión (E_B , M_B , E_A y M_A). Los pasos de la fase de acuerdo del protocolo SSL real son los siguientes:

1. El cliente envía la lista de algoritmos criptográficos que soporta, junto con un número distintivo de cliente.
2. A partir de la lista, el servidor elige un algoritmo simétrico (por ejemplo, AES), un algoritmo de clave pública (por ejemplo, RSA con una longitud específica de clave) y un algoritmo MAC. Devuelve al cliente las elecciones que ha hecho, así como un certificado y un número distintivo de servidor.
3. El cliente verifica el certificado, extrae la clave pública del servidor, genera una clave pre-maestra (PMS, *Pre-Master Secret*), cifra la PMS con la clave pública del servidor y envía la PMS cifrada al servidor.
4. Utilizando la misma función de deducción de clave (como la especificada por el estándar SSL), el cliente y el servidor calculan independientemente la clave maestra (MS) a partir de la PMS y de los números distintivos. La MS se divide entonces para generar las dos claves de cifrado y las dos claves MAC. Además, cuando el cifrado simétrico elegido emplea CBC (tal como 3DES o AES) también se obtienen a partir de la MS dos vectores de inicialización (IV), uno para cada lado de la conexión. A partir de este momento todos los mensajes intercambiados entre el cliente y el servidor son cifrados y autenticados (con un código MAC).
5. El cliente envía un código MAC de todos los mensajes de acuerdo.
6. El servidor envía un código MAC de todos los mensajes de acuerdo.

Los dos últimos pasos protegen el procedimiento de acuerdo frente a posibles modificaciones de los datos. Para comprobarlo, observe que en el paso 1 el cliente normalmente ofrece una lista de algoritmos (algunos de ellos fuertes y otros más débiles). Esta lista de algoritmos se envía como texto en claro, dado que todavía no se han acordado los algoritmos de cifrado y las claves. Gertrudis, como atacante interpuesto (*man-in-the-middle*), podría borrar los algoritmos más fuertes de la lista obligando al cliente a seleccionar un algoritmo débil. Para evitar un ataque por modificación, en el paso 5, el cliente envía un valor MAC de la concatenación de todos los mensajes de acuerdo que ha enviado y ha recibido. El servidor puede comparar dicho valor MAC con el valor MAC de los mensajes de acuerdo que haya recibido y enviado. Si existe alguna incoherencia, el servidor puede terminar la conexión. De forma similar, el servidor envía un mensaje MAC de los mensajes de acuerdo que ha visto, permitiendo al cliente detectar cualquier incoherencia.

El lector puede estar preguntándose por qué se introducen valores distintivos en los pasos 1 y 2. ¿No bastaría con utilizar números de secuencia para impedir los ataques por reproducción de segmentos? La respuesta es que sí, pero esos números de secuencia no impiden, por sí mismos, los “ataques por reproducción de la conexión”. Considere, por ejemplo, el siguiente ataque por reproducción de la conexión: suponga que Gertrudis capture todos los mensajes intercambiados entre Alicia y Benito. Al día siguiente, Gertrudis se hace pasar por Benito y envía a Alicia exactamente la misma secuencia de mensajes que Benito le envió el día anterior. Si Alicia no utiliza números distintivos responderá con exactamente la misma secuencia de mensajes que envió el día anterior. Alicia no sospechará nada raro, ya que cada mensaje que reciba pasará las comprobaciones de integridad. Si Alicia es un servidor de comercio electrónico, pensará que Benito está realizando un segundo pedido (solicitando exactamente los mismos artículos). Por otro lado, incluyendo un número distintivo en el protocolo Alicia enviará números distintivos diferentes en cada sesión TCP, haciendo que las claves de cifrado sean distintas en cada uno de los dos días. Por tanto, cuando Alicia reciba una serie de registros SSL reproducidos procedentes de Gertrudis, esos registros no pasarán las comprobaciones de integridad

y la transacción de comercio electrónico falsa no llegará a completarse. En resumen, en SSL los números distintivos se emplean para defenderse de los “ataques por reproducción de la conexión”, mientras que los números de secuencia se emplean para defenderse frente a la reproducción de paquetes individuales durante un sesión activa.

Cierre de la conexión

En algún momento, Benito o Alicia querrán terminar la sesión SSL. Una posible técnica consistiría en dejar que Benito terminara la sesión SSL simplemente terminando la conexión TCP subyacente; es decir, hacer que Benito envíe un segmento TCP FIN a Alicia. Pero ese diseño tan simplista abre la puerta a los *ataques de truncamiento* en los que Gertrudis se introduce de nuevo en mitad de una sesión SSL activa y termina la sesión prematuramente con un segmento TCP FIN. Si Gertrudis hiciera esto, Alicia pensaría que ha recibido todos los datos de Benito, cuando en realidad solo ha recibido una parte de los mismos. La solución a este problema consiste en indicar en el campo de tipo si el registro sirve para terminar la sesión SSL. (Aunque el tipo SSL se envía como texto en claro, siempre es autenticado en el receptor utilizando el valor MAC del registro.) Incluyendo dicho campo, si Alicia recibiera un segmento TCP FIN antes de recibir un registro SSL de cierre deduciría inmediatamente que algo raro está sucediendo.

Esto completa nuestra introducción a SSL. Hemos visto que esta tecnología utiliza muchos de los principios criptográficos explicados en las Secciones 8.2 y 8.3. Los lectores que deseen explorar SSL a un nivel más profundo pueden consultar el libro de Rescorla sobre SSL, que es bastante cómodo de leer [Rescorla 2001].

8.7 Seguridad de la capa de red: IPsec y redes privadas virtuales

El protocolo de seguridad IP, más conocido como **IPsec**, proporciona seguridad en la capa de red. IPsec proporciona seguridad a los datagramas IP intercambiados por cualesquiera dos entidades de la capa de red, incluyendo hosts y routers. Como enseñada veremos, muchas instituciones (corporaciones, agencias gubernamentales, organizaciones sin ánimo de lucro, etc.) utilizan IPsec para crear **redes privadas virtuales (VPN, Virtual Private Network)**, que funcionan sobre la red Internet pública.

Antes de entrar en los detalles específicos de IPsec, demos un paso atrás y consideremos qué es lo que implica proporcionar confidencialidad en la capa de red. Con la confidencialidad en la capa de red entre una pareja de entidades de red (por ejemplo, entre dos routers, entre dos hosts o entre un router y un host) la entidad emisora cifra las cargas útiles de todos los datagramas que envíe hacia la entidad receptora. La carga útil cifrada podría ser un segmento TCP, un segmento UDP, un mensaje ICMP, etc. Si dispusiéramos de tal servicio de la capa de red, todos los datos enviados de una entidad a la otra (incluyendo los mensajes de correo electrónico, las páginas web, los mensajes de acuerdo TCP y los mensajes de administración, como ICMP y SNMP) estarían ocultos a ojos de posibles terceros que pudieran estar husmeando los mensajes que circulan por la red. Por esta razón, decimos que la seguridad de la capa de red proporciona un servicio básico de “ocultación”.

Además de la confidencialidad, un protocolo de seguridad de la capa de red podría potencialmente proporcionar otros servicios de seguridad. Por ejemplo, podría ofrecer mecanismos de autenticación del origen de modo que la entidad receptora pueda verificar cuál es el origen del datagrama seguro. Un protocolo de seguridad de la capa de red podría proporcionar un servicio de integridad de los datos de modo que la entidad receptora pueda comprobar si se ha producido alguna alteración del datagrama mientras este se encontraba en tránsito. Un servicio de seguridad de la capa de red también podría proporcionar mecanismos para prevenir ataques por reproducción, lo que significa que Benito podría detectar cualquier datagrama duplicado que un atacante pudiera insertar. Como

pronto veremos, IPsec de hecho proporciona mecanismos para todos estos servicios de seguridad, es decir, para la confidencialidad, la autenticación de origen, la integridad de los datos y la prevención de los ataques por reproducción.

8.7.1 IPsec y redes privadas virtuales (VPN)

Normalmente, una institución que abarque múltiples regiones geográficas deseará disponer de su propia red IP, de modo que sus hosts y servidores puedan intercambiarse datos de forma segura y confidencial. Para conseguir este objetivo, esta institución podría implantar realmente una red física independiente (incluyendo routers, enlaces y una infraestructura DNS) que esté completamente separada de la red Internet pública. Dicha red separada, dedicada a una institución concreta, se denomina **red privada**. No es sorprendente que tales redes privadas puedan llegar a ser muy costosas, ya que la institución necesitará comprar, instalar y mantener su propia infraestructura física de red.

En lugar de implantar y mantener una red privada, muchas instituciones crean actualmente redes VPN sobre la red Internet pública existente. Con una VPN el tráfico entre sucursales se envía a través de la red Internet pública, en lugar de enviarse a través de una red físicamente independiente. Pero para proporcionar confidencialidad, el tráfico entre sucursales se cifra antes de entrar en la Internet pública. En la Figura 8.27 se muestra un ejemplo simple de red VPN. Aquí, la institución está compuesta por una oficina principal, una sucursal y una serie de vendedores itinerantes que suelen acceder a Internet desde la habitación de su hotel. (En la figura solo se muestra uno de esos vendedores.) En esta VPN, cuando dos hosts situados en la oficina principal se intercambian datagramas IP o cuando dos hosts de la sucursal quieren comunicarse utilizan el protocolo simple y tradicional IPv4 (es decir, sin servicios IPsec). Sin embargo, cuando dos hosts de la institución se comunican a través de una ruta que atraviesa la red Internet pública, el tráfico se cifra antes de entrar en Internet.

Para entender cómo funciona una red VPN, veamos un ejemplo simple en el contexto de la Figura 8.27. Cuando un host de la oficina principal envía un datagrama IP a un vendedor que

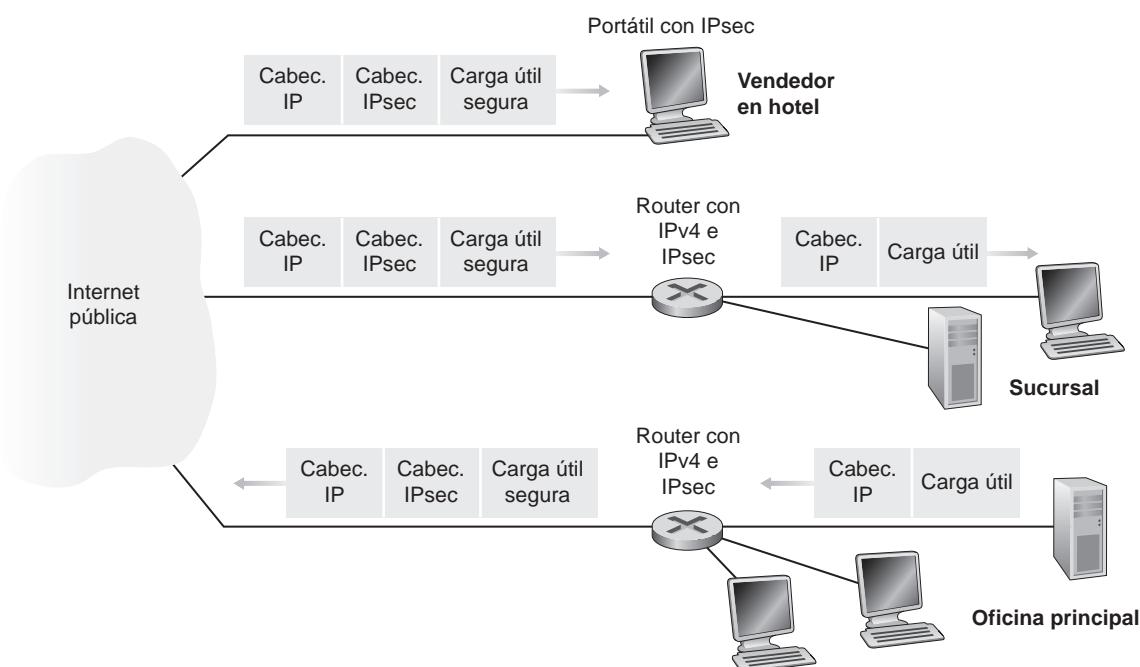


Figura 8.27 ♦ Red privada virtual (VPN).

se encuentra en un hotel, el router de pasarela de la oficina principal convierte el datagrama IPv4 simple en un datagrama IPsec y luego reenvía dicho datagrama IPsec hacia Internet. Este datagrama IPsec tiene de hecho una cabecera IPv4 tradicional, de modo que los routers de la red Internet pública procesan el datagrama como si se tratara de un datagrama IPv4 normal; para ellos el datagrama es, de hecho, como cualquier otro. Pero como se muestra en la Figura 8.27, la carga útil del datagrama IPsec incluye una cabecera IPsec, que es utilizada para el procesamiento IPsec; además, la carga útil del datagrama IPsec está cifrada. Cuando el datagrama IPsec llega al portátil del vendedor, el sistema operativo del equipo descifra la carga útil y proporciona algunos otros servicios de seguridad, como la verificación de la integridad de los datos, y pasa la carga útil descifrada hacia el protocolo de la capa superior (por ejemplo, hacia TCP o UDP).

Esto es solo una pequeña panorámica de cómo una institución podría utilizar IPsec para crear una red VPN. Para que los árboles no nos oculten el bosque, hemos dejado conscientemente de lado muchos detalles importantes. Realicemos ahora un examen más detallado.

8.7.2 Los protocolos AH y ESP

IPsec es un protocolo bastante complejo que está definido en más de una docena de documentos RFC. Dos documentos importantes son RFC 4301, que describe la arquitectura global de seguridad IP, y RFC 6071, que proporciona una panorámica de la serie de protocolos IPsec. Como siempre, nuestro objetivo en este libro de texto no es simplemente repetir los arcanos y áridos documentos RFC, sino más bien adoptar un enfoque más operativo y pedagógico a la hora de describir los protocolos.

En la serie de protocolos IPsec hay dos protocolos principales: el protocolo de **cabecera de autenticación (AH, Authentication Header)** y el protocolo de **carga útil de seguridad para encapsulación (ESP, Encapsulation Security Payload)**. Cuando una entidad IPsec de origen (normalmente un host o un router) envía datagramas seguros a una entidad de destino (también un host o un router) lo hace con el protocolo AH o el protocolo ESP. El protocolo AH proporciona autenticación del origen e integridad de los datos, pero *no* proporciona confidencialidad. El protocolo ESP proporciona autenticación del origen, integridad de los datos y confidencialidad. Puesto que la confidencialidad a menudo es crítica para las redes VPN y otras aplicaciones IPsec, el protocolo ESP se utiliza mucho más ampliamente que el protocolo AH. Con el fin de desmitificar IPsec y evitar buena parte de las complicaciones asociadas nos vamos por tanto a centrar exclusivamente en el protocolo ESP. Aquellos lectores que deseen también aprender los fundamentos del protocolo AH pueden explorar los documentos RFC y otros recursos en línea.

8.7.3 Asociaciones de seguridad

Los datagramas IPsec se intercambian entre parejas de entidades de red, como por ejemplo entre dos hosts, entre dos routers o entre un host y un router. Antes de enviar datagramas IPsec desde la entidad de origen a la de destino, ambas entidades crean una conexión lógica en la capa de red. Esta conexión lógica se denomina **asociación de seguridad (SA, Security Association)**. Una asociación de seguridad es una conexión lógica de tipo simple; es decir, una conexión unidireccional desde el origen al destino. Si ambas entidades desean enviarse datagramas seguros entre sí, entonces será necesario establecer dos SA (es decir, dos conexiones lógicas), una en cada dirección.

Por ejemplo, considere de nuevo la VPN institucional de la Figura 8.27. Esta institución consta de una oficina principal, una sucursal y un cierto número, por ejemplo, n , de vendedores itinerantes. Supongamos, como ejemplo, que existe tráfico IPsec bidireccional entre la oficina principal y la sucursal y entre la oficina principal y los vendedores. En esta VPN, ¿cuántas asociaciones de seguridad existirían? Para responder a esta cuestión, observe que hay dos SA entre el router de pasarela de la oficina principal y el router de pasarela de la sucursal (una en cada dirección); para la computadora portátil de cada vendedor también habrá dos SA entre el router de pasarela de la oficina principal y el portátil (de nuevo, una en cada dirección). Por tanto, en total, habrá $(2 + 2n)$

asociaciones de seguridad. Sin embargo, recuerde que no todo el tráfico enviado hacia Internet por los routers de pasarela o por las computadoras portátiles estará protegido mediante IPsec. Por ejemplo, un host situado en la oficina principal podría querer acceder a un servidor web (como Amazon o Google) disponible en la red Internet pública. Por tanto, el router de pasarela (y los portátiles) enviará hacia Internet tanto datagramas IPv4 normales como datagramas dotados de seguridad IPsec.

Tratemos ahora de examinar las interioridades de una asociación de seguridad. Para que las explicaciones sean tangibles y concretas vamos hacerlo en el contexto de una asociación de seguridad existente entre el router R1 y el router R2 de la Figura 8.28. (Podemos considerar que el router R1 es el router de pasarela de la oficina principal y que el router R2 es el router de pasarela de la sucursal de la Figura 8.27.) El router R1 mantendrá una cierta información de estado acerca de esta SA, la cual incluirá:

- Un identificador de 32 bits para la SA, denominado **Índice de parámetro de seguridad (SPI, Security Parameter Index)**.
- La interfaz de origen de la SA (en este caso, 200.168.1.100) y la interfaz de destino de la SA (en este caso 193.68.2.23).
- El tipo de cifrado que se va a utilizar (por ejemplo, 3DES con CBC).
- La clave de cifrado.
- El tipo de comprobación de integridad (por ejemplo, HMAC con MD5).
- La clave de autenticación.

Cada vez que el router R1 necesite construir un datagrama IPsec para reenviarlo a través de esta SA, accederá a esta información de estado para determinar cómo debe autenticar y cifrar el datagrama. De forma similar, el router R2 mantendrá la misma información de estado para esta SA y utilizará esta información para autenticar y descifrar todos los datagramas IPsec que lleguen desde dicha asociación de seguridad.

Cada entidad IPsec (router o host) suele mantener información de estado para muchas asociaciones de seguridad. Por ejemplo, en el ejemplo de la red VPN de la Figura 8.27 con n vendedores, el router de pasarela de la oficina principal mantiene información de estado para $(2 + 2n)$ asociaciones de seguridad. Cada entidad IPsec almacena la información de estado para todas sus asociaciones de seguridad en su **base de datos de asociaciones de seguridad (SAD, Security Association Database)**, que es una estructura de datos contenida en el kernel del sistema operativo de esa entidad.

8.7.4 El datagrama IPsec

Habiendo descrito las asociaciones de seguridad, podemos ahora describir la estructura real del datagrama IPsec. IPsec tiene dos formas distintas de paquete, una para el denominado **modo túnel**

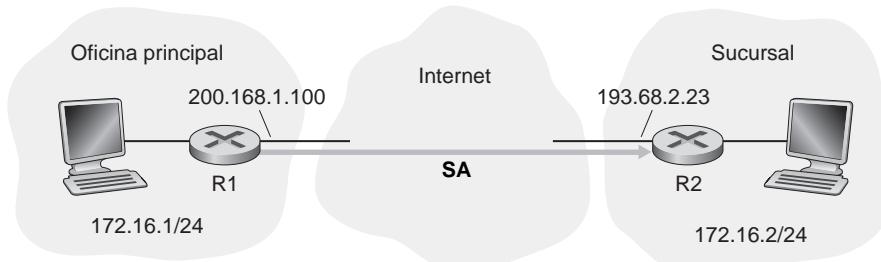


Figura 8.28 ♦ Asociación de seguridad (SA) de R1 a R2.

y otra para el denominado **modo transporte**. El modo túnel, al ser más apropiado para las redes VPN, está más ampliamente implantado que el modo transporte. Con el fin de desmitificar todavía más IPsec y evitar buena parte de los aspectos más complejos, nos vamos a centrar por tanto exclusivamente en el modo túnel. Una vez que tenga una sólida comprensión de dicho modo, el lector debería poder aprender por su cuenta los detalles acerca del modo transporte.

El formato de paquete del datagrama IPsec se muestra en la Figura 8.29. Puede que el lector crea que los formatos de paquete son aburridos e insípidos, pero pronto comprobará que el datagrama IPsec tiene en realidad la apariencia y el sabor de un manjar tex-mex. Examinemos los campos IPsec en el contexto de la Figura 8.28. Suponga que el router R1 recibe un datagrama IPv4 normal procedente del host 172.16.1.17 (situado en la red de la oficina principal) que está destinado al host 172.16.2.48 (situado en la red de la sucursal). El router R1 utiliza la siguiente receta para convertir este “datagrama IPv4 original” en un datagrama IPsec:

- Añade al final del datagrama IPv4 original (¡que incluye los campos originales de cabecera!) un campo de “cola ESP”.
- Cifra el resultado utilizando el algoritmo y la clave especificados por la asociación de seguridad.
- Añade al principio de este paquete cifrado un campo denominado “cabecera ESP”; el paquete resultante se conoce como “enchilada”.
- Crea un valor MAC de autenticación para *toda la enchilada* utilizando el algoritmo y la clave especificados en la SA.
- Añade el valor MAC al final de la enchilada formando así *la carga útil*.
- Por último, crea una nueva cabecera IP con todos los campos clásicos de la cabecera IPv4 (que suman normalmente 20 bytes de longitud) y añade dicha cabecera al principio de la carga útil.

Observe que el datagrama IPsec resultante es un datagrama IPv4 perfectamente normal, con los campos tradicionales de cabecera IPv4 seguidos de una carga útil. Pero en este caso la carga útil contiene una cabecera ESP, el datagrama IP original, una cola ESP y un campo de autenticación ESP (estando cifrados el datagrama original y la cola ESP). El datagrama IP original tiene el valor 172.16.1.17 como dirección IP de origen y el 172.16.2.48 como dirección IP de destino. Puesto que el datagrama IPsec incluye el datagrama IP original, estas direcciones se incluyen (y se cifran) como parte de la carga útil del paquete IPsec. ¿Pero qué sucede con las direcciones IP de origen y de destino contenidas en la nueva cabecera IP, es decir, en la cabecera situada más a la izquierda en el datagrama IPsec? Como cabría esperar, esos valores se configuran con las direcciones de las interfaces de router de origen y de destino situadas en los dos extremos de los túneles, es decir, con los valores 200.168.1.100 y 193.68.2.23. Asimismo, el número de protocolo en este nuevo campo de cabecera IPv4 no se configura con el valor correspondiente a TCP, UDP o SMTP, sino con el valor 50, que indica que se trata de un datagrama IPsec que está empleando el protocolo ESP.

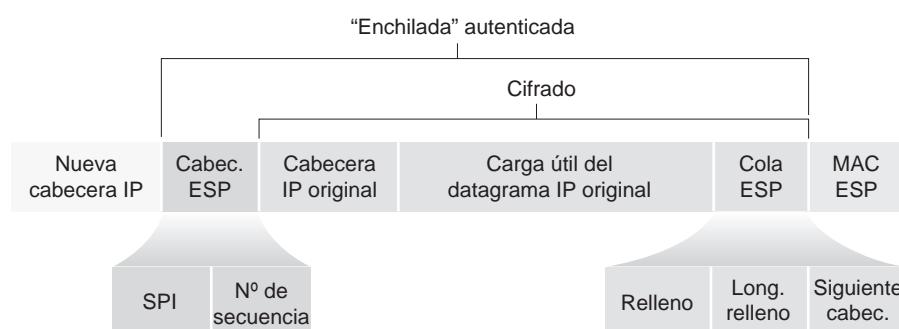


Figura 8.29 ♦ Formato del datagrama IPsec.

Después de que R1 envíe el datagrama IPsec hacia la red Internet pública, este pasará a través de muchos routers antes de alcanzar R2. Cada uno de estos routers procesará el datagrama como si fuera un datagrama normal; de hecho, todos esos routers no son conscientes de que el datagrama esté transportando datos cifrados mediante IPsec. Para estos routers de la red Internet pública, puesto que la dirección IP de destino contenida en la cabecera externa es R2, el destino último del datagrama es R2.

Habiendo examinado este ejemplo de cómo se construye un datagrama IPsec, veamos ahora con más detalle los ingredientes de la enchilada. Como podemos ver en la Figura 8.29, la cola ESP está compuesta por tres campos: relleno, longitud de relleno y siguiente cabecera. Recuerde que los sistemas de cifrado de bloque requieren que el mensaje que hay que cifrar sea un múltiplo entero de la longitud de bloque. Por ello se emplea un relleno (compuesto por bytes que no tienen ningún significado) para que, al añadirlo al datagrama original (junto con los campos de longitud de relleno y de siguiente cabecera), el “mensaje” resultante tenga un número entero de bloques. El campo de longitud de relleno indica a la entidad receptora cuánto relleno se ha insertado (y, por tanto, cuánto relleno habrá que eliminar). El campo de siguiente cabecera indica el tipo (por ejemplo, UDP) de los datos contenidos en el campo de datos de carga útil. Los datos de carga útil (normalmente, el datagrama IP original) y la cola ESP se concatenan y se cifran.

Delante de esta unidad cifrada se encuentra la cabecera ESP, que se envía como texto en claro y que consta de dos campos: el SPI y el campo de número de secuencia. El SPI indica a la entidad receptora cuál es la SA a la que pertenece el datagrama; la entidad receptora puede entonces indexar su base de datos SAD con el índice SPI para determinar los algoritmos y claves apropiados de autenticación/descifrado. El campo de número de secuencia se utiliza para defenderse frente a los ataques por reproducción.

La entidad emisora también añade un código MAC de autenticación. Como hemos dicho anteriormente, la entidad emisora calcula un código MAC para toda la enchilada (compuesta por la cabecera ESP, el datagrama IP original y la cola ESP, estando el datagrama y la cola cifrados). Recuerde que para calcular un valor MAC, el emisor añade una clave secreta MAC a la enchilada y luego calcula un valor hash de longitud fija para el resultado.

Cuando R2 recibe el datagrama IPsec, observa que la dirección IP de destino del datagrama es el propio R2, por lo que dicho router se encarga de procesar el datagrama. Puesto que el campo de protocolo (en la cabecera IP situada más a la izquierda) tiene el valor 50, R2 ve que debe aplicar el procesamiento ESP de IPsec al datagrama. En primer lugar, analizando la enchilada, R2 utiliza el SPI para determinar a qué asociación de seguridad (SA) pertenece el datagrama. En segundo lugar, calcula el valor MAC de la enchilada y verifica que es coherente con el valor contenido en el campo ESP MAC. Si lo es, el router sabrá que la enchilada procede del router R1 y que no ha sido manipulada. En tercer lugar, comprueba el campo de número de secuencia para verificar que el datagrama sea reciente y no un datagrama reproducido. En cuarto lugar, descifra la unidad cifrada utilizando la clave y el algoritmo de descifrado asociados con la SA. En quinto lugar, elimina el relleno y extrae el datagrama IP normal original. Y, finalmente, en sexto lugar, reenvía el datagrama original a la red de la sucursal para que el datagrama llegue a su verdadero destino. Es una receta un tanto complicada, ¿verdad? ¡Bueno, nunca dijimos que preparar una enchilada fuera fácil!

Existe todavía otra sutileza importante que necesitamos explicar y que está centrada en la siguiente cuestión: cuando el router R1 recibe un datagrama (no dotado de seguridad) procedente de un host de la red de la oficina principal y dicho datagrama está destinado a alguna dirección IP de destino situada fuera de la oficina principal, ¿cómo sabe R1 si ese datagrama debe ser convertido en un datagrama IPsec? Y si tiene que ser procesado por IPsec, ¿cómo sabe R1 qué SA (de las muchas asociaciones de seguridad existentes en su base de datos SAD) hay que utilizar para construir el datagrama IPsec? El problema se resuelve de la forma siguiente. Junto con una base de datos SAD, la entidad IPsec también mantiene otra estructura de datos denominada **base de datos de políticas de seguridad (SPD, Security Policy Database)**. La SPD indica qué tipos de datagramas (en función de la dirección IP de origen, la dirección IP de destino y el tipo de protocolo) hay que procesar mediante IPsec; y para aquellos que haya que procesar mediante IPsec, qué SA debe emplearse. En un cierto

sentido, la información de una SPD indica “qué” hacer con los datagramas que lleguen, mientras que la información de la SAD indica “cómo” hay que hacerlo.

Resumen de los servicios IPsec

¿Qué servicios proporciona IPsec exactamente? Examinemos estos servicios desde la perspectiva de un atacante, como por ejemplo Gertrudis, que se ha interpuesto (*man-in-the-middle*) en la comunicación, situándose en algún lugar de la ruta entre los routers R1 y R2 de la Figura 8.28. Vamos a suponer a lo largo de estas explicaciones que Gertrudis no conoce las claves de cifrado y de autenticación empleadas por la SA. ¿Qué cosas puede hacer Gertrudis y cuáles no? En primer lugar, Gertrudis no puede ver el datagrama original. De hecho, no solo están los datos del datagrama original ocultos a ojos de Gertrudis, sino que también lo están el número de protocolo, la dirección IP de origen y la dirección IP de destino. Para los datagramas enviados a través de la SA, Gertrudis solo sabe que el datagrama tiene su origen en algún host de la red 172.16.1.0/24 y que está destinado a algún host de la red 172.16.2.0/24. No sabe si está transportando datos TCP, UDP o ICMP; no sabe si está transportando HTTP, SMTP, o algún otro tipo de datos de aplicación. Esta confidencialidad, por tanto, va bastante más allá que en SSL. En segundo lugar, suponga que Gertrudis trata de alterar un datagrama en la SA modificando algunos de sus bits. Cuando este datagrama alterado llegue a R2 no pasará las comprobaciones de integridad (utilizando el valor MAC), desbaratando una vez más las intenciones de Gertrudis. En tercer lugar, suponga que Gertrudis intenta hacerse pasar por R1, creando un datagrama IPsec cuyo origen sea 200.168.1.100 y cuyo destino sea 193.68.2.23. El ataque de Gertrudis no tendrá ningún efecto, ya que este datagrama de nuevo no pasará la comprobación de integridad realizada en R2. Finalmente, puesto que IPsec incluye números de secuencia, Gertrudis no podrá desarrollar con éxito ningún ataque por reproducción. En resumen, y tal como dijimos al principio de esta sección, IPsec proporciona (entre cualquier pareja de dispositivos que procesen paquetes en la capa de red) mecanismos de confidencialidad, de autenticación del origen, de integridad de los datos y de prevención de los ataques por reproducción.

8.7.5 IKE: gestión de claves en IPsec

Cuando una red VPN tiene un pequeño número de puntos terminales (por ejemplo, solo dos routers, como en la Figura 8.28), el administrador de la red puede introducir manualmente la información de la SA (claves y algoritmos de cifrado/autenticación y los índices SPI) en las bases de datos SAD de los puntos terminales. Este tipo de “introducción manual” de las claves resulta obviamente poco práctico para una VPN de gran tamaño, que puede constar de centenares o incluso miles de hosts y routers IPsec. Las tareas de implantación de gran envergadura y geográficamente distribuidas requieren un mecanismo automatizado para la creación de las SA. IPsec lleva a cabo este tipo de tarea mediante el protocolo de Intercambio de claves de Internet (IKE, *Internet Key Exchange*), especificado en RFC 5996.

IKE presenta algunas similitudes con el procedimiento de acuerdo en SSL (véase la Sección 8.6). Cada entidad IPsec tiene un certificado, que incluye la clave pública de la entidad. Al igual que en SSL, el protocolo IKE exige que las dos entidades intercambien certificados, negocien los algoritmos de autenticación y cifrado e intercambien de modo seguro el material necesario para crear las claves de sesión para las SA de IPsec. A diferencia de SSL, IKE emplea dos fases para llevar a cabo estas tareas.

Vamos a analizar estas dos fases en el contexto de los routers R1 y R2 de la Figura 8.28. La primera fase está compuesta por dos intercambios de parejas de mensajes entre R1 y R2:

- Durante el primer intercambio de mensajes los dos lados utilizan Diffie-Hellman (véanse los problemas incluidos al final del capítulo) para crear una **IKE SA** bidireccional entre los routers. Para confundir a los estudiosos, esta SA IKE bidireccional es enteramente distinta de las SA IPsec presentadas en las Secciones 8.6.3 y 8.6.4. La IKE SA proporciona un canal autenticado y cifrado entre los dos routers. Durante este primer intercambio de parejas de mensajes se estable-

cen las claves de cifrado y autenticación para la IKE SA. También se establece un valor secreto maestro que se utilizará para calcular las claves IPsec SA posteriormente en la fase 2. Observe que durante este primer paso no se utilizan claves pública y privada RSA. En particular, ni R1 ni R2 revelan su identidad firmando un mensaje con su clave privada.

- Durante el segundo intercambio de mensajes ambos lados se revelan mutuamente su identidad, firmando sus mensajes. Sin embargo, las identidades no son reveladas a nadie que esté husmeando pasivamente el canal de comunicación, ya que los mensajes se envían a través del canal IKE SA seguro. También durante esta fase los dos lados negocian los algoritmos de cifrado y autenticación IPsec que serán empleados por las asociaciones de seguridad IPsec.

En la fase 2 de IKE los dos lados crean una SA en cada dirección. Al final de la fase 2 las claves de sesión para cifrado y autenticación habrán sido establecidas en ambos terminales para las dos SA. Los dos lados pueden emplear entonces las SA para enviar datagramas seguros, como se describe en las Secciones 8.7.3 y 8.7.4. La principal motivación para que existan dos fases en IKE es el coste computacional: puesto que la segunda fase no implica ningún tipo de criptografía de clave pública, IKE puede generar un gran número de asociaciones de seguridad entre las dos entidades IPsec con un coste de computación relativamente bajo.

8.8 Asegurando las redes LAN inalámbricas

La seguridad es una preocupación de particular importancia en las redes inalámbricas, en las que las ondas de radio que transportan la tramas pueden propagarse bastante más allá de los límites del edificio que alberga a los hosts y a la estación base inalámbrica. En esta sección vamos a presentar una breve introducción al tema de la seguridad inalámbrica. El lector interesado en ver un tratamiento más detallado puede leer el libro de Edney y Arbaugh [Edney 2003] que es bastante legible.

El problema de la seguridad en 802.11 ha atraído una considerable atención tanto en los círculos técnicos como en los medios de comunicación. Aunque ha habido mucha discusión en realidad se ha producido bastante poco debate, ya que parece existir un consenso universal en que la especificación 802.11 original contiene diversos fallos importantes de seguridad. De hecho, cualquiera puede descargarse actualmente software de dominio público que aprovecha los agujeros de seguridad, haciendo que los usuarios que emplean los mecanismos de seguridad simples de 802.11 sean tan vulnerables como aquellos usuarios que no emplean ninguna característica de seguridad en absoluto.

En la siguiente sección vamos a analizar los mecanismos de seguridad inicialmente estandarizados en la especificación 802.11, que se conocen colectivamente con el nombre de **privacidad equivalente a la del cable (WEP, Wired Equivalent Privacy)**. Como el nombre sugiere, WEP pretendía proporcionar un nivel de seguridad similar al que podemos encontrar en las redes cableadas. Posteriormente analizaremos algunos de los agujeros de seguridad de WEP y veremos el estándar 802.11i, que es una versión bastante más segura que 802.11 y que se adoptó en el año 2004.

8.8.1 WEP (Wired Equivalent Privacy)

El protocolo WEP del estándar IEEE 802.11 fue diseñado en 1999 para proporcionar autenticación y cifrado de datos entre un host y un punto de acceso inalámbrico (es decir, una estación base) utilizando una técnica basada en una clave simétrica compartida. WEP no especifica ningún algoritmo de gestión de claves, por lo que se presupone que el host y el punto de acceso inalámbrico han acordado de alguna manera qué clave utilizar, empleando para ello algún método fuera de banda. La autenticación se lleva a cabo de la forma siguiente:

1. Un host inalámbrico solicita la autenticación por parte de un punto de acceso.
2. El punto de acceso responde a la solicitud de autenticación con un número distintivo (*nonce*) de 128 bytes.

3. El host inalámbrico cifra el número distintivo utilizando la clave simétrica que comparte con el punto de acceso.
4. El punto de acceso descifra el número distintivo cifrado por el host.

Si el número distintivo descifrado se corresponde con el valor del número distintivo originalmente enviado al host, entonces el host quedará autenticado por el punto de acceso.

El algoritmo de cifrado de datos de WEP se ilustra en la Figura 8.30. Se supone que tanto el host como el punto de acceso conocen una clave simétrica secreta de 40 bits, K_S . Además, se añade un vector de inicialización (IV) de 24 bits a la clave de 40 bits para crear una clave de 64 bits que se usará para cifrar una única trama. El vector IV cambiará de una trama a la siguiente, por lo que cada trama será cifrada con una clave de 64 bits distinta. El cifrado se realiza de la manera siguiente. En primer lugar se calcula un valor CRC de 4 bytes (véase la Sección 6.2) para la carga útil de datos. La carga útil y los 4 bytes de CRC se cifran entonces utilizando el mecanismo de cifrado de flujo RC4. No vamos a analizar aquí los detalles de RC4 (puede encontrarlos en [Schneier 1995] y [Edney 2003]). Para nuestros propósitos, basta con saber que el algoritmo RC4, cuando se le suministra un valor de clave (en este caso, la clave (K_S , IV) de 64 bits), produce un flujo de valores de claves, $k_1^{IV}, k_2^{IV}, k_3^{IV}, \dots$ que se utilizan para cifrar los datos y el valor de CRC de una trama. A efectos prácticos, podemos considerar que estas operaciones se realizan de byte en byte. El cifrado se lleva a cabo combinando mediante XOR el i -ésimo byte de datos, d_i , con el i -ésimo valor de clave, k_i^{IV} , perteneciente al flujo de valores de clave generado por la pareja (K_S , IV); con ello se genera el i -ésimo byte de texto cifrado, c_i :

$$c_i = d_i \oplus k_i^{IV}$$

El valor del vector IV cambia de una trama a la siguiente y se incluye como *texto en claro* en la cabecera de cada trama 802.11 cifrada con WEP, como se muestra en la Figura 8.30. El receptor toma la clave simétrica secreta de 40 bits que comparte con el emisor, le añade el vector IV y emplea la clave de 64 bits resultante (que es idéntica a la clave usada por el emisor para realizar el cifrado) con el fin de descifrar la trama:

$$d_i = c_i \oplus k_i^{IV}$$

El uso apropiado del algoritmo RC4 requiere que *nunca* se utilice más de una vez un mismo valor de clave de 64 bits. Recuerde que la clave WEP cambia de una trama a otra. Para un valor K_S dado (que rara vez cambia, si es que cambia alguna vez), esto quiere decir que solo hay 2^{24} claves únicas. Si estas claves se seleccionan aleatoriamente, se puede demostrar [Edney 2003] que la probabilidad de haber elegido el mismo valor del vector IV (y por tanto de haber empleado la misma clave de 64 bits) dos veces es superior al 99 por ciento después de solo 12.000 tramas. Con un tamaño de trama de 1 kbyte y una velocidad de transmisión de 11 Mbps, bastan unos pocos segundos para transmitir 12.000 tramas. Además, puesto que el valor de IV se transmite como texto

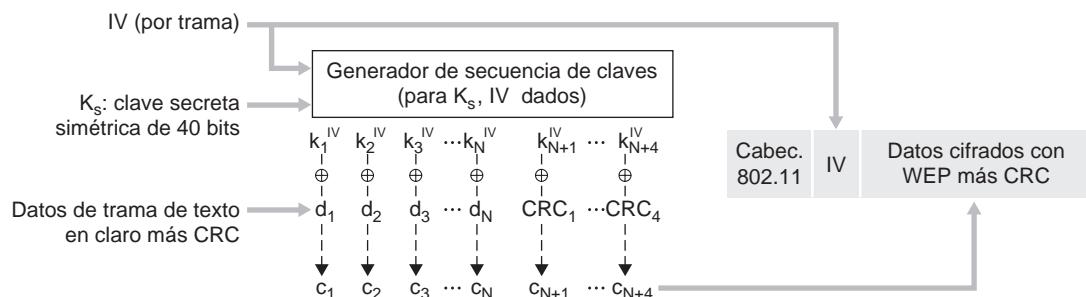


Figura 8.30 ♦ Protocolo WEP 802.11.

en claro dentro de la trama, un curioso que esté capturando la comunicación podrá saber si se ha utilizado un valor de IV duplicado.

Para comprender uno de los problemas que se presentan cuando se utiliza una clave duplicada, considere el siguiente ataque de texto en claro seleccionado realizado por Gertrudis contra Alicia. Suponga que Gertrudis (posiblemente utilizando alguna técnica de suplantación de dirección IP) envía una solicitud (por ejemplo, una solicitud HTTP o FTP) a Alicia para transmitir un archivo con un contenido conocido, $d_1, d_2, d_3, d_4, \dots$. Gertrudis también observa los datos cifrados $c_1, c_2, c_3, c_4, \dots$. Puesto que $d_i = c_i \oplus k_i^{\text{IV}}$, si combinamos mediante XOR c_i con cada lado de esta igualdad obtenemos

$$d_i \oplus c_i = k_i^{\text{IV}}$$

Con esta relación, Gertrudis puede utilizar los valores conocidos de d_i y c_i para calcular k_i^{IV} . La siguiente vez que Gertrudis vea que se está utilizando el mismo valor de IV conocerá la secuencia de clave $k_1^{\text{IV}}, k_2^{\text{IV}}, k_3^{\text{IV}}, \dots$ y podrá, por tanto, descifrar el mensaje cifrado.

Existen también varios otros problemas adicionales de seguridad que afectan a WEP. [Fluhrer 2001] describió un ataque que aprovecha una debilidad conocida de RC4 cuando se seleccionan ciertas claves débiles. [Stubblefield 2002] analiza varias formas eficientes de implementar y aprovechar este ataque. Otro problema relativo a WEP está relacionado con los bits CRC mostrados en la Figura 8.30 y transmitidos en la trama 802.11 para detectar posibles alteraciones de los bits de la carga útil. Sin embargo, un atacante que cambie el contenido cifrado (es decir, que sustituya los datos cifrados originales por bits sin sentido), que calcule el CRC para los bits sin sentido insertados y que coloque el CRC en una trama WEP puede generar una trama 802.11 que será aceptada por el receptor. Lo que hacen falta aquí son técnicas de integridad de mensajes como las que hemos estudiado en la Sección 8.3 para detectar la alteración o sustitución del contenido. Para conocer más detalles acerca de la seguridad en WEP, consulte [Edney 2003; Wright 2015] y las referencias en ellos contenidas.

8.8.2 IEEE 802.11i

Poco después de la publicación en 1999 de la norma IEEE 802.11 comenzó el trabajo para desarrollar una versión nueva y mejorada de 802.11 con mecanismos de seguridad más robustos. El nuevo estándar, conocido con el nombre de 802.11i, obtuvo la ratificación final en 2004. Como veremos, mientras que WEP proporcionaba un cifrado relativamente débil, junto con una única forma de llevar a cabo la autenticación y ningún mecanismo de distribución de claves, IEEE 802.11i proporciona formas mucho más fuertes de cifrado, un conjunto ampliable de mecanismos de autenticación y un mecanismo de distribución de claves. De aquí en adelante vamos a presentar una panorámica de 802.11i; puede encontrar una excelente presentación técnica (un flujo de audio) de 802.11i en [TechOnline 2012].

La Figura 8.31 presenta el marco conceptual de 802.11i. Además del punto de acceso y del cliente inalámbrico, 802.11i define un servidor de autenticación con el que el AP puede comunicarse. Separar el servidor de autenticación del AP permite que un mismo servidor de autenticación preste servicio a muchos puntos de acceso, centralizando las (a menudo sensibles) decisiones concernientes a la autenticación y al acceso dentro de ese único servidor y manteniendo a un nivel bajo el coste y la complejidad de los puntos de acceso. 802.11i opera en cuatro fases:

1. *Descubrimiento.* En la fase de descubrimiento el AP anuncia su presencia y las formas de autenticación y cifrado que puede proporcionar al nodo de cliente inalámbrico. El cliente solicita entonces las formas específicas de autenticación y cifrado que desea. Aunque el cliente y el AP ya están intercambiando mensajes, el cliente no habrá sido todavía autenticado ni dispondrá aún de una clave de cifrado, por lo que son necesarios varios pasos más antes de que el cliente pueda comunicarse con un host remoto arbitrario a través del canal inalámbrico.

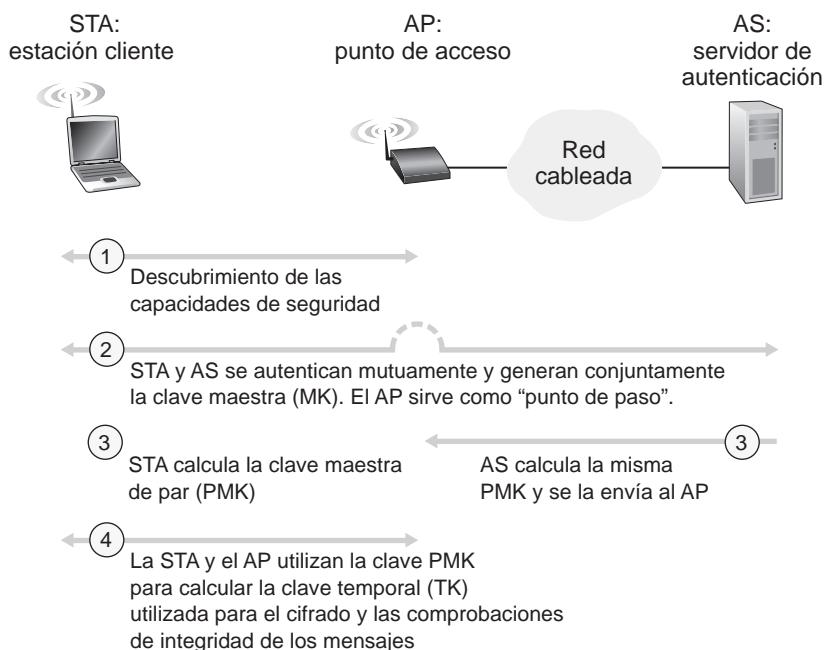


Figura 8.31 ♦ 802.11i: cuatro fases de operación.

2. *Autenticación mutua y generación de la clave maestra (MK, Master Key).* La autenticación tiene lugar entre el cliente inalámbrico y el servidor de autenticación. En esta fase, el punto de acceso actúa básicamente como repetidor, reenviando los mensajes entre el cliente y el servidor de autenticación. El **protocolo EAP (Extensible Authentication Protocol, Protocolo ampliable de autenticación)** [RFC 3748] define los formatos de los mensajes terminal a terminal utilizados en un modo de interacción simple de tipo solicitud/respuesta entre el cliente y el servidor de autenticación. Como se muestra en la Figura 8.32, los mensajes EAP se encapsulan utilizando **EAPoL** (EAP sobre LAN, [IEEE 802.1X]) y se envían a través del enlace inalámbrico 802.11. Estos mensajes EAP son entonces desencapsulados en el punto de acceso y luego re-encapsulados utilizando el protocolo **RADIUS** para su transmisión sobre UDP/IP hacia el servidor de autenticación. Aunque el protocolo [RFC 2865] y el servidor RADIUS no son requeridos obligatoriamente por el protocolo 802.11i, son componentes estándar *de facto* para 802.11i. Es posible que el protocolo **DIAMETER** [RFC 3588] recientemente estandarizado sustituya a RADIUS en un futuro próximo.
- Con EAP, el servidor de autenticación puede seleccionar una de varias formas para llevar a cabo la autenticación. Aunque 802.11i no impone un método de autenticación concreto, a menudo se emplea el esquema de autenticación EAP-TLS [RFC 5216]. EAP-TLS utiliza técnicas de clave pública (incluyendo cifrado con números distintivos y resúmenes de mensajes) similares a las que hemos estudiado en la Sección 8.3 para permitir que el cliente y el servidor de autenticación se autentiquen mutuamente entre sí y para calcular una clave maestra (MK) que será conocida por ambas partes.
3. *Generación de la clave maestra de par (PMK, Pairwise Master Key).* La MK es un secreto compartido que solo conocen el cliente y el servidor de autenticación y que ambos emplean para generar una segunda clave, la clave maestra de par (PMK). El servidor de autenticación envía entonces la PMK al AP. ¡Aquí es donde queríamos llegar! El cliente y el AP ahora disponen de una clave compartida (recuerde que, en WEP, el problema de la distribución de claves ni siquiera se contemplaba) y se habrán autenticado mutuamente entre sí. Ya están casi listos para poder comenzar con la operación real.

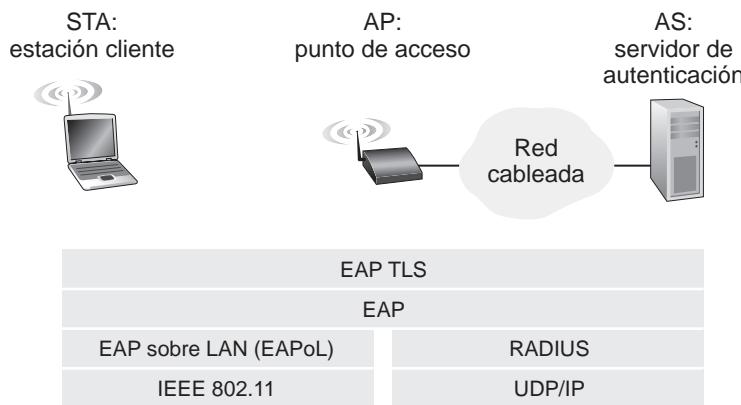


Figura 8.32 ♦ EAP es un protocolo extremo a extremo. Los mensajes EAP se encapsulan utilizando EAPoL sobre el enlace inalámbrico existente entre el cliente y el punto de acceso, y empleando RADIUS sobre UDP/IP entre el punto de acceso y el servidor de autenticación.

4. *Generación de la clave temporal (TK, Temporal Key).* Con la PMK, el cliente inalámbrico y el AP pueden ahora generar claves adicionales que se utilizarán para la comunicación. De particular interés es la clave temporal (TK) que se utilizará para realizar el cifrado de nivel de enlace de los datos enviados a través del enlace inalámbrico hacia un host remoto arbitrario.

802.11i proporciona varias formas de cifrado, incluyendo el esquema de cifrado basado en AES y una versión más fuerte del cifrado WEP.

8.9 Seguridad operacional: cortafuegos y sistemas de detección de intrusiones

Hemos visto a lo largo de este capítulo que Internet no es un lugar muy seguro (los malos están ahí fuera, infligiendo toda clase de estragos). Conocida la naturaleza hostil de Internet, vamos a considerar ahora la red de una organización y al administrador de red que la gestiona. Desde el punto de vista de un administrador de red, el mundo se divide de forma bastante nítida en dos bandos: los buenos (aquellos que pertenecen a la red de la organización y que deben poder acceder a los recursos internos de la misma de una forma relativamente poco restringida) y los malos (todos los demás, aquellos que deben ser cuidadosamente escrutados a la hora de acceder a los recursos de la red). En muchas organizaciones, desde los castillos medievales a los modernos edificios de oficinas, existe un único punto de entrada/salida donde se hace una comprobación de seguridad tanto de los buenos como de los malos que entran y salen de la organización. En un castillo esto se hacía en la puerta situada en el extremo de un puente levadizo; en un edificio de oficinas, esto se hace en el control de seguridad de entrada. En una red de computadoras, cuando se comprueba si el tráfico que entra/sale de la red es seguro, cuando se registra ese tráfico y cuando se elimina o reenvía, quienes se encargan de esas tareas son una serie de dispositivos operacionales conocidos como cortafuegos (*firewalls*), sistemas de detección de intrusiones (*IDS, Intrusion Detection System*) y sistemas de prevención de intrusiones (*IPS, Intrusion Prevention System*).

8.9.1 Cortafuegos

Un **cortafuegos** es una combinación de hardware y software que aísla la red interna de la organización de Internet, permitiendo pasar a algunos paquetes y bloqueando a otros. Un cortafuegos permite

a un administrador de red controlar el acceso entre el mundo exterior y los recursos dentro de la red administrada, gestionando el flujo de tráfico hacia y desde esos recursos. Un cortafuegos tiene tres objetivos:

- *Todo el tráfico que va del exterior hacia el interior de la red, y viceversa, pasa a través del cortafuegos.* La Figura 8.33 muestra un cortafuegos, que se encuentra en el límite entre la red administrada y el resto de Internet. Aunque las organizaciones de gran tamaño pueden utilizar varios niveles de cortafuegos o cortafuegos distribuidos [Skoudis 2006], colocar un cortafuegos en un único punto de acceso a la red, como se muestra en la Figura 8.33, facilita la gestión y el imponer una política de control de acceso.
- *Solo se permite el paso del tráfico autorizado de acuerdo con la política de seguridad local.* Con todo el tráfico de entrada y de salida de la red institucional pasando a través del cortafuegos, este puede restringir el acceso al tráfico autorizado.
- *El propio cortafuegos es inmune a la penetración.* El propio cortafuegos es un dispositivo conectado a la red. Si no está diseñado o instalado apropiadamente puede verse comprometido, en cuyo caso solo proporciona una falsa sensación de seguridad (lo que es peor que no disponer de cortafuegos).

Cisco y Check Point son dos de las empresas líderes actuales de distribución de cortafuegos. También puede crearse fácilmente un cortafuegos (filtro de paquetes) a partir de una máquina Linux utilizando iptables (software de dominio público, que se suministra habitualmente con Linux). Además, como hemos visto en los Capítulos 4 y 5, ahora los cortafuegos se implementan frecuentemente en los routers y se controlan de forma remota mediante SDN.

Los cortafuegos se pueden clasificar en tres categorías: **filtros de paquetes tradicionales**, **filtros con memoria del estado** y **pasarelas de aplicación**. En las siguientes subsecciones abordaremos cada uno de estos tipos de cortafuegos.

Filtros de paquetes tradicionales

Como se muestra en la Figura 8.33, una organización dispone normalmente de un router de pasarela que conecta su red interna con su ISP (y por tanto con la gran red Internet pública). Todo el tráfico que sale y entra en la red interna pasa a través de este router y es en este router donde tiene lugar el **filtrado de paquetes**. Un filtro de paquetes examina cada datagrama aisladamente, determinando si debe pasar o debe ser eliminado basándose en las reglas especificadas por el administrador. Las decisiones de filtrado normalmente se basan en:

- Las direcciones IP de origen o de destino.
- El tipo de protocolo especificado en el campo de datagrama IP: TCP, UDP, ICMP, OSPF, etc.
- El puerto de destino y de origen TCP o UDP.
- Los bits indicadores TCP: SYN, ACK, etc.
- El tipo de mensaje ICMP.
- Diferentes reglas para los datagramas que salen y entran en la red.
- Diferentes reglas para las distintas interfaces del router.

Un administrador de red configura el cortafuegos basándose en la política de la organización. La política puede tener en cuenta la productividad del usuario y el uso del ancho de banda, así como los problemas de seguridad de una organización. La Tabla 8.5 enumera algunas de las posibles políticas que una organización puede tener y cómo podrían controlarse mediante un sistema de filtrado de paquetes. Por ejemplo, si la organización no desea permitir ninguna conexión TCP entrante excepto aquellas destinadas a su servidor web público, puede bloquear todos los segmentos TCP SYN entrantes, salvo aquellos cuyo puerto de destino sea el puerto 80 y cuya dirección IP de destino

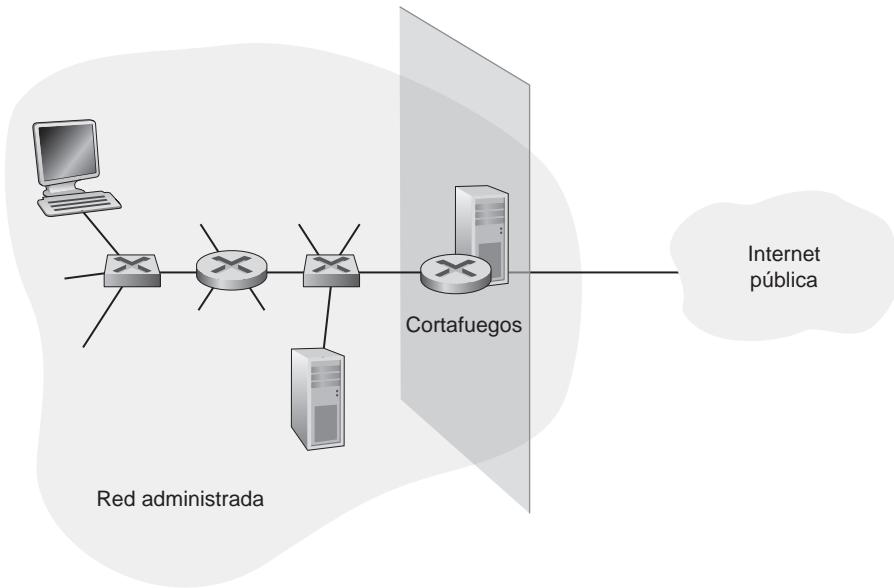


Figura 8.33 ♦ Colocación de un cortafuegos entre la red administrada y el mundo exterior.

sea la correspondiente al servidor web. Si la organización no desea que sus usuarios monopolicen el ancho de banda de acceso con aplicaciones de radio por Internet, puede bloquear todo el tráfico UDP no crítico (ya que la radio por Internet a menudo se transmite sobre UDP). Si la organización no desea que su red interna sea analizada (con Traceroute) por alguien externo, puede bloquear todos los mensajes ICMP TTL caducados que salen de la red de la organización.

Una política de filtrado puede estar basada en una combinación de direcciones y números de puerto. Por ejemplo, un router de filtrado podría reenviar todos los datagramas Telnet (aquejlos con un número de puerto igual a 23) excepto aquellos que se dirigen y proceden de una dirección incluida en una lista de direcciones IP especificadas. Esta política permite las conexiones Telnet dirigidas y procedentes de los hosts especificados en la lista de direcciones permitidas. Lamentablemente, basar la política en direcciones externas no proporciona ninguna protección frente a datagramas en los que su dirección de origen haya sido falsificada.

Política	Configuración del cortafuegos
Sin acceso web externo.	Eliminar todos los paquetes salientes hacia cualquier dirección IP, puerto 80.
Sin conexiones TCP entrantes, excepto las destinadas al servidor web público de la organización.	Eliminar todos los paquetes TCP SYN entrantes hacia cualquier IP excepto 130.207.244.203, puerto 80.
Impedir que las aplicaciones de radio web consuman el ancho de banda disponible.	Eliminar todos los paquetes UDP entrantes, excepto los paquetes DNS.
Impedir que la red sea utilizada para llevar a cabo un ataque DoS distribuido.	Eliminar todos los paquetes ping ICMP hacia una dirección de "difusión" (por ejemplo, 130.207.255.255).
Impedir que la red sea examinada con Traceroute.	Eliminar todo el tráfico ICMP TTL saliente caducado.

Tabla 8.5 ♦ Políticas y reglas de filtrado correspondientes de la red 130.207/16 de una organización con un servidor web en 130.207.244.203

El filtrado también puede basarse en si se ha configurado o no el bit TCP ACK. Este truco resulta bastante útil si una organización desea permitir que sus clientes internos se conecten a servidores externos, pero quiere impedir que clientes externos se conecten a los servidores internos. Recuerde de la Sección 3.5 que el primer segmento de toda conexión TCP tiene puesto a 0 el bit ACK, mientras que los restantes segmentos de la conexión tienen puesto a 1 dicho bit. Por tanto, si una organización desea impedir que los clientes externos inicien conexiones con los servidores internos, basta con filtrar todos los segmentos entrantes que tengan el bit ACK puesto a 0. Esta política prohíbe todas las conexiones TCP que tienen su origen en el exterior, pero permite las conexiones originadas internamente.

Las reglas de cortafuegos se implementan en los routers mediante listas de control de acceso, teniendo cada interfaz del router su propia lista. En la Tabla 8.6 se muestra un ejemplo de una lista de control de acceso para una organización 222.22/16. Esta lista de control de acceso es para una interfaz que conecta el router con los ISP externos a la organización. Las reglas se aplican a cada datagrama que atraviesa la interfaz de arriba a abajo. Las dos primeras reglas juntas permiten a los usuarios internos navegar por la Web: la primera regla permite salir de la red de la organización a cualquier paquete TCP con el puerto de destino 80; la segunda regla permite entrar en la red de la organización a cualquier paquete TCP que tenga el puerto de origen 80 y el bit ACK activado. Observe que si un origen externo intenta establecer una conexión TCP con un host interno la conexión será bloqueada, incluso aunque el puerto de origen o el de destino sea el puerto 80. Las dos reglas siguientes juntas permiten a los paquetes DNS entrar y salir de la red de la organización. En resumen, esta lista de control de acceso bastante restrictiva bloquea todo el tráfico excepto el tráfico web iniciado dentro de la organización y el tráfico DNS. [CERT Filtering 2012] proporciona una lista de filtros de paquetes basados en puertos/protocolos recomendados para evitar una serie de agujeros de seguridad bien conocidos en las aplicaciones de red existentes.

Filtros de paquetes con memoria del estado

En un filtro de paquetes tradicional las decisiones de filtrado se toman para cada paquete de forma aislada. Los filtros con memoria del estado realmente controlan las conexiones TCP y utilizan dicha información para tomar las decisiones de filtrado.

Para entender los filtros con memoria del estado, examinemos de nuevo la lista de control de acceso de la Tabla 8.6. Aunque bastante restrictiva, la lista de control de acceso de esta tabla permite que cualquier paquete procedente del exterior con ACK = 1 y puerto de origen 80 atraviese el filtro. Tales paquetes podrían ser utilizados por posibles atacantes para intentar hacer fallar a los sistemas internos con paquetes mal formados, llevar a cabo ataques de denegación de servicio o realizar un mapa de la red interna. La solución más sencilla consiste en bloquear también los paquetes TCPACK, pero este método impediría a los usuarios internos de la organización navegar por la Web.

Acción	Dirección de origen	Dirección de destino	Protocolo	Puerto de origen	Puerto de destino	Bit indicador
Permitir	222.22/16	Fuera de 222.22/16	TCP	> 1023	80	Cualquiera
Permitir	Fuera de 222.22/16	222.22/16	TCP	80	> 1023	ACK
Permitir	222.22/16	Fuera de 222.22/16	UDP	> 1023	53	—
Permitir	Fuera de 222.22/16	222.22/16	UDP	53	> 1023	—
Denegar	Todos	Todos	Todos	Todos	Todos	Todos

Tabla 8.6 ▶ Lista de control de acceso para una interfaz de router.

Los filtros con memoria del estado resuelven este problema almacenando la información de todas las conexiones TCP activas en una tabla de conexiones. Esto es posible porque el cortafuegos puede observar el inicio de una nueva conexión observando un acuerdo en tres fases (SYN, SYNACK y ACK) y puede observar el fin de una conexión cuando ve un paquete FIN para la conexión. El cortafuegos también puede suponer (de forma conservadora) que la conexión ha terminado cuando no ha observado ninguna actividad en la misma durante, por ejemplo, 60 segundos. En la Tabla 8.7 se muestra un ejemplo de una tabla de conexiones para un cortafuegos. Esta tabla de conexiones indica que actualmente hay tres conexiones TCP activas, habiéndose iniciado todas ellas dentro de la organización. Adicionalmente, el filtro con memoria del estado incluye una nueva columna, “Comprobar conexión”, en su lista de control de acceso, como se muestra en la Tabla 8.8. Observe que esta tabla es idéntica a la lista de control de acceso de la Tabla 8.6, excepto en que ahora indica que la conexión debería ser comprobada para dos de las reglas.

Veamos algunos ejemplos para examinar cómo trabajan conjuntamente la tabla de conexiones y la lista de control de acceso ampliada. Suponga que un atacante intenta introducir un paquete mal formado en la red de la organización enviando un datagrama con el puerto de origen TCP número 80 y con el bit indicador ACK activado. Suponga además que este paquete tiene el número de puerto de origen 12543 y la dirección IP de origen 150.23.23.155. Cuando este paquete llega al cortafuegos, este comprueba la lista de control de acceso de la Tabla 8.7, la cual indica que la tabla de conexiones también tiene que ser comprobada antes de permitir la entrada a este paquete en la red de la organización. El cortafuegos comprueba debidamente la tabla de conexiones, ve que ese paquete no es parte de una conexión TCP activa y lo rechaza. Como segundo ejemplo, suponga que un usuario interno desea navegar por un sitio web externo. Puesto que este usuario en primer lugar envía un segmento TCP SYN, la conexión TCP del usuario se registra en la tabla de conexiones. Cuando el servidor web devuelve los paquetes (con el bit ACK necesariamente activado), el cortafuegos comprueba la tabla y ve que la correspondiente conexión está en curso. Por tanto, el cortafuegos deja pasar a estos paquetes, no interviniendo entonces en la actividad de navegación por la Web del usuario interno.

Dirección de origen	Dirección de destino	Puerto de origen	Puerto de destino
222.22.1.7	37.96.87.123	12699	80
222.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

Tabla 8.7 ♦ Tabla de conexiones de un filtro con memoria del estado.

Acción	Dirección de origen	Dirección de destino	Protocolo	Puerto de origen	Puerto de destino	Bit indicador	Comprobar conexión
Permitir	222.22/16	Fuera de 222.22/16	TCP	> 1023	80	any	
Permitir	Fuera de 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
Permitir	222.22/16	Fuera de 222.22/16	UDP	> 1023	53	—	
Permitir	Fuera de 222.22/16	222.22/16	UDP	53	> 1023	—	X
Denegar	Todos	Todos	Todos	Todos	Todos	Todos	

Tabla 8.8 ♦ Lista de control de acceso para un filtro con memoria del estado.

Pasarela de aplicación

En los ejemplos anteriores hemos visto que el filtrado en el nivel de paquetes permite a una organización realizar un filtrado báscio, basándose en los contenidos de las cabeceras IP y TCP/UDP, incluyendo las direcciones IP, los números de puerto y los bits de reconocimiento (ACK). Pero, ¿qué ocurre si una organización desea proporcionar un servicio Telnet a un conjunto restringido de usuarios internos (en oposición a direcciones IP)? ¿Y qué ocurre si la organización desea que tales usuarios privilegiados se autentiquen a sí mismos antes de permitirles establecer conexiones Telnet con el mundo exterior? Tales tareas quedan fuera de las capacidades de los filtros tradicionales y con memoria del estado. De hecho, la información acerca de la identidad de los usuarios internos está en los datos de la capa de aplicación y no está incluida en las cabeceras IP/TCP/UDP.

Para conseguir seguridad con un nivel de granularidad más fino, los cortafuegos deben combinar los filtros de paquetes con pasarelas de aplicación. Las pasarelas de aplicación miran más allá de la cabeceras IP/TCP/UDP y toman sus decisiones basándose en los datos de aplicación. Una **pasarela de aplicación** es un servidor específico de aplicación a través del cual deben pasar todos los datos de aplicación (entrantes y salientes). Pueden ejecutarse varias aplicaciones de pasarela sobre el mismo host, pero cada pasarela es un servidor separado con sus propios procesos.

Con el fin de proporcionar algunas nociones sobre las pasarelas de aplicación, vamos a diseñar un cortafuegos que permita solo a un grupo restringido de usuarios internos establecer una conexión Telnet con el exterior y que impida a todos los clientes externos establecer una conexión Telnet con la red interna. Tal política se puede conseguir implementando una combinación de un filtro de paquetes (en un router) y una pasarela de aplicación Telnet, como se muestra en la Figura 8.34. El filtro del router está configurado para bloquear todas las conexiones Telnet excepto aquellas que tienen su origen en la dirección IP de la pasarela de aplicación. Una configuración de filtro como esta fuerza a todas las conexiones Telnet salientes a pasar a través de la pasarela de aplicación. Considere ahora un usuario interno que quiera conectarse mediante Telnet con el mundo exterior. El usuario tiene que establecer en primer lugar una sesión Telnet con la pasarela de aplicación. Una aplicación que se ejecute en la pasarela y que está a la escucha de sesiones Telnet entrantes pedirá al usuario que introduzca un ID de usuario y una contraseña. Cuando el usuario proporcione esta información, la pasarela de aplicación la comprobará para ver si el usuario tiene permiso para

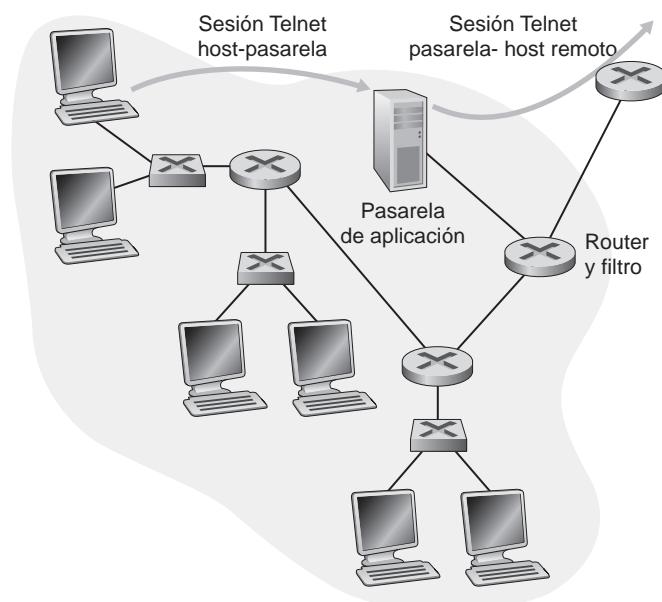


Figura 8.34 ♦ Cortafuegos que consta de una pasarela de aplicación y un filtro.

establecer una conexión Telnet con el mundo exterior. Si no es así, la pasarela termina la conexión Telnet que tiene con el usuario interno. Si el usuario tiene permiso, entonces la pasarela (1) pide al usuario el nombre del host externo con el que desea conectarse, (2) establece una sesión Telnet entre la pasarela y el host externo y (3) reenvía al host externo todos los datos que lleguen del usuario, de la misma manera que reenvía el usuario todos los datos que lleguen desde el host externo. De este modo, la pasarela de aplicación Telnet no solo se encarga de realizar la autorización del usuario, sino que actúa como servidor y cliente Telnet, retransmitiendo la información entre el usuario y el servidor Telnet remoto. Observe que el filtro permitirá que se lleve a cabo el paso 2, dado que es la pasarela quien inicia la conexión Telnet con el mundo exterior.

Las redes internas a menudo disponen de múltiples pasarelas de aplicación, por ejemplo, pasarelas para Telnet, HTTP, FTP y correo electrónico. De hecho, el servidor de correo de una organización (véase la Sección 2.3) y la caché web son pasarelas de aplicación.

Pero las pasarelas de aplicación también tienen sus desventajas. En primer lugar, hace falta una pasarela de aplicación diferente para cada aplicación. En segundo lugar, el rendimiento se verá afectado, dado que todos los datos tendrán que ser reenviados a través de la pasarela. Esto

HISTORIA

ANONIMATO Y PRIVACIDAD

Suponga que deseamos visitar un sitio web controvertido (por ejemplo, un sitio de activismo político) y que (1) no queremos revelar nuestra dirección IP al sitio web, (2) no queremos que el ISP local (que podría ser el ISP de nuestro domicilio o empresa) sepa que estamos visitando dicho sitio y (3) no deseamos que el ISP local vea los datos que estamos intercambiando con el sitio. Si se emplea el método tradicional de conexión directa con el sitio web sin ningún tipo de cifrado no será posible conseguir ninguno de los tres objetivos anteriores. Incluso utilizando SSL los dos primeros objetivos serán inalcanzables: la dirección IP de origen será presentada al sitio web en cada datagrama que envíemos y la dirección de destino de cada paquete que envíemos podrá ser fácilmente husmeada por el ISP local.

Para conseguir la confidencialidad y el anonimato, podemos en su lugar utilizar una combinación de servidor proxy de confianza y SSL, como se muestra en la Figura 8.35. Con esta técnica, primero establecemos una conexión SSL con el proxy de confianza. Después enviamos a través de esta conexión SSL una solicitud HTTP de una página situada en el sitio web deseado. Cuando el proxy reciba la solicitud HTTP cifrada mediante SSL la descifrará y reenviará la solicitud HTTP de texto en claro al sitio web. El sitio web responderá entonces al proxy que a su vez nos reenviará la respuesta a través de SSL. Puesto que el sitio web solo ve la dirección IP del proxy, y no la dirección de nuestro cliente, estaremos obteniendo un acceso verdaderamente anónimo al sitio web. Y como todo el tráfico entre nosotros y el proxy está cifrado, nuestro ISP local no puede invadir nuestra intimidad almacenando en un registro el nombre del sitio que hemos visitado o los datos que estamos intercambiando. Muchas empresas ofrecen en la actualidad tales servicios proxy (como por ejemplo proxify.com).

Por supuesto, con esta solución, nuestro proxy conoce todos los datos: conoce nuestra dirección IP y la dirección IP del sitio que estamos navegando; y puede ver como texto en claro todo el tráfico intercambiado entre nosotros y el sitio web. Dicha solución, por tanto, solo será buena si nuestra confianza en el proxy es alta. Un enfoque más robusto adoptado por el servicio de anonimato y privacidad TOR consiste en enrutar el tráfico a través de una serie de servidores proxy no confabulados [TOR 2016]. En particular, TOR permite a los usuarios sugerir nuevos proxies para su lista. Cuando un usuario se conecta a un servidor utilizando TOR, este servicio selecciona aleatoriamente (de entre su lista de proxies) una cadena de tres proxies y enruta todo el tráfico entre el cliente y el servidor a través de esa cadena. De esta forma, suponiendo que los proxies no estén confabulados, nadie sabe que ha tenido lugar una comunicación entre nuestra dirección IP y el sitio web objetivo. Además, aunque la comunicación se realice como texto en claro entre el último proxy y el servidor, el último proxy no sabe qué dirección IP está enviando y recibiendo ese texto en claro.

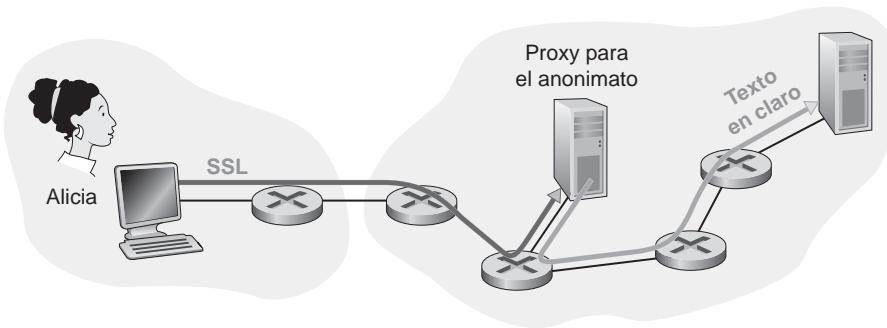


Figura 8.35 ♦ Cómo proporcionar anonimato y privacidad con un proxy.

se convierte en un problema particularmente en aquellos casos en los que hay varios usuarios o aplicaciones utilizando la misma máquina pasarela. Finalmente, el software de cliente debe saber cómo contactar con la pasarela cuando el usuario realiza una solicitud, y debe saber también cómo decir a la pasarela de aplicación con qué servidor externo hay que conectarse.

8.9.2 Sistemas de detección de intrusiones

Ya hemos visto que un filtro de paquetes (tradicional y con memoria del estado) inspecciona los campos de cabecera IP, TCP, UDP e ICMP a la hora de decidir qué paquetes deben pasar a través del cortafuegos. Sin embargo, para detectar muchos tipos de ataques necesitamos llevar a cabo una **inspección profunda de paquetes**, es decir, mirar más allá de los campos de cabecera examinando los propios datos de aplicación transportados por los paquetes. Como hemos visto en la Sección 8.9.1, las pasarelas de aplicación realizan a menudo una inspección profunda de los paquetes. Pero cada pasarela de aplicación solo lleva a cabo esa tarea para una aplicación específica.

Por ello, existirá un nicho para otro tipo más de dispositivo: un dispositivo que no solo examine las cabeceras de todos los paquetes que le atraviesen (como los filtros de paquetes), sino que también lleve a cabo una inspección profunda de los paquetes (a diferencia de lo que sucede con los filtros de paquetes). Cuando uno de tales dispositivos detecte un paquete sospechoso o una serie sospechosa de paquetes puede impedir que esos paquetes entren en la red de la organización. O, si la actividad solo se considera sospechosa, el dispositivo podría dejar pasar los paquetes pero enviando alertas a un administrador de red, que puede de ese modo echar un vistazo más detallado a dicho tráfico y tomar las medidas oportunas. Un dispositivo que genere alertas cuando observe la presencia de tráfico potencialmente malicioso se denomina **sistema de detección de intrusiones (IDS, Intrusion Detection System)**. Un dispositivo que filtre el tráfico sospechoso se denomina **sistema de prevención de intrusiones (IPS, Intrusion Prevention System)**. En esta sección vamos a estudiar ambos tipos de sistemas (IDS e IPS) conjuntamente, dado que el aspecto técnico más interesante de dichos sistemas es cómo detectan el tráfico sospechoso (y no si envían alertas o eliminan los paquetes). Por tanto, vamos a utilizar el término sistemas IDS para referirnos tanto a los sistemas IPS como a los IDS.

Un sistema IDS se puede emplear para detectar una amplia gama de ataques, incluyendo los de mapeado de red (generados, por ejemplo, por nmap), escaneo de puertos, escaneos de la pila TCP, ataques DoS de inundación del ancho de banda, gusanos y virus, ataques de vulnerabilidades del sistema operativo y ataques de vulnerabilidades de aplicación. (En la Sección 1.6 puede ver una panorámica de los ataques de red.) Actualmente, miles de organizaciones utilizan sistemas IDS. Muchos de estos sistemas implementados son propietarios, comercializados por Cisco, Check Point y otros fabricantes de equipos de seguridad. Pero otros muchos de los sistemas IDS implantados son sistemas de dominio público, como el tremadamente popular sistema IDS Snort del que hablaremos en breve.

Una organización puede implantar uno o más sensores IDS en su red. La Figura 8.36 muestra una organización con tres sensores IDS. Cuando se implantan múltiples sensores, normalmente funcionan de manera concertada, enviando información acerca de las actividades de tráfico sospechoso a un procesador IDS central que recopila e integra la información y envía alarmas a los administradores de la red cuando lo considera apropiado. En la Figura 8.36, la organización ha dividido su red en dos regiones: una región de alta seguridad, protegida por un filtro de paquetes y una pasarela de aplicación y monitorizada por sensores IDS; y una región de menor seguridad, denominada **zona desmilitarizada (DMZ, Demilitarized Zone)** que está protegida solo por el filtro de paquetes, aunque también está monitorizada mediante sensores IDS. Observe que la DMZ incluye los servidores de la organización que necesitan comunicarse con el mundo exterior, como su servidor web público y su servidor DNS autoritativo.

El lector puede estar preguntándose al llegar a este punto por qué se emplean múltiples sensores IDS. ¿Por qué no colocar simplemente un sensor IDS justo detrás del filtro de paquetes (o incluso integrado con el filtro de paquetes) en la Figura 8.36? Como vamos a ver en breve, un IDS no solo tiene que llevar a cabo una inspección profunda de los paquetes, sino que también debe comparar cada paquete que pasa con decenas de miles de “firmas”; esto puede requerir una cantidad significativa de procesamiento, en particular si la organización recibe del orden de gigabits/segundo de tráfico procedente de Internet. Colocando los sensores IDS un poco más aguas abajo, cada sensor solo ve una fracción del tráfico de la organización y puede cumplir más fácilmente con su tarea. De todos modos, hoy día hay disponibles sistemas IDS e IPS de altas prestaciones y muchas organizaciones suelen conformarse con un único sensor localizado cerca de su router de acceso.

Los sistemas IDS se pueden clasificar en términos generales en **sistemas basados en firma** o **sistemas basados en anomalías**. Un IDS basado en firmas mantiene una amplia base de datos de firmas de ataque. Cada firma es un conjunto de reglas concernientes a una actividad de intrusión.

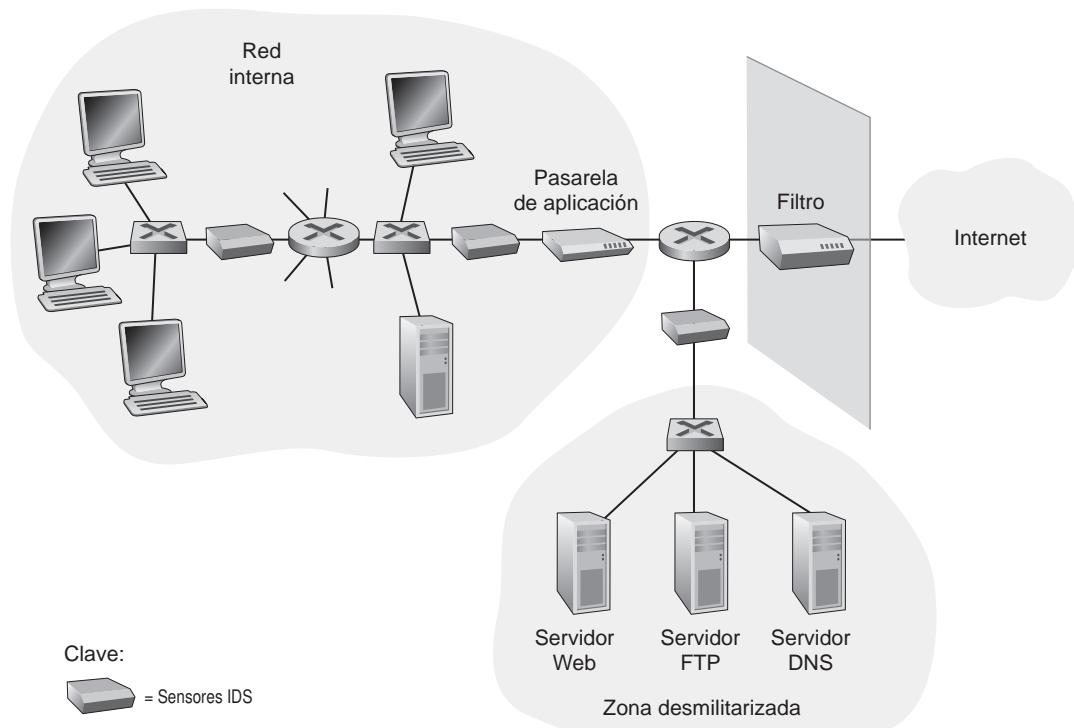


Figura 8.36 ♦ Una organización en la que se ha implantado un filtro, una pasarela de aplicación y sensores IDS.

Una firma puede ser simplemente una lista de características acerca de un determinado paquete (por ejemplo, números de puerto de origen y de destino, tipo de protocolo y una cadena específica de bits en la carga útil del paquete) o puede estar relacionada con una serie de paquetes. Las firmas normalmente son creadas por ingenieros de seguridad de red experimentados que se dedican a investigar los ataques conocidos. El administrador de red de una organización puede personalizar las firmas o añadir otras de su creación a la base de datos.

Operacionalmente, un sistema IDS basado en firmas analiza cada paquete que pasa a su través, comparando cada paquete husmeado con las firmas de su base de datos. Si un paquete (o una serie de paquetes) concuerda con una firma de la base de datos, el IDS genera una alerta. La alerta podría enviarse al administrador de red en un mensaje de correo electrónico, podría mandarse al sistema de gestión de la red o podría simplemente almacenarse en un registro para su futura inspección.

Los sistemas IDS basados en firma, aunque están ampliamente implantados, presentan una serie de limitaciones. La más importante es que se requiere un conocimiento previo del ataque para generar una firma precisa. En otras palabras, un IDS basado en firmas es completamente inútil frente a nuevos ataques que todavía no hayan sido investigados. Otra desventaja es que incluso si se produce una concordancia con una firma, puede que dicha concordancia no sea el resultado de un ataque, con lo que se generaría una falsa alarma. Finalmente, puesto que es necesario comparar cada paquete con una amplia colección de firmas, el IDS puede verse desbordado por las necesidades de procesamiento y debido a ello fracasar a la hora de detectar muchos paquetes maliciosos.

Un IDS basado en anomalías crea un perfil de tráfico observando el tráfico durante la operación normal. Después busca flujos de paquetes que sean estadísticamente inusuales, como por ejemplo un porcentaje inusual de paquetes ICMP o un crecimiento exponencial súbito en el escaneo de puertos y barridos mediante ping. Lo mejor de los sistemas IDS basados en anomalías es que no dependen del conocimiento previo acerca de los ataques existentes; es decir, pueden detectar potencialmente nuevos ataques no documentados. Por otro lado, el problema de distinguir entre el tráfico normal y el tráfico estadísticamente inusual es extremadamente complejo. A fecha de hoy, la mayoría de los sistemas IDS implantados son basados en firmas, aunque algunos de ellos incluyen algunas características de los sistemas basados en anomalías.

Snort

Snort es un IDS de dominio público y código abierto, con centenares de miles de implantaciones conocidas [Snort 2012; Koziol 2003]. Puede ejecutarse sobre plataformas Linux, UNIX y Windows. Utiliza la interfaz genérica de análisis libpcap, que también es utilizada por Wireshark y otros muchos husmeadores de paquetes. Puede gestionar fácilmente 100 Mbps de tráfico; para instalaciones con velocidades de tráfico del orden de gibabit/segundo puede ser necesario emplear múltiples sensores Snort.

Para entender un poco cómo funciona Snort, examinemos un ejemplo de firma utilizada por Snort:

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ICMP PING NMAP"; dsiz: 0; itype: 8;)
```

Esta firma concordará con cualquier paquete ICMP que entre en la red de la organización (\$HOME_NET) desde el exterior (\$EXTERNAL_NET), que sea de tipo 8 (ping ICMP) y que carezca de carga útil (dsiz = 0). Puesto que nmap (véase la Sección 1.6) genera paquetes ping con estas características específicas, esta firma está diseñada para detectar los barridos ping realizados con nmap. Cuando un paquete concuerda con esta firma, Snort genera una alerta que incluye el mensaje “ICMP PING NMAP”.

Lo más impresionante acerca de Snort quizás sea la enorme comunidad de usuarios y expertos de seguridad que mantienen su base de datos de firmas. Normalmente, al cabo de pocas horas de detectarse un nuevo ataque, la comunidad Snort escribe y publica una firma del ataque que después es descargada por los centenares de miles de implantaciones Snort distribuidas por todo el mundo.

Además, utilizando la sintaxis de firmas de Snort, los administradores de red pueden adaptar las firmas a las necesidades de su propia organización modificando las firmas existentes o creando otras completamente nuevas.

8.10 Resumen

En este capítulo hemos examinado los diversos mecanismos que nuestros amantes secretos, Benito y Alicia, pueden utilizar para comunicarse de forma segura. Hemos visto que Benito y Alicia están interesados en la confidencialidad (en el sentido de que solo ellos sean capaces de entender el contenido de los mensajes transmitidos), en la autenticación del punto terminal (con el fin de estar seguros de que están hablando entre ellos) y en la integridad de los mensajes (quieren estar seguros de que sus mensajes no son alterados en el camino). Por supuesto, la necesidad de comunicaciones seguras no está confinada a los amantes secretos. De hecho, hemos visto en las Secciones 8.5 a 8.8 que la seguridad puede utilizarse en varias capas de una arquitectura de red para protegerse frente a los malos que tienen en sus manos un enorme arsenal de posibles ataques.

En la primera parte del capítulo hemos presentado varios de los principios que subyacen a las comunicaciones seguras. En la Sección 8.2 hemos cubierto técnicas criptográficas para el cifrado y descifrado de datos, incluyendo la criptografía de clave simétrica y la criptografía de clave pública. DES y RSA se han examinado como casos de estudio específicos de estas dos clases principales de técnicas criptográficas que se emplean en las redes actuales.

En la Sección 8.3 hemos examinado dos métodos que permiten proporcionar mecanismos para asegurar la integridad de los mensajes: los códigos de autenticación de mensajes (MAC, *Message Authentication Code*) y las firmas digitales. Los dos métodos presentan una serie de paralelismos. Ambos utilizan funciones hash criptográficas y ambas técnicas nos permiten verificar el origen del mensaje, así como la integridad del propio mensaje. Una diferencia importante es que los códigos MAC no se basan en el cifrado, mientras que las firmas digitales requieren una infraestructura de clave pública. Ambas técnicas se usan ampliamente en la práctica, como hemos visto en las Secciones 8.5 a 8.8. Además, las firmas digitales se utilizan para crear certificados digitales, los cuales son importantes para verificar la validez de las claves públicas. En la Sección 8.4 hemos examinado la autenticación del punto terminal y hemos visto cómo pueden emplearse los números distintivos para frustrar los ataques por reproducción.

En las Secciones 8.5 a 8.8 hemos examinado varios protocolos de red seguros que disfrutan de un amplio uso en la práctica. Hemos visto que la criptografía de clave simétrica se encuentra en el núcleo de PGP, SSL, IPsec y la seguridad inalámbrica. Hemos visto que la criptografía de clave pública es crucial tanto para PGP como para SSL. También hemos visto que PGP utiliza firmas digitales para proporcionar la integridad de los mensajes, mientras que SSL e IPsec utilizan códigos MAC. Conociendo ahora los principios básicos de la criptografía y habiendo estudiado cómo se utilizan realmente estos principios, el lector está ahora en posición de diseñar sus propios protocolos de red seguros.

Con las técnicas cubiertas en las Secciones 8.2 a 8.8 Benito y Alicia pueden comunicarse de forma segura. (Suponiendo que sean estudiantes de redes que han aprovechado el material de este libro y que de esa forma pueden evitar que Gertrudis descubra su relación amorosa.) Pero la confidencialidad es solo una pequeña parte de la panorámica general de la seguridad de las redes. Como hemos señalado en la Sección 8.9, el énfasis en la seguridad de red se ha puesto cada vez más en proteger la infraestructura de la red frente a potenciales ataques realizados por los malos. En la última parte del capítulo hemos abordado los cortafuegos y los sistemas IDS que inspeccionan los paquetes que entran y salen de la red de una organización.

Este capítulo ha cubierto una gran cantidad de temas, centrándose en los aspectos más importantes de la seguridad en las redes modernas. Animamos a los lectores que deseen profundizar a investigar las referencias citadas a lo largo del capítulo. En particular, recomendamos [Skoudis 2006] para

los temas relacionados con los ataques a la seguridad operacional, [Kaufman 1995] para el estudio de la criptografía y cómo se aplica a la seguridad de redes, [Rescorla 2001] que proporciona un tratamiento profundo pero asequible sobre SSL y [Edney 2003] que ofrece una exposición sobre la seguridad 802.11, incluyendo una investigación intuitiva sobre WEP y sus desventajas.

Problemas y cuestiones de repaso

Capítulo 8 Problemas de repaso

SECCIÓN 8.1

- R1. ¿Cuáles son las diferencias entre la confidencialidad de los mensajes y la integridad de los mismos? ¿Puede existir confidencialidad sin integridad? ¿Puede existir integridad sin confidencialidad? Justifique su respuesta.
- R2. Las entidades de Internet (routers, switches, servidores DNS, servidores web, sistemas terminales de usuario, etc.) a menudo necesitan comunicarse de forma segura. Proporcione tres parejas específicas de ejemplo de entidades de Internet que puedan desear comunicarse de forma segura.

SECCIÓN 8.2

- R3. Desde la perspectiva de un servicio, ¿cuál es una diferencia importante entre un sistema de clave simétrica y un sistema de clave pública?
- R4. Suponga que un intruso dispone de un mensaje cifrado así como de la versión descifrada de dicho mensaje. ¿Puede el intruso montar un ataque de solo texto cifrado, un ataque de texto en claro conocido o un ataque de texto claro seleccionado?
- R5. Considere un cifrado de 8 bloques. ¿Cuántos posibles bloques de entrada tiene este sistema de cifrado? ¿Cuántas posibles correspondencias existen? Si interpretamos cada correspondencia como una clave, entonces ¿cuántas posibles claves tiene este sistema de cifrado?
- R6. Suponga que N personas desean comunicarse con cada una de las otras $N - 1$ personas utilizando un sistema de cifrado de clave simétrica. Todas las comunicaciones entre cualesquier dos personas, i y j , son visibles para todas las demás personas de ese grupo de N y ninguna otra persona en este grupo debe poder decodificar sus comunicaciones. ¿Cuántas claves son necesarias en el sistema, considerado como un todo? Suponga ahora que se utiliza un sistema de cifrado de clave pública. ¿Cuántas claves se requerirán en este caso?
- R7. Suponga que $n = 10.000$, $a = 10.023$ y $b = 10.004$. Utilice una identidad de aritmética modular para calcular mentalmente $(a \cdot b) \bmod n$.
- R8. Suponga que desea cifrar el mensaje 10101111 cifrando el número decimal correspondiente al mensaje. ¿Cuál es el número decimal?

SECCIONES 8.3–8.4

- R9. ¿De qué forma una función hash proporciona una mejor comprobación de la integridad de los mensajes que una suma de comprobación, tal como la suma de comprobación de Internet?
- R10. ¿Se puede “descifrar” un valor hash de un mensaje para obtener el mensaje original? Explique su respuesta.
- R11. Considere una variante del algoritmo MAC (Figura 8.9) en la que el emisor envía $(m, H(m) + s)$, siendo $H(m) + s$ la concatenación de $H(m)$ y s . ¿Tiene algún defecto esta variante? ¿Por qué?
- R12. ¿Qué quiere decir que un documento firmado sea verificable y no falsificable?

- R13. ¿En qué sentido es mejor utilizar como firma digital la versión cifrada mediante clave pública del valor hash de un mensaje, en lugar de usar la versión cifrada mediante clave pública del mensaje completo?
- R14. Suponga que certificador.com crea un certificado para foo.com. Normalmente, el certificado completo será cifrado con la clave pública de certificador.com. ¿Verdadero o falso?
- R15. Suponga que Alicia tiene un mensaje que está dispuesta a enviar a cualquiera que se lo solicite. Miles de personas desean obtener el mensaje de Alicia, pero cada una de ellas desea estar segura de la integridad del mensaje. En este contexto, ¿qué piensa que es más adecuado: un esquema de integridad basado en MAC o uno basado en firma digital? ¿Por qué?
- R16. ¿Cuál es el propósito de un número distintivo en un protocolo de autenticación de punto terminal?
- R17. ¿Qué quiere decir que un número distintivo es un valor que solo se usa una vez en la vida? ¿En la vida de quién?
- R18. El esquema de integridad de los mensajes basado en HMAC, ¿es susceptible a los ataques por reproducción? En caso afirmativo, ¿cómo se puede incorporar al esquema un número distintivo con el fin de eliminar dicha susceptibilidad?

SECCIONES 8.5–8.8

- R19. Suponga que Benito recibe un mensaje PGP de Alicia. ¿Cómo puede Benito estar seguro de que Alicia ha creado el mensaje (en lugar de, por ejemplo, Gertrudis)? ¿Utiliza PGP un valor MAC para garantizar la integridad del mensaje?
- R20. En el registro SSL existe un campo para los números de secuencia SSL. ¿Verdadero o falso?
- R21. ¿Cuál es el propósito de los números distintivos aleatorios en el proceso de acuerdo de SSL?
- R22. Suponga que una sesión SSL emplea un cifrado de bloque con CBC. Verdadero o falso: el servidor envía al cliente el vector de inicialización (IV) en claro.
- R23. Suponga que Benito inicia una conexión TCP con Gertrudis, que pretende hacerse pasar por Alicia. Durante la fase de acuerdo, Gertrudis envía a Benito el certificado de Alicia. ¿En qué paso del algoritmo de acuerdo SSL descubrirá Benito que no está comunicándose con Alicia?
- R24. Considere la transmisión de un flujo de paquetes desde el host A al host B utilizando IPsec. Normalmente se establecerá una nueva asociación de seguridad (SA) para cada paquete enviado del flujo. ¿Verdadero o falso?
- R25. Suponga que TCP está ejecutándose sobre IPsec entre las sucursales y la oficina principal de la Figura 8.28. Si TCP retransmite el mismo paquete, entonces los dos paquetes correspondientes enviados por R1 tendrán el mismo número de secuencia en la cabecera ESP. ¿Verdadero o falso?
- R26. ¿Es lo mismo una asociación de seguridad IKE que una asociación de seguridad IPsec? ¿Verdadero o falso?
- R27. Considere el protocolo WEP para 802.11. Suponga que los datos son 10101100 y que el flujo de claves es 1111000. ¿Cuál será el texto cifrado resultante?
- R28. En WEP, en cada trama se envía un vector de inicialización (IV) en claro. ¿Verdadero o falso?

SECTION 8.9

- R29. Los filtros de paquetes con memoria del estado mantienen dos estructuras de datos. Nómbralas y describa brevemente qué hacen.
- R30. Considere un filtro de paquetes tradicional (sin conocimiento del estado). Este filtro puede filtrar paquetes basándose en los bits indicadores TCP, así como en otros campos de cabecera. ¿Verdadero o falso?

- R31. En un filtro de paquetes tradicional, cada interfaz puede tener su propia lista de control de acceso. ¿Verdadero o falso?
- R32. ¿Por qué una pasarela de aplicación debe trabajar conjuntamente con un filtro de router para ser efectiva?
- R33. Los sistemas IDS e IPS basados en firma inspeccionan las cargas útiles de los segmentos TCP y UDP. ¿Verdadero o falso?

Problemas

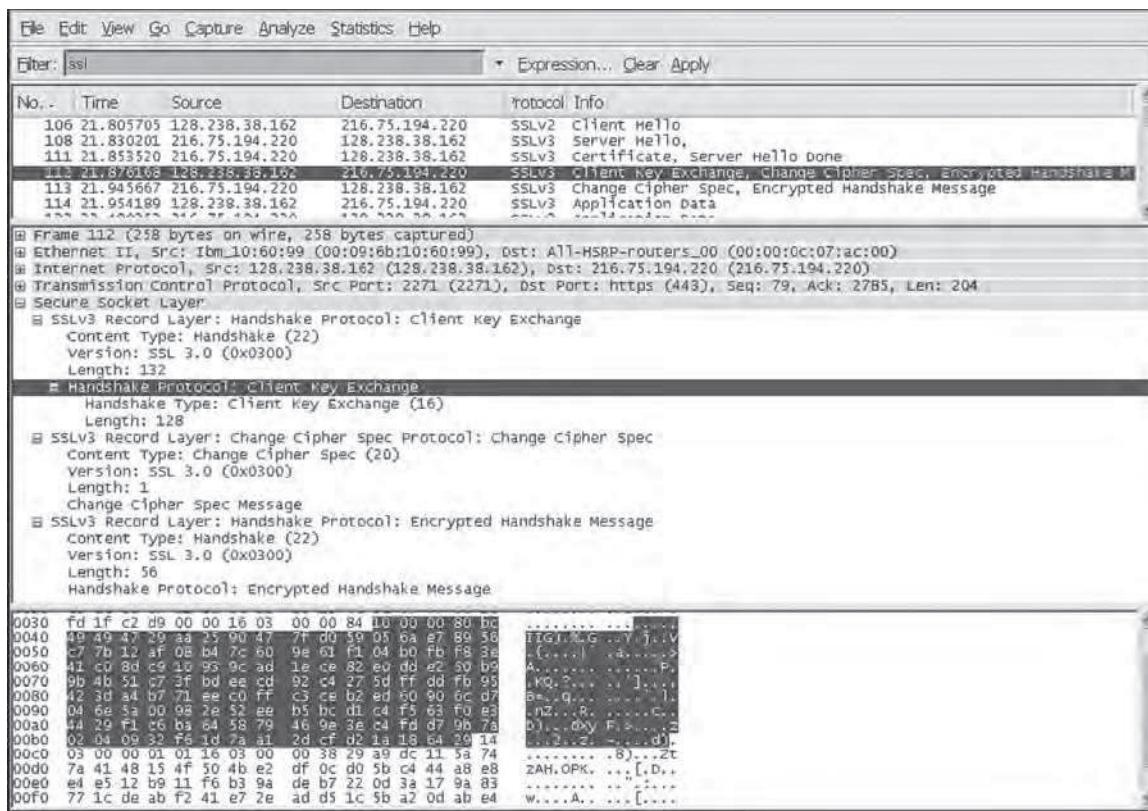
- P1. Utilizando el cifrado monoalfabético de la Figura 8.3, codifique el mensaje “Este problema es fácil.” Decodifique el mensaje “rmij’u uamu xyj”.
- P2. Demuestre que el ataque de texto en claro conocido de Gertrudis, en el que Gertrudis conoce las parejas de traducción (texto cifrado, texto en claro) de siete letras, reduce el número de posibles sustituciones que tienen que comprobarse en el ejemplo de la Sección 8.2.1 en un factor de aproximadamente 10^9 .
- P3. Considere el sistema polialfabético mostrado en la Figura 8.4. Un ataque de texto en claro seleccionado que es capaz de obtener la codificación del texto en claro del mensaje “Es extraño mojar queso en la cerveza o probar whisky de garrafa”, ¿será suficiente para decodificar todos los mensajes? ¿Por qué?
- P4. Considere el cifrado de bloque de la Figura 8.5. Suponga que cada cifrado de bloque T_i simplemente invierte el orden de los ocho bits de entrada (de manera que, por ejemplo, 11110000 se convierte en 00001111). Suponga además que el aleatorizador de 64 bits no modifica ninguno de los bits (de manera que el valor de salida del bit m -ésimo es igual al valor de entrada del mismo bit). (a) Con $n = 3$ y siendo la entrada de 64 bits original igual a 10100000 repetido ocho veces, ¿cuál es el valor de salida? (b) Repita el apartado (a) pero ahora cambie el último bit de la entrada de 64 bits original de 0 a 1. (c) Repita los apartados (a) y (b) pero ahora suponga que el aleatorizador de 64 bits invierte el orden de los 64 bits.
- P5. Considere el cifrado de bloque de la Figura 8.5. Para una determinada “clave”, Alicia y Benito tendrán que mantener ocho tablas, cada una de ellas de 8 bits por 8 bits. Para que Alicia (o Benito) pueda almacenar las ocho tablas, ¿cuántos bits de almacenamiento son necesarios? ¿Cómo se compara este número con el número de bits requeridos para un sistema de cifrado de bloque de 64 bits de tabla completa?
- P6. Considere el cifrado de bloque de 3 bits de la Tabla 8.1 y que dispone del texto en claro 100100100. (a) Suponga que inicialmente no se utiliza CBC. ¿Cuál es el texto cifrado resultante? (b) Suponga que Gertrudis husmea el texto cifrado. Suponiendo que Gertrudis sabe que se está utilizando un cifrado de bloque de 3 bits sin CBC (aunque no sabe cuál es el cifrado específico) ¿qué cosas podrá deducir Gertrudis? (c) Ahora suponga que se utiliza CBC con IV = 111. ¿Cuál es el texto cifrado resultante?
- P7. (a) Utilizando el algoritmo RSA, elija $p = 3$ y $q = 11$, y codifique la palabra “dog” cifrando cada letra por separado. Aplicando el algoritmo de descifrado a la versión cifrada recupere el mensaje de texto en claro original. (b) Repita el apartado (a) pero cifrando ahora la palabra “dog” como un mensaje m .
- P8. Considere RSA con $p = 5$ y $q = 11$.
 - Calcule el valor de n y z .
 - Sea e igual a 3. ¿Por qué este es un valor aceptable para e ?
 - Determine d tal que $de \equiv 1 \pmod{z}$ y $d < 160$.
 - Cifre el mensaje $m = 8$ utilizando la clave (n, e) . Sea c el texto cifrado correspondiente. Indique los detalles de todos los cálculos realizados. *Sugerencia:* para simplificar los cálculos, utilice la fórmula siguiente:

$$[(a \bmod n) \cdot (b \bmod n)] \bmod n = (a \cdot b) \bmod n$$

- P9. En este problema vamos a explorar el algoritmo de cifrado de clave pública de Diffie-Hellman (DH), que permite a dos entidades acordar una clave compartida. El algoritmo de Diffie-Hellman utiliza un número primo grande p y otro número también grande g pero menor que p . Tanto p como g se hacen públicos (por lo que cualquier atacante podría conocerlos). Con el algoritmo DH, Alicia y Benito seleccionan de forma independiente sendas claves secretas, S_A y S_B , respectivamente. Alicia calcula entonces su clave pública, T_A , elevando g a S_A y calculando luego mod p . Benito calcula de forma similar su propia clave pública, T_B , elevando g a S_B y tomando luego mod p . Alicia y Benito intercambian entonces sus claves públicas a través de Internet. A continuación Alicia calcula la clave secreta compartida S elevando T_B a S_A y luego reduciendo a mod p . De forma similar, Benito calcula la clave compartida S' elevando T_A a S_B y tomando después mod p .
- Demuestre que, en general, Alicia y Benito obtienen la misma clave simétrica, es decir, demuestre que $S = S'$.
 - Sean $p = 11$ y $g = 2$, y suponga que Alicia y Benito seleccionan sendas claves privadas $S_A = 5$ y $S_B = 12$, respectivamente. Calcule las claves públicas de Alicia y Benito, T_A y T_B . Indique todos los cálculos realizados.
 - Continuando con el apartado (b), calcule ahora la clave simétrica compartida S . Indique todos los cálculos realizados.
 - Dibuje un diagrama de temporización que muestre cómo puede atacarse el algoritmo de Diffie-Hellman utilizando un ataque por interposición. El diagrama de temporización debe tener tres líneas verticales: una para Alicia, otra para Benito y una tercera para la atacante Gertrudis.
- P10. Suponga que Alicia quiere comunicarse con Benito utilizando criptografía de clave simétrica con una clave de sesión K_S . En la Sección 8.2 hemos visto cómo puede utilizarse la criptografía de clave pública para distribuir la clave de sesión de Alicia a Benito. En este problema vamos a analizar cómo puede distribuirse la clave de sesión (sin criptografía de clave pública) utilizando un centro de distribución de claves (KDC, *Key Distribution Center*). El KDC es un servidor que comparte una clave simétrica secreta única con cada usuario registrado. Para Alicia y Benito vamos a designar estas claves mediante K_{A-KDC} y K_{B-KDC} . Diseñe un esquema que utilice el KDC para distribuir K_S a Alicia y a Benito. El esquema debe utilizar tres mensajes para distribuir la clave de sesión: un mensaje de Alicia hacia el KDC, un mensaje del KDC a Alicia y, finalmente, un mensaje de Alicia a Benito. El primer mensaje será $K_{A-KDC}(A, B)$. Utilizando la notación $K_{A-KDC}, K_{B-KDC}, S, A$ y B responda a las siguientes cuestiones:
- ¿Cuál es el segundo mensaje?
 - ¿Cuál es el tercer mensaje?
- P11. Calcule un tercer mensaje diferente de los dos mensajes de la Figura 8.8, que tenga la misma suma de comprobación que los mensajes de la Figura 8.8.
- P12. Suponga que Alicia y Benito comparten dos claves secretas: una clave de autenticación S_1 y una clave simétrica de cifrado S_2 . Amplíe la Figura 8.9 de forma que se proporcionen tanto integridad como confidencialidad.
- P13. En el protocolo de distribución de archivos P2P BitTorrent (véase el Capítulo 2) la semilla descompone el archivo en bloques y los pares se redistribuyen los bloques unos a otros. Si no se utilizara ninguna protección, un atacante podría fácilmente hacer que un torrente dejara de funcionar haciéndose pasar por un par benevolente y enviando bloques falsos a un pequeño subconjunto de pares del torrente. Estos pares no sospechosos redistribuirían entonces los bloques falsos a otros pares, que a su vez los distribuirían a otros. Por tanto, es crítico para el funcionamiento de BitTorrent que el sistema tenga un mecanismo que permita a un par verificar la integridad de un bloque, de modo que no se redistribuyan bloques falsos. Suponga que cuando un par se une a un torrente obtiene inicialmente un archivo .torrent de un origen

completamente de confianza. Describa un esquema simple que permita a los pares verificar la integridad de los bloques.

- P14. El protocolo de enrutamiento OSPF utiliza un valor MAC en lugar de firmas digitales para garantizar la integridad de los mensajes. ¿Por qué cree que se eligió un valor MAC en lugar de firmas digitales?
- P15. Considere nuestro protocolo de autenticación de la Figura 8.18, en el que Alicia se autentica ante Benito y que hemos visto que funciona bien (es decir, no hemos encontrado en él ningún fallo). Suponga ahora que mientras que Alicia se está autenticando ante Benito, este debe autenticarse ante Alicia. Indique el escenario mediante el cual Gertrudis, haciéndose pasar por Alicia, podría ahora autenticarse ante Benito como si fuera Alicia. (*Sugerencia:* considere que la secuencia de operaciones del protocolo, una en la que Gertrudis es el iniciador y otra en la que el iniciador es Benito, puede entremezclarse arbitrariamente. Preste atención especial al hecho de que tanto Benito como Alicia utilizan un número distintivo y que, si no se tiene cuidado, alguien podría utilizar maliciosamente el mismo número distintivo.)
- P16. Una pregunta natural que podemos plantearnos es si podemos utilizar un número distintivo y criptografía de clave pública para resolver el problema de la autenticación del punto terminal de la Sección 8.4. Considere el siguiente protocolo natural: (1) Alicia envía a Benito el mensaje “*Soy Alicia*”. (2) Benito selecciona un número distintivo, R , y se lo envía a Alicia. (3) Alicia emplea su clave *privada* para cifrar el número distintivo y envía el valor resultante a Benito. (4) Benito aplica la clave pública de Alicia al mensaje recibido. Así, Benito calcula R y autentica a Alicia.
- Realice un diagrama de este protocolo utilizando para las claves públicas y privadas la notación empleada en el libro.
 - Suponga que no se utilizan certificados. Describa cómo Gertrudis puede convertirse en un atacante por interposición (*man-in-the-middle*), interceptando los mensajes de Alicia y pretendiendo después ser Alicia ante Benito.
- P17. La Figura 8.19 muestra las operaciones que Alicia debe realizar con PGP para proporcionar confidencialidad, autenticación e integridad de los mensajes. Haga un diagrama de las correspondientes operaciones que Benito debe realizar con el paquete recibido de Alicia.
- P18. Suponga que Alicia quiere enviar un correo electrónico a Benito. Benito tiene una pareja de claves pública-privada (K_B^+, K_B^-) , y Alicia dispone del certificado de Benito. Pero Alicia no tiene una pareja de claves pública y privada. Alicia y Benito (y todo el mundo) comparten la misma función hash $H(\cdot)$.
- En esta situación, ¿es posible diseñar un esquema mediante el que Benito pueda verificar que es Alicia quien ha creado el mensaje? En caso afirmativo, indique cómo mediante un diagrama de bloques para Alicia y Benito.
 - ¿Es posible diseñar un esquema que proporcione confidencialidad para enviar el mensaje de Alicia a Benito? En caso afirmativo, indique cómo mediante un diagrama de bloques para Alicia y Benito.
- P19. Considere la salida de Wireshark mostrada en la página siguiente para una parte de una sesión SSL.
- ¿El paquete Wireshark 112 ha sido enviado por el cliente o por el servidor?
 - ¿Cuáles son el número de puerto y la dirección IP del servidor?
 - Suponiendo que no hay pérdidas ni retransmisiones, ¿cuál será el número de secuencia del siguiente segmento TCP enviado por el cliente?
 - ¿Cuántos registros SSL contiene el paquete Wireshark 112?
 - El paquete 112, ¿contiene una clave maestra (MS) o una clave maestra cifrada (EMS) o ninguna de las dos?
 - Suponiendo que el campo de tipo de acuerdo es de 1 byte y que cada campo de longitud es de 3 bytes, ¿cuáles son los valores del primer y último byte de la clave maestra (o de la clave maestra cifrada)?



(Pantalla de Wireshark reproducida con permiso de Wireshark Foundation.)

- g. ¿Cuántos registros SSL tiene en cuenta el mensaje cifrado de acuerdo del cliente?
 h. ¿Cuántos registros SSL tiene en cuenta el mensaje cifrado de acuerdo del servidor?
- P20. En la Sección 8.6.1 se muestra que, sin números de secuencia, Gertrudis (*man-in-the middle*) puede causar una catástrofe en una sesión SSL intercambiando segmentos TCP. ¿Podría Gertrudis hacer algo similar borrando un segmento TCP? ¿Qué necesitaría hacer para tener éxito con dicho ataque de borrado? ¿Qué efecto tendría?
- P21. Suponga que Alicia y Benito se están comunicando mediante una sesión SSL. Suponga que un atacante, que no dispone de ninguna de las claves compartidas, inserta un segmento TCP falso en un flujo de paquetes con la suma de comprobación TCP y los números de secuencia correctos (y con direcciones IP y números de puerto también correctos). ¿Aceptará el SSL en el lado receptor el paquete falso y pasará la correspondiente carga útil a la aplicación receptora? ¿Por qué?
- P22. Responda a las estas preguntas de tipo verdadero/falso que hacen referencia a la Figura 8.28.
- Cuando un host de 172.16.1/24 envía un datagrama a un servidor Amazon.com, el router R1 cifrará el datagrama utilizando IPsec.
 - Cuando un host de 172.16.1/24 envía un datagrama a un host de 172.16.2/24, el router R1 cambiará la dirección de origen y de destino del datagrama IP.
 - Suponga que un host de 172.16.1/24 inicia una conexión TCP con un servidor web situado en 172.16.2/24. Como parte de esta conexión, todos los datagramas enviados por R1 tendrán el número de protocolo 50 en el campo de cabecera IPv4 situado más a la izquierda.
 - Considere el envío de un segmento TCP desde un host situado en 172.16.1/24 a un host que se encuentra en 172.16.2/24. Suponga que el reconocimiento de este segmento se pier-

de, de modo que TCP reenvía el segmento. Puesto que IPsec utiliza números de secuencia, R1 no reenviará el segmento TCP.

- P23. Considere el ejemplo de la Figura 8.28. Suponga que Gertrudis está realizando un ataque por interposición (*man-in-the-middle*) y puede insertar datagramas en el flujo de datagramas que va de R1 a R2. Como parte de un ataque por reproducción, Gertrudis envía una copia duplicada de uno de los datagramas enviados de R1 a R2. ¿Descifrará R2 el datagrama duplicado y lo reenviará hacia la red de la sucursal? En caso negativo, describa en detalle cómo detecta R2 el datagrama duplicado.
- P24. Considere el siguiente protocolo pseudo-WEP. La clave es de 4 bits y el vector IV tiene 2 bits. El vector IV se añade al final de la clave al generar el flujo de claves. Suponga que la clave secreta compartida es 1010. Los flujos de claves para las cuatro entradas posibles son los siguientes:

101000: 00101011010101001011010100100 . . .
 101001: 1010011011001010110100100101101 . . .
 101010: 000110100011100010100101001111 . . .
 101011: 111110101000000101010100010111 . . .

Suponga que todos los mensajes tienen una longitud de 8 bits. Suponga que el ICV (comprobación de integridad) tiene una longitud de 4 bits y que se calcula combinando mediante XOR los primeros 4 bits de datos con los últimos 4 bits de datos. Suponga que el paquete pseudo-WEP consta de tres campos: el primero es el campo IV, luego el campo de mensaje y el último es el campo ICV, estando algunos de estos campos cifrados.

- a. Deseamos enviar el mensaje $m = 10100000$ utilizando el vector IV = 11 y WEP. ¿Cuáles serán los valores de los tres campos WEP?
 - b. Demuestre que cuando el receptor descifra el paquete WEP, recupera el mensaje y el ICV.
 - c. Suponga que Gertrudis intercepta un paquete WEP (no necesariamente con el vector IV = 11) y quiere modificarlo antes de reenviarlo hacia el receptor. Suponga que Gertrudis invierte el primer bit de ICV. Suponiendo que Gertrudis no conoce los flujos de claves para ninguno de los vectores IV, ¿qué otro bit o qué otros bits tiene que invertir también Gertrudis para que el paquete recibido pase la comprobación de ICV?
 - d. Justifique su respuesta modificando los bits del paquete WEP del apartado (a), descifrando el paquete resultante y verificando que el paquete pasa la comprobación de integridad.
- P25. Proporcione una tabla de filtrado y una tabla de conexión para un cortafuegos con memoria del estado que sea lo más restrictivo posible, pero que lleve a cabo lo siguiente:
- a. Permitir a todos los usuarios internos establecer sesiones Telnet con hosts externos.
 - b. Permitir a los usuarios externos navegar por el sitio web de la empresa en la dirección 222.22.0.12.
 - c. En caso contrario, bloquear todos los restantes tipos de tráfico entrante y saliente.
- La red interna es 222.22/16. En su solución, suponga que la tabla de conexiones almacena actualmente tres conexiones en caché, todas desde el interior hacia el exterior. En su solución tendrá que inventar los números de puerto y las direcciones IP apropiados.
- P26. Suponga que Alicia desea visitar el sitio web activistas.com utilizando un servicio de tipo TOR. Este servicio utiliza dos servidores proxy no confabulados Proxy1 y Proxy2. Alicia obtiene en primer lugar los certificados (cada uno de ellos contiene una clave pública) para Proxy1 y Proxy2 de algún servidor central. Designaremos mediante $K_1^+(\cdot), K_2^+(\cdot), K_1^-(\cdot)$ y $K_2^-(\cdot)$ las operaciones de cifrado/descifrado con las claves pública y privada RSA.
- a. Utilizando un diagrama de temporización, proporcione un protocolo (lo más simple posible) que permita a Alicia establecer una clave de sesión compartida S_1 con Proxy1. Designe mediante $S_1(m)$ el cifrado/descifrado de los datos m con la clave compartida S_1 .

- b. Utilizando un diagrama de temporización, proporcione un protocolo (lo más simple posible) que permita a Alicia establecer una clave de sesión compartida S_2 con Proxy2 *sin revelar su dirección IP a Proxy2*.
- c. Suponga ahora que las claves compartidas S_1 y S_2 ya están establecidas. Utilizando un diagrama de temporización proporcione un protocolo (lo más simple posible y que *no utilice criptografía de clave pública*) que permita a Alicia solicitar una página html de activistas.com *sin revelar su dirección IP a Proxy2 y sin revelar a Proxy1 qué sitio está visitando*. El diagrama debe terminar con la llegada de la solicitud HTTP al sitio activistas.com.

Práctica de laboratorio con Wireshark

En esta práctica de laboratorio (disponible en el sitio web del libro) vamos a investigar el protocolo SSL (*Secure Sockets Layer*, capa de sockets seguros). Recuerde de la Sección 8.6 que SSL se utiliza para dotar de seguridad a una conexión TCP y que se usa ampliamente en la práctica para proporcionar seguridad a las transacciones por Internet. En esta práctica vamos a centrarnos en los registros SSL enviados a través de la conexión TCP. Trataremos de perfilar y clasificar cada uno de los registros, con el objetivo de entender el por qué y el cómo de cada uno de ellos. Investigaremos los diversos tipos de registros SSL, así como los campos de los mensajes SSL. Lo haremos analizando una traza de los registros SSL intercambiados entre nuestro host y un servidor de comercio electrónico.

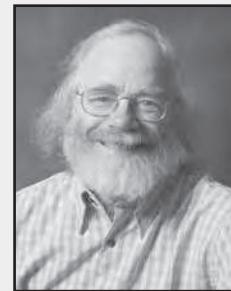
Práctica de laboratorio con IPsec

En esta práctica de laboratorio (disponible en el sitio web del libro), vamos a explorar cómo crear asociaciones de seguridad IPsec entre máquinas Linux. Puede realizar la primera parte de la práctica con dos máquinas Linux normales, cada una de las cuales deberá disponer de un adaptador Ethernet. Pero para la segunda parte de la práctica necesitará cuatro máquinas Linux, teniendo dos de ellas dos adaptadores Ethernet. En la segunda mitad de la práctica de laboratorio tendrá que crear asociaciones de seguridad IPsec utilizando el protocolo ESP en modo túnel. Tendrá que hacer esto definiendo primero manualmente las asociaciones de seguridad y luego haciendo que IKE las cree.

UNA ENTREVISTA CON...

Steven M. Bellovin

Steven M. Bellovin se unió al claustro de profesores de la Universidad de Columbia después de estar muchos años en el Laboratorio de Investigación de Servicios de Red de AT&T Labs Research en Florham Park, New Jersey, Estados Unidos. Su interés principal se centra en las redes, en la seguridad y en por qué ambos conceptos son incompatibles. En 1995 le concedieron el galardón Usenix Lifetime Achievement Award por su trabajo en la creación de Usenet, la primera red de intercambio de grupos de noticias que permitió enlazar varias computadoras y que los usuarios compartieran información y participaran en discusiones. Steve también es miembro electo de la National Academy of Engineering. Se graduó en la Universidad de Columbia y es doctor por la Universidad de Carolina del Norte en Chapel Hill.



¿Qué le condujo a especializarse en el área de la seguridad de redes?

Puede que le parezca extraño, pero la respuesta es muy simple: era divertido. Mi formación era en programación y administración de sistemas, lo que conduce de una forma bastante natural al campo de la seguridad. Y siempre me han interesado las comunicaciones, desde los tiempos en que realizaba trabajos de programación sobre sistemas en tiempo compartido cuando estudiaba en la universidad.

Mi trabajo en el campo de la seguridad sigue estando motivado por dos cosas: el deseo de continuar haciendo que las computadoras sean útiles, lo que significa que su función no pueda verse corrompida por ningún atacante y el deseo de proteger la privacidad.

¿Cuál era su visión de Usenet mientras la desarrollaba? ¿Y ahora?

Originalmente la veíamos como una forma de hablar acerca de la informática y de la programación de computadoras con otras personas dispersas por todo el país, con el añadido de grupos de usuarios locales para cuestiones administrativas, anuncios de compra-venta, etc. De hecho, mi predicción original es que se iban a generar uno o dos mensajes por día procedentes de entre 50 y 100 sitios como máximo. Pero el crecimiento real experimentado por esa red fue en temas relacionados con la sociedad, incluyendo (pero sin limitarse a ello) la interacción de los seres humanos con las computadoras. Mis grupos de noticias favoritos a lo largo de los años han sido cosas como rec.woodworking (dedicado a la carpintería), así como sci.crypt (dedicado a la criptografía).

Hasta cierto punto, netnews se ha visto desplazado por la Web. Si tuviera que comenzar a rediseñarla hoy en día tendría un aspecto muy distinto. Pero continúa siendo una herramienta excelente para apelar a una audiencia muy amplia interesada en un determinado tema, sin tener que depender de ningún sitio web concreto.

¿Hay alguien que le haya inspirado profesionalmente? ¿En qué forma?

El profesor Fred Brooks (el fundador y director original del departamento de Ciencias de la Computación de la Universidad de Carolina del Norte en Chapel Hill, que era también el jefe del equipo que desarrolló el IBM S/360 y el OS/360, y autor de *The Mythical Man-Month*) tuvo una enorme influencia en mi carrera profesional. Por encima de todo, me enseñó a mirar las cosas con una perspectiva amplia y a entender los compromisos necesarios: cómo examinar los problemas en el contexto del mundo real (y cuánto más lioso era el mundo real de lo que a un teórico le gustaría) y cómo equilibrar una serie de intereses en conflicto a la hora de diseñar una solución. La mayor parte del trabajo con computadoras es de ingeniería: el arte de llegar a los compromisos adecuados con el fin de satisfacer muchos objetivos contradictorios.

¿Cuál es su visión sobre el futuro de las redes y la seguridad?

Hasta ahora, buena parte de la seguridad de la que disfrutamos procede del aislamiento. Por ejemplo, un cortafuegos funciona cortando el acceso a ciertas máquinas y servicios. Pero nos encontramos en una era de conectividad creciente, con lo que cada vez es más complicado poder aislar las cosas. Y lo que es aún peor es que nuestros sistemas de producción requieren cada vez más elementos separados interconectados mediante redes. Dotar de seguridad a todo esto es uno de nuestros mayores desafíos.

¿Cuáles cree que han sido los mayores avances en el campo de la seguridad? ¿Cuánto más allá debemos ir todavía?

Al menos científicamente sabemos cómo hacer criptografía, lo que ha sido de una gran ayuda. Pero la mayor parte de los problemas de seguridad se deben a los errores en el código y ese es otro problema bastante más complicado. De hecho, es el problema más antiguo aún no resuelto en las ciencias de la computación y, en mi opinión, continuará siendo así. El desafío estriba en concebir cómo dotar de seguridad a los sistemas cuando no tenemos otro remedio que construirlos a partir de elementos inseguros. Hoy día ya podemos hacer eso para conseguir fiabilidad en presencia de fallos hardware; ¿podríamos hacer lo mismo para la seguridad?

¿Qué consejo le daría a los estudiantes relativo a la seguridad de Internet y de las redes en general?

El aprender los mecanismos es la parte más fácil. Lo que es más duro es aprender a pensar “en forma paranoica”. Es preciso recordar que las distribuciones de probabilidad no son aplicables en este campo: los atacantes pueden encontrar condiciones improbables y de hecho lo harán. Y los detalles importan; de hecho importan muchísimo.



Redes multimedia

Tumbada en la cama o mientras monta en autobús o en metro, gente de todo el mundo utiliza actualmente Internet para ver películas y programas de televisión a la carta. Empresas de distribución de películas y de televisión a través de Internet —como Netflix y Amazon en Norteamérica y Youku y Kankan en China— se han hecho enormemente populares. Pero la gente no se limita a ver videos por Internet, sino que utiliza sitios como Youtube para cargar y distribuir su propio contenido generado por el usuario, convirtiéndose en productores (y no solo consumidores) de video por Internet. Además, aplicaciones de red tales como Skype, Google Talk y WeChat (enormemente popular en China) permite a la gente no solo realizar “llamadas telefónicas” a través de Internet, sino también complementar esas llamadas con video y conferencias multipersonales. De hecho, cabe predecir que, a finales de la década actual, la mayor parte del consumo de video y de conversaciones de voz tendrá lugar extremo a extremo a través de Internet, normalmente utilizando dispositivos inalámbricos conectados a Internet mediante redes de acceso celulares y WiFi. La telefonía y la televisión tradicionales están quedando rápidamente obsoletas.

Comenzaremos este capítulo con una taxonomía de las aplicaciones multimedia en la Sección 9.1. Veremos que una aplicación multimedia puede clasificarse como *flujo de audio/vídeo almacenado, conversación de voz/vídeo sobre IP o flujo de audio/vídeo en vivo*. Veremos que cada una de estas clases de aplicaciones tiene su propio conjunto de requisitos de servicio, que difieren significativamente de los de las aplicaciones tradicionales elásticas como el correo electrónico, la navegación por la Web o el inicio de sesión remoto. En la Sección 9.2 analizaremos con un cierto detalle los flujos de video. Exploraremos muchos de los principios subyacentes a la tecnología de flujos de video, incluyendo el almacenamiento en buffer por parte del cliente, la precarga y la adaptación de la calidad del video al ancho de banda disponible. En la Sección 9.3 investigaremos las aplicaciones de conversación con voz y video, que, a diferencia de las aplicaciones elásticas, son altamente sensibles al retardo extremo a extremo, pero pueden tolerar ocasionales pérdidas de datos. Allí examinaremos el modo en que técnicas tales como la reproducción adaptativa, la corrección de errores hacia adelante y la ocultación de errores pueden mitigar los efectos de la pérdida de paquetes y el

retardo inducidos por la red. También analizaremos Skype como caso de estudio. En la Sección 9.4 estudiaremos RTP y SIP, dos protocolos populares para aplicaciones de conversaciones de voz y vídeo en tiempo real. En la Sección 9.5, investigaremos mecanismos de la red que pueden utilizarse para diferenciar una clase de tráfico (por ejemplo, aplicaciones sensibles al retardo, como las de conversación por voz) de otra (por ejemplo, aplicaciones elásticas como la navegación de páginas web), y para proporcionar un servicio diferenciado a múltiples clases de tráfico.

9.1 Aplicaciones multimedia en red

Definimos una aplicación de red multimedia como cualquier aplicación de red que emplee audio o vídeo. En esta sección vamos a proporcionar una taxonomía de las aplicaciones multimedia. Veremos que cada una de las clases de aplicaciones de la taxonomía tiene su propio conjunto de requisitos de servicio y de problemas de diseño. Pero antes de entrar en un análisis detallado de las aplicaciones multimedia en Internet, resulta útil considerar las características intrínsecas de los medios de audio y vídeo.

9.1.1 Propiedades del vídeo

Quizá la característica más sobresaliente del vídeo sea su **alta tasa de bits**. El vídeo que se distribuye a través de Internet suele ir de unos 100 kbps para la videoconferencia de baja calidad, a más de 3 Mbps para la distribución en tiempo real de películas de alta definición. Para entender cómo las demandas de ancho de banda del vídeo difieren de las de otras aplicaciones Internet, pensemos brevemente en tres usuarios distintos, cada uno de los cuales usa una aplicación Internet diferente. Nuestro primer usuario, Francisco, está hojeando rápidamente las fotografías publicadas en las páginas de Facebook de sus amigos. Supongamos que Francisco está accediendo a una nueva foto cada 10 segundos y que las fotos tienen un tamaño medio de 200 kbytes. (Como viene siendo habitual, y para simplificar las explicaciones, vamos a suponer que 1 kbyte = 8.000 bits.) Nuestro segundo usuario, Marta, está escuchando en su teléfono inteligente música que descarga de Internet (“la nube”). Supongamos que Marta está utilizando un servicio como Spotify para escuchar, una detrás de otra, múltiples canciones MP3, todas las cuales están codificadas a una tasa de 128 kbps. Nuestro tercer usuario, Víctor, está viendo un vídeo que ha sido codificado a 2 Mbps. Finalmente, supongamos que la longitud de sesión para los tres usuarios es de 4.000 segundos (unos 67 minutos). La Tabla 9.1 compara la velocidad de bit y la cantidad total de bytes transferidos para estos tres usuarios. Vemos que el flujo de vídeo consume, con mucha diferencia, el mayor ancho de banda, con una velocidad de bit que es más de diez veces superior a la de las aplicaciones de Facebook y de descarga de música. Por tanto, a la hora de diseñar aplicaciones de vídeo en red, lo primero que debemos tener en mente es la alta tasa de bits requerida por el vídeo. Dada la popularidad del vídeo y su alta tasa de bits, no le resultará sorprendente al lector que Cisco prediga [Cisco 2015] que los flujos de vídeo en vivo o almacenado representarán aproximadamente el 80% del tráfico global del mercado de consumo en Internet en el año 2019.

Otra característica importante del vídeo es que se lo puede comprimir, sacrificando calidad del vídeo a cambio de reducir la tasa de bits. Un vídeo es una secuencia de imágenes, que normalmente se muestran a velocidad constante, de por ejemplo 24 o 30 fotogramas por segundo. Una imagen con codificación digital y no comprimida está compuesta por una matriz de píxeles, estando cada píxel codificado mediante una serie de bits que representan la luminancia y el color. En un vídeo existen dos tipos de redundancia, pudiendo aprovecharse ambos gracias a la **compresión de vídeo**. La *redundancia espacial* es la redundancia dentro de una imagen determinada. Intuitivamente, una imagen compuesta principalmente por espacio en blanco tiene un alto grado de redundancia, por lo que podrá ser comprimida de manera eficiente sin sacrificar significativamente la calidad de la imagen. La *redundancia temporal* refleja la repetición entre una imagen y la siguiente. Si,

	Tasa de bits	Bytes transferidos en 67 min
Francisco - Facebook	160 kbps	80 Mbytes
Marta - Música	128 kbps	64 Mbytes
Victor - Vídeo	2 Mbps	1 Gbyte

Tabla 9.1 ♦ Comparación de los requisitos de tasas de bits de tres aplicaciones Internet.

por ejemplo, una imagen y la siguiente son exactamente iguales, no hay razón para recodificar la segunda imagen; en lugar de ello, es más eficiente indicar simplemente, durante la codificación, que la siguiente imagen es exactamente la misma. Los algoritmos de compresión comerciales de hoy en día pueden comprimir un vídeo a casi cualquier velocidad que se desee. Por supuesto, cuanto mayor sea la tasa de bits, mejores serán la calidad de imagen y la experiencia global de visualización por parte del usuario.

También podemos usar la compresión para crear **múltiples versiones** del mismo vídeo, cada una de ellas con distinto nivel de calidad. Por ejemplo, podemos usar la compresión para crear tres versiones del mismo vídeo, con velocidades de 300 kbps, 1 Mbps y 3 Mbps. Los usuarios pueden entonces decidir qué versión quieren ver, en función de su ancho de banda actualmente disponible. Los usuarios con conexiones Internet de alta velocidad podrían elegir la versión a 3 Mbps; los usuarios que ven el vídeo a través de 3G en su teléfono inteligente podrían elegir la versión a 300 kbps. De forma similar, el vídeo de una aplicación de videoconferencia puede comprimirse “sobre la marcha” para proporcionar la mejor calidad de vídeo que permita el ancho de banda extremo a extremo disponible para los usuarios que mantienen la conversación.

9.1.2 Propiedades del audio

El audio digital (incluyendo la voz y la música digitalizadas) tiene requisitos de ancho de banda significativamente menores que los del vídeo. Sin embargo, el audio digital tiene propiedades distintivas que hay que tener en cuenta a la hora de diseñar aplicaciones multimedia en red. Para entender estas propiedades, veamos primero cómo se convierte el audio analógico (que los seres humanos y los instrumentos musicales generan) en una señal digital:

- La señal de audio analógica se muestrea a una tasa fija, por ejemplo a 8.000 muestras por segundo. El valor de cada muestra será un cierto número real.
- A continuación, cada una de las muestras se redondea a uno de un número finito de valores. Esta operación se conoce con el nombre de **cuantización**. El número de tales valores finitos (denominados valores de cuantización) normalmente es una potencia de dos, por ejemplo 256 valores de cuantización.
- Cada uno de los valores de cuantización se representa mediante un número fijo de bits. Por ejemplo, si hay 256 valores de cuantización, entonces cada valor (y por tanto cada muestra de audio) se representa mediante 1 byte. Las representaciones en forma de bits de todas las muestras se concatenan para formar la representación digital de la señal completa. Por ejemplo, si una señal de audio analógica se muestrea con una tasa de 8.000 muestras por segundo y cada muestra se cuantiza y representa mediante 8 bits, entonces la señal digital resultante tendrá una tasa de 64.000 bits por segundo. Esta señal digital puede entonces convertirse de nuevo (es decir, decodificarse) en una señal analógica para su reproducción a través de unos altavoces. Sin embargo, la señal analógica decodificada es solo una aproximación de la señal de audio original y la calidad del sonido puede verse perceptiblemente degradada (por ejemplo, los sonidos de alta frecuencia pueden no estar presentes en la señal decodificada). Aumentando la tasa de muestreo y el número de valores de cuantización, la señal decodificada puede aproximarse mejor a la señal analógica.

original. Por tanto (al igual que sucede con el vídeo), existe un compromiso entre la calidad de la señal decodificada y los requisitos de almacenamiento y tasa de bits de la señal digital.

La técnica de codificación básica que acabamos de describir se conoce como **modulación por código de pulsos (PCM, Pulse Code Modulation)**. La codificación de voz normalmente utiliza PCM, con una tasa de muestreo de 8.000 muestras por segundo y 8 bits por muestra, lo que proporciona una tasa de 64 kbps. Los discos de audio compacto (CD) también emplean la modulación PCM, con una tasa de muestreo de 44.100 muestras por segundo y 16 bits por muestra; esto proporciona una tasa de 705,6 kbps para mono y de 1,411 Mbps para estéreo.

La voz y la música codificadas mediante PCM rara vez se utilizan en Internet. En su lugar (igual que pasa con el vídeo) se emplean técnicas de compresión para reducir las tasas de bit de los flujos. La voz humana puede ser comprimida a menos de 10 kbps y seguir siendo inteligible. Una técnica de compresión popular para música estéreo de calidad tipo CD es **MPEG 1 capa 3**, más comúnmente conocida como **MP3**. Los codificadores de MP3 pueden comprimir a muchas tasas diferentes; 128 kbps es la tasa de codificación más común y degrada muy poco la calidad del sonido. Un estándar relacionado es la denominada **codificación de audio avanzada (AAC, Advanced Audio Coding)**, que ha sido popularizado por Apple. Igual que con el vídeo, pueden crearse múltiples versiones de un flujo de audio pregrabado, cada una con una tasa de bits diferente.

Aunque las tasas de bit del audio suelen ser mucho menores que las del vídeo, los usuarios tienden a ser mucho más sensibles a los cortes en el audio que a los de un vídeo. Piense, por ejemplo, en una videoconferencia a través de Internet. Si se pierde la señal de vídeo durante unos segundos de vez en cuando, normalmente la videoconferencia puede celebrarse sin que los usuarios se frustren demasiado. Sin embargo, si la señal de audio se pierde con frecuencia, los usuarios pueden verse obligados a terminar la sesión.

9.1.3 Tipos de aplicaciones multimedia en red

Internet soporta una amplia variedad de útiles y entretenidas aplicaciones multimedia. En esta subsección, vamos a clasificar las aplicaciones multimedia en tres amplias categorías: (i) *flujos de audio/vídeo almacenado*, (ii) *conversaciones de voz/vídeo sobre IP* y (iii) *flujos de audio/vídeo en vivo*. Como pronto veremos, cada uno de estos tipos de aplicación tiene su propio conjunto de requisitos de servicio y problemas de diseño.

Flujos de audio/vídeo almacenado

Para mantener nuestra exposición en términos concretos, aquí vamos a centrarnos en los flujos de vídeo almacenado, que normalmente combinan componentes de audio y de vídeo. Los flujos de audio almacenado (como el servicio Spotify de descarga de música) son muy similares a los flujos de vídeo almacenado, aunque las tasas de bit suelen ser mucho más bajas.

En esta clase de aplicaciones, el medio subyacente es un vídeo pregrabado, como por ejemplo una película, un programa de televisión, un evento deportivo pregrabado o un vídeo pregrabado generado por el usuario (como los que nos encontramos habitualmente en YouTube). Estos vídeos pregrabados se almacenan en servidores, y los usuarios envían solicitudes a los servidores para ver estos vídeos *a la carta (on demand)*. Muchas empresas actuales de Internet proporcionan flujos de vídeo, incluyendo YouTube (Google), Netflix, Amazon y Hulu. Los flujos de vídeo almacenado tienen tres características distintivas principales.

- *Flujos*. En una aplicación de flujos de vídeo almacenado, el cliente inicia normalmente la reproducción del vídeo unos pocos segundos después de comenzar a recibir el vídeo desde el servidor. Esto significa que el cliente reproducirá el vídeo desde una posición del archivo mientras está recibiendo del servidor partes posteriores del mismo. Esta técnica, conocida como **transmisión de flujos (streaming)**, evita tener que descargar el archivo completo de vídeo (e incurrir en un retardo potencialmente largo) antes de comenzar la reproducción.

- *Interactividad.* Puesto que está pregrabado, el usuario puede poner en pausa el vídeo, saltar hacia delante, saltar hacia atrás, realizar un avance rápido, etc., a través del contenido del vídeo. El tiempo transcurrido desde que el usuario hace una de esas solicitudes, hasta que la acción se lleva a cabo en el cliente, debe ser inferior a unos pocos segundos para que la capacidad de respuesta del sistema se considere adecuada.
- *Reproducción continua.* Una vez que se inicia la reproducción del vídeo, debe proseguir de acuerdo con la temporización original de la grabación. Por tanto, los datos deben recibirse del servidor a tiempo para su reproducción en el cliente; si no es así, el usuario experimentará una congelación de la imagen (cuando el cliente espera los fotogramas retrasados) o un salto en el vídeo (cuando el cliente se salta los fotogramas retrasados).

La medida de rendimiento más importante para el vídeo almacenado es, con mucha diferencia, la tasa de transferencia media. Para poder permitir una reproducción continua, la red debe proporcionar a la aplicación de flujos multimedia una tasa de transferencia media igual o superior a la tasa de bits del propio vídeo. Como veremos en la Sección 9.2, utilizando búferes y precarga, es posible proporcionar una reproducción continua incluso aunque la tasa de transferencia fluctúe, siempre y cuando la tasa de transferencia media (promediada cada 5-10 segundos) siga siendo superior a la tasa de vídeo [Wang 2008].

En muchas aplicaciones de flujos de vídeo, el vídeo pregrabado se almacena y se distribuye utilizando una CDN, en lugar de un único centro de datos. Hay también muchas aplicaciones de flujos de vídeo P2P en las que el vídeo se almacena en los hosts de los usuarios (pares), llegando al cliente distintos fragmentos de vídeo desde diferentes pares, que pueden estar dispersos por todo el mundo. Dada la importancia de los flujos de vídeo en Internet, hablaremos de ellos con un cierto detalle en la Sección 9.2, prestando especial atención al almacenamiento en buffer dentro del cliente, a la precarga, a la adaptación de la calidad al ancho de banda disponibles y a la distribución a través de una red CDN.

Conversaciones de voz y vídeo sobre IP

Las conversaciones de voz en tiempo real a través de Internet suelen denominarse **telefonía Internet**, ya que, desde la perspectiva del usuario, son similares al servicio tradicional de telefonía de conmutación de circuitos. También se las denomina comúnmente **voz sobre IP (VoIP)**. Las conversaciones de vídeo son similares, salvo porque incluyen el vídeo de los participantes, además de sus voces. La mayoría de los sistemas actuales de conversación de voz y vídeo permiten a los usuarios crear conferencias con tres o más participantes. Hoy en día, las conversaciones de voz y vídeo se utilizan ampliamente en Internet, habiendo empresas como Skype, QQ y Google Talk que presumen de tener cientos de millones de usuarios diarios.

En nuestra exposición del Capítulo 2 acerca de los requisitos de servicio de las aplicaciones (Figura 2.4), identificamos una serie de ejes a lo largo de los cuales pueden clasificarse estos requisitos. Dos de estos ejes (consideraciones sobre temporización y tolerancia a la pérdida de datos) son particularmente importantes para las aplicaciones de conversación de voz y vídeo. Las consideraciones sobre temporización son importantes porque estas aplicaciones son extremadamente **sensibles a los retardos**. En una conversación con dos o más interlocutores, el retardo desde que un usuario habla o se mueve hasta que la acción se manifiesta en el otro extremo debe ser menor que unos pocos cientos de milisegundos. En el caso de voz, los retardos menores de 150 milisegundos no son percibidos por el oído humano, los retardos comprendidos entre 150 y 400 milisegundos pueden ser aceptables y los retardos mayores de 400 milisegundos pueden dar lugar a conversaciones frustrantes, si no completamente ininteligibles.

Por otro lado, las aplicaciones multimedia de conversación son **tolerantes a las pérdidas** (las pérdidas ocasionales sólo causan cortes ocasionales en la reproducción del audio/vídeo y estas pérdidas a menudo pueden ser parcial o totalmente disimuladas). Estas características de sensibilidad a los retardos y tolerancia a las pérdidas son claramente diferentes de las correspondientes a las

aplicaciones de datos elásticas como la navegación por la Web, el correo electrónico, las redes sociales y el inicio remoto de sesión. En las aplicaciones elásticas, los retardos largos son molestos pero no especialmente dañinos; la completitud y la integridad de los datos transferidos son, sin embargo, de suma importancia. Exploraremos con más detalle las conversaciones de voz y vídeo en la Sección 9.3, prestando particular atención a cómo la reproducción adaptativa, la corrección de errores hacia delante y la ocultación de errores pueden mitigar las pérdidas de paquetes y los retardos provocados por la red.

Flujos de audio y vídeo en vivo

Esta tercera clase de aplicaciones es similar a la difusión tradicional de radio y televisión, excepto porque la transmisión tiene lugar a través de Internet. Estas aplicaciones permiten a un usuario recibir una transmisión de radio o televisión *en vivo* (como, por ejemplo, un programa de noticias o un evento deportivo) emitida desde cualquier rincón del mundo. Hoy día, miles de emisoras de radio y televisión de todo el mundo emiten contenido a través de Internet.

Las aplicaciones en vivo de tipo difusión a menudo tienen muchos clientes que reciben el mismo programa de audio/vídeo simultáneamente. En la Internet actual, esto se suele realizar mediante redes CDN (Sección 2.6). Al igual que con los flujos multimedia almacenados, la red debe proporcionar a cada flujo multimedia en vivo una tasa de transferencia media que sea superior a la tasa a la que el vídeo se consume. Puesto que la transmisión es *en vivo*, el retardo también puede ser un problema, aunque las restricciones de temporización son menos estrictas que para las conversaciones de voz. Pueden tolerarse retardos de hasta unos diez segundos desde el momento en que el usuario solicita ver una transmisión *en vivo*, hasta que comienza la reproducción. En este libro no vamos a hablar de los flujos multimedia en vivo, porque muchas de las técnicas utilizadas para estos flujos (retardo inicial de almacenamiento en buffer, uso adaptativo del ancho de banda y distribución mediante una red CDN) son similares a las utilizadas con los flujos multimedia almacenados.

9.2 Flujos de vídeo almacenado

Para las aplicaciones de flujos de vídeo, los vídeos pregrabados se almacenan en servidores y los usuarios envían solicitudes a esos servidores para ver los vídeos a la carta. El usuario puede ver el vídeo de principio a fin sin interrupciones, puede dejar de ver el vídeo mucho antes de que termine o puede interactuar con el vídeo poniéndolo en pausa o saltando a una escena anterior o futura. Los sistemas de flujos de vídeo pueden clasificarse en tres categorías: **flujos UDP**, **flujos HTTP** y **flujos HTTP adaptativos** (véase la Sección 2.6). Aunque en la práctica se utilizan los tres tipos de sistemas, la mayoría de los sistemas actuales emplean flujos HTTP y flujos HTTP adaptativos.

Una característica común a los tres tipos de flujos de vídeo es el uso intensivo de buffers de aplicación en el lado del cliente, para mitigar los efectos de la variabilidad en los retardos extremo a extremo y en el ancho de banda disponible entre el servidor y el cliente. Para los flujos de vídeo (tanto almacenado como en vivo), los usuarios generalmente toleran un pequeño retardo inicial de varios segundos entre el momento en que el cliente solicita un vídeo y el momento en el que da comienzo la reproducción en el cliente. En consecuencia, cuando el vídeo comienza a llegar al cliente, éste no tiene por qué comenzar la reproducción de inmediato, sino que puede acumular una reserva de vídeo en un buffer de la aplicación. Una vez que el cliente ha acumulado una reserva de varios segundos de vídeo (que está almacenado en el buffer, pero que aún no se ha reproducido), el cliente puede comenzar la reproducción del vídeo. Ese tipo de **almacenamiento en buffer en el cliente** proporciona dos ventajas importantes. En primer lugar, al almacenamiento en buffer en el lado del cliente permite absorber variaciones en el retardo experimentado entre el servidor y el cliente. Si un fragmento específico de datos de vídeo se retrasa, ese largo retardo no será percibido por el usuario, siempre que esos datos lleguen antes de que se agote la reserva de vídeo recibido pero

aún no reproducido. En segundo lugar, si el ancho de banda disponible entre el servidor y el cliente cae brevemente por debajo de la tasa de consumo de vídeo, el usuario puede, de nuevo, continuar disfrutando de una reproducción continua, mientras que no se vacíe completamente el buffer de aplicación del cliente.

La Figura 9.1 ilustra el almacenamiento en buffer en el lado del cliente. En este ejemplo simple, suponga que el vídeo se codifica a una tasa fija de bits, por lo que cada bloque de vídeo contiene fotogramas que deben reproducirse en una misma cantidad fija de tiempo, Δ . El servidor transmite el primer bloque de vídeo en t_0 , el segundo bloque en $t_0 + \Delta$, el tercer bloque en $t_0 + 2\Delta$, y así sucesivamente. Una vez que el cliente comienza la reproducción, cada bloque debe reproducirse Δ unidades de tiempo después del bloque anterior, para poder mantener la temporización del vídeo grabado original. Debido a la variabilidad de los retardos extremo a extremo de la red, los diferentes bloques de vídeo experimentan diferentes retardos. El primer bloque de vídeo llega al cliente en el instante t_1 y el segundo lo hace en el instante t_2 . El retardo de red para el i -ésimo bloque es la distancia horizontal entre el instante en que el bloque fue transmitido por el servidor y el momento en que es recibido en el cliente; observe que el retardo de red varía de un bloque de vídeo al siguiente. En este ejemplo, si el cliente comenzara la reproducción en cuanto llegara el primer bloque, en el instante t_1 , entonces el segundo bloque no llegaría a tiempo para ser reproducido en $t_1 + \Delta$. En ese caso, la reproducción del vídeo tendría que detenerse (esperando a que llegue el bloque 2), o podríamos saltarnos el bloque 2; ambas soluciones darían lugar a defectos indeseables en la reproducción. En lugar de ello, si el cliente retrasa el inicio de la reproducción hasta t_3 , cuando ya han llegado los bloques 1 a 6, la reproducción periódica puede tener lugar, ya que *todos* los bloques se reciben antes del instante fijado para su reproducción.

9.2.1 Flujos UDP

Aquí solo vamos a hablar brevemente de los flujos UDP, remitiendo al lector, cuando sea apropiado, a explicaciones más detalladas de los protocolos utilizados por estos sistemas. Con los flujos UDP, el servidor transmite el vídeo a una velocidad igual a la velocidad de consumo del vídeo por parte del cliente, transmitiendo los fragmentos sobre UDP a una tasa constante. Por ejemplo, si la velocidad de consumo del vídeo es de 2 Mbps y cada paquete UDP transporta 8.000 bits de vídeo, el servidor transmitiría un paquete UDP a través de su socket cada $(8.000 \text{ bits})/(2 \text{ Mbps}) = 4 \text{ ms}$. Como vimos en el Capítulo 3, como UDP no utiliza ningún mecanismo de control de congestión, el servidor puede transmitir paquetes a través de la red a la velocidad de consumo del vídeo, sin las restricciones de control de la velocidad que tiene TCP. Los flujos UDP suelen emplear un buffer pequeño en el lado del cliente, del tamaño suficiente para almacenar menos de un segundo de vídeo.

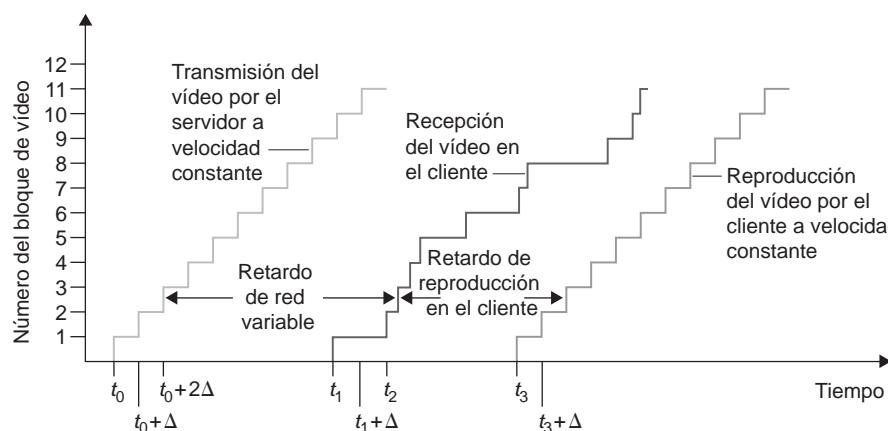


Figura 9.1 ♦ Retardo de reproducción de un flujo de vídeo en el cliente.

Antes de pasar los fragmentos de vídeo a UDP, el servidor los encapsula en paquetes de transporte especialmente diseñados para transportar audio y vídeo, utilizando el Protocolo de transporte en tiempo real (RTP, Real-Time Transport Protocol) [RFC 3550] u otro esquema similar (posiblemente propietario). Dejamos las explicaciones sobre RTP hasta la Sección 9.3, en la que hablaremos de RTP en el contexto de los sistemas de conversación de voz y vídeo.

Otra propiedad distintiva de los flujos UDP es que, además del flujo de vídeo que va del servidor al cliente, el cliente y el servidor también mantienen, en paralelo, una conexión de control separada, a través de la cual el cliente envía comandos relativos a cambios en el estado de la sesión (como pausa, continuación, reposicionamiento, etc.). El Protocolo de flujos en tiempo real (RTSP, *Real-Time Streaming Protocol*) [RFC 2326], que se explica con un cierto grado de detalle en el sitio web de este libro, es un popular protocolo abierto para ese tipo de conexiones de control.

Aunque los flujos UDP se han utilizado en muchos sistemas de código abierto y productos propietarios, presenta tres desventajas significativas. En primer lugar, debido a que el ancho de banda disponible entre el servidor y el cliente es variable e impredecible, los flujos UDP a velocidad constante pueden no garantizar una reproducción continua. Por ejemplo, piense en un caso en el que la velocidad de consumo del vídeo sea de 1 Mbps y en el que el ancho de banda disponible entre el servidor y el cliente sea usualmente superior a 1 Mbps, pero cada pocos minutos caiga por debajo de 1 Mbps durante unos pocos segundos. En esas circunstancias, un sistema de flujos UDP que transmita el vídeo a una velocidad constante de 1 Mbps sobre RTP/UDP probablemente proporcione una mala experiencia de usuario, con imágenes congeladas o fotogramas perdidos poco tiempo después de que el ancho de banda disponible caiga por debajo de 1 Mbps. La segunda desventaja de los flujos UDP es que necesitan un servidor de control de medios, como por ejemplo un servidor RTSP, para procesar las solicitudes de interactividad cliente-servidor y para realizar el seguimiento del estado del cliente (por ejemplo, el punto de reproducción del cliente dentro del vídeo, si el vídeo está en pausa o reproduciéndose, etc.) para *cada* sesión de cliente activa. Esto hace que aumenten el coste y complejidad globales de implantación de un sistema a gran escala de vídeo a la carta. La tercera desventaja es que muchos cortafuegos están configurados para bloquear el tráfico UDP, lo que impide que los usuarios situados detrás de esos cortafuegos reciban vídeo UDP.

9.2.2 Flujos HTTP

En los flujos HTTP, el vídeo simplemente se almacena en un servidor HTTP como un archivo normal, con un URL específico. Cuando un usuario quiere ver el vídeo, el cliente establece una conexión TCP con el servidor y emite la solicitud GET HTTP para ese URL. El servidor envía entonces el archivo de vídeo dentro de un mensaje de respuesta HTTP, tan rápidamente como sea posible, es decir, tan rápido como permitan los mecanismos de control de flujo y control de congestión de TCP. En el lado del cliente, los bytes se acumulan en un buffer de la aplicación cliente. Una vez que el número de bytes almacenados en el buffer supera un umbral predeterminado, la aplicación cliente comienza la reproducción: específicamente, extrae periódicamente fotogramas de vídeo del buffer de la aplicación cliente, los descomprime y los muestra en la pantalla del usuario.

Hemos visto en el Capítulo 3 que, al transferir un archivo a través de TCP, la velocidad de transmisión de servidor a cliente puede variar significativamente, debido al mecanismo de control de congestión de TCP. En particular, no es raro que la velocidad de transmisión varíe con un perfil de “dientes de sierra”, asociado con el control de congestión TCP. Además, los paquetes pueden sufrir retardos significativos debido al mecanismo de retransmisión de TCP. Debido a estas características de TCP, era creencia común en la década de los noventa que los flujos de vídeo nunca funcionarían bien sobre TCP. Con el tiempo, sin embargo, los diseñadores de sistemas de flujos de vídeo aprendieron que el control de congestión TCP y los mecanismos de transferencia fiable de los datos no impiden necesariamente la reproducción continua, cuando se utilizan el almacenamiento en buffer por parte del cliente y la técnica de precarga (de la que hablaremos en la siguiente sección).

El uso de HTTP sobre TCP también permite que el vídeo atraviese cortafuegos y sistemas NAT (que a menudo están configurados para bloquear la mayor parte del tráfico UDP y dejar pasar la mayor parte del tráfico HTTP) más fácilmente. Los flujos HTTP también eliminan la necesidad de disponer de un servidor de control de medios, como por ejemplo un servidor RTSP, reduciendo el coste de las implantaciones a gran escala a través de Internet. Debido a todas estas ventajas, la mayoría de las aplicaciones actuales de flujos de vídeo —incluyendo YouTube y Netflix— utilizan flujos HTTP (sobre TCP) como protocolos de transmisión de flujos subyacente.

Precarga de vídeo

Como acabamos de ver, se puede utilizar el almacenamiento en buffer en el lado del cliente para mitigar los efectos de las variaciones en los retardos extremo a extremo y en el ancho de banda disponible. En nuestro ejemplo anterior de la Figura 9.1, el servidor transmitía el vídeo a la velocidad a la que debía ser reproducido. Sin embargo, para los flujos de vídeo *almacenado*, el cliente puede tratar de descargar el vídeo a una velocidad *superior* a la tasa de consumo, **precargando** así fotogramas de vídeo que se consumirán en el futuro. Este vídeo precargado se almacena, naturalmente, en el buffer de la aplicación cliente. Dicha precarga se produce de forma natural con los flujos TCP, dado que el mecanismo de control de congestión de TCP tratará de utilizar todo el ancho de banda disponible entre el cliente y el servidor.

Para tratar de entender el mecanismo de precarga, veamos un ejemplo sencillo. Suponga que la velocidad de consumo del vídeo es de 1 Mbps, pero que la red es capaz de transmitir el vídeo desde el servidor al cliente a una velocidad constante de 1,5 Mbps. Entonces el cliente no solo será capaz de reproducir el vídeo con un retardo de reproducción muy pequeño, sino que también podrá incrementar en 500 Kbits cada segundo la cantidad de datos de vídeo almacenados en el buffer. De esta forma, si en el futuro el cliente recibiera los datos a una velocidad inferior a 1 Mbps durante un breve periodo de tiempo, podría continuar realizando una reproducción continua, gracias a los datos de reserva acumulados en su buffer. [Wang 2008] muestra que, cuando la tasa de transferencia TCP es aproximadamente el doble de la velocidad de bit de la información multimedia, los flujos sobre TCP dan como resultado una mínima inanición (agotamiento del buffer) y retardos de almacenamiento en el buffer pequeños.

Buffer de la aplicación cliente y buffers TCP

La Figura 9.2 ilustra la interacción entre el cliente y el servidor para el caso de los flujos HTTP. En el lado del servidor, la parte del archivo de vídeo coloreada de blanco ya ha sido enviada a través del socket del servidor, mientras que la parte más oscura es lo que queda por enviar. Después de “atravesar la puerta del socket”, los bytes se almacenan en el buffer de transmisión de TCP antes de ser enviados a Internet, como se describe en el Capítulo 3. En el caso de la Figura 9.2, como el buffer de transmisión de TCP en el lado del servidor está lleno, el servidor está momentáneamente impedido de enviar más bytes del archivo de vídeo a través del socket. En el lado del cliente, la aplicación cliente (el reproductor multimedia) lee bytes del buffer de recepción de TCP (a través de su socket de cliente) e inserta los bytes en el buffer de la aplicación cliente. Simultáneamente, la aplicación cliente extrae periódicamente fotogramas del buffer de la aplicación cliente, descomprime los fotogramas y los muestra en la pantalla del usuario. Observe que, si el buffer de la aplicación cliente tiene un tamaño mayor que el archivo de vídeo, entonces todo el proceso de mover los bytes desde el sistema de almacenamiento del servidor hasta el buffer de la aplicación cliente es equivalente a una descarga normal de archivo a través de HTTP: ¡el cliente simplemente extrae del servidor el archivo de vídeo tan rápidamente como lo permita TCP!

Considere ahora lo que sucede cuando el usuario pone el vídeo en pausa durante el proceso de envío del flujo. Mientras que dura la pausa, no se extraen bits del buffer de la aplicación cliente, aunque sí que continúan entrando bits en él, procedentes del servidor. Si el buffer de la aplicación cliente tiene un tamaño finito, puede llegar a llenarse, lo que dará lugar a una “presión inversa” que

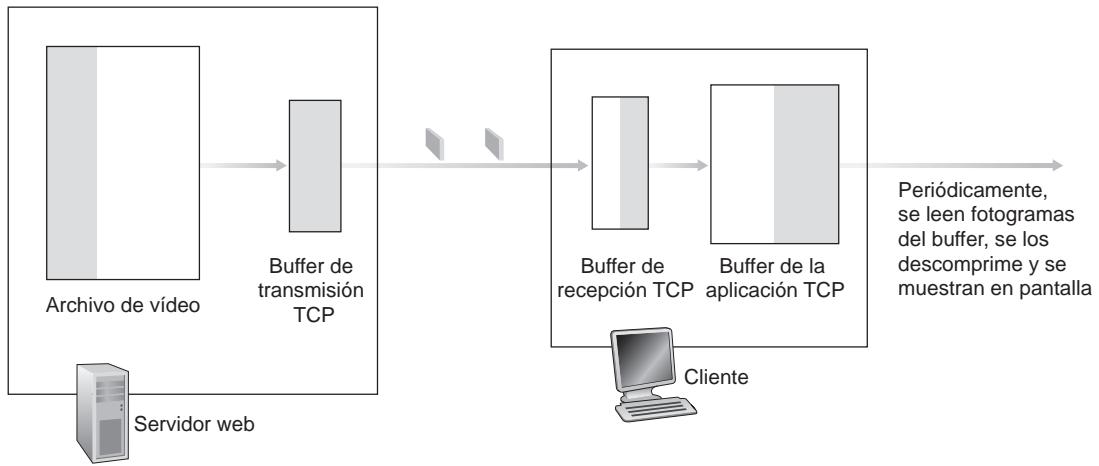


Figura 9.2 ♦ Flujo de vídeo almacenado a través de HTTP/TCP.

terminará alcanzando al servidor. Específicamente, una vez que se llena el buffer de la aplicación cliente, ya no pueden extraerse bytes del buffer de recepción TCP del cliente, por lo que también se llenará. Una vez que se llena el buffer de recepción TCP del cliente, ya no pueden extraerse bytes del buffer de transmisión TCP del servidor, que también termina por llenarse. Una vez que se ha llenado el buffer de transmisión TCP, el servidor deja de poder enviar más bytes a través del socket. Por tanto, si el usuario pone en pausa el video, el servidor puede verse obligado a dejar de transmitir, en cuyo caso quedará bloqueado hasta que el usuario reanude la reproducción.

De hecho, incluso durante la reproducción normal (es decir, sin pausas), si el buffer de la aplicación cliente se llena, la presión inversa hará que también se llenen los buffers TCP, lo que forzará al servidor a reducir su tasa de transmisión. Para determinar la tasa resultante, fíjese en que cuando la aplicación cliente extrae f bits, crea espacio para f bits en el buffer de la aplicación cliente, lo que a su vez permite al servidor enviar f bits adicionales. Por tanto la velocidad de transmisión del servidor no puede ser mayor que la velocidad de consumo del video por parte del cliente. Por tanto, *un buffer de la aplicación cliente lleno impone indirectamente un límite a la velocidad a la que puede enviarse ese video desde el servidor al cliente, cuando el flujo se transmite a través de HTTP*.

Análisis de la transmisión de un flujo de video

Un modelado sencillo nos permitirá comprender mejor el retardo inicial de reproducción y la congelación de la imagen debida al vaciado del buffer de la aplicación. Como se muestra en la Figura 9.3, sea B el tamaño (en bits) del buffer de la aplicación cliente y sea Q el número de bits que debe contener el buffer antes de que la aplicación cliente comience la reproducción. (Por supuesto, $Q < B$.) Sea r la velocidad de consumo del video, es decir, la velocidad a la que el cliente extrae los bits del buffer de la aplicación cliente durante la reproducción. Así, por ejemplo, si la velocidad del video es de 30 fotogramas/s y cada trama (comprimida) ocupa 100.000 bits, entonces $r = 3$ Mbps. Para que los árboles no nos impidan ver el bosque, vamos a ignorar los buffers TCP de transmisión y recepción.

Supongamos que el servidor envía bits a una velocidad constante x cuando el buffer del cliente no está lleno. (Esto es una enorme simplificación, ya que la velocidad de transmisión de TCP varía debido al control de congestión; examinaremos otras velocidades más realistas, $x(t)$, dependientes del tiempo, en los problemas del final del capítulo.) Suponga que el instante $t = 0$, el buffer de la aplicación cliente está vacío y el video comienza a llegar a él. ¿En qué momento $t = t_p$ comienza la reproducción? Y de paso, ¿en qué momento $t = t_f$ se llenará el buffer de la aplicación cliente?

En primer lugar, vamos a determinar t_p , el momento en que Q bits han entrado en el buffer de la aplicación y da comienzo la reproducción. Recuerde que los bits llegan al buffer de la

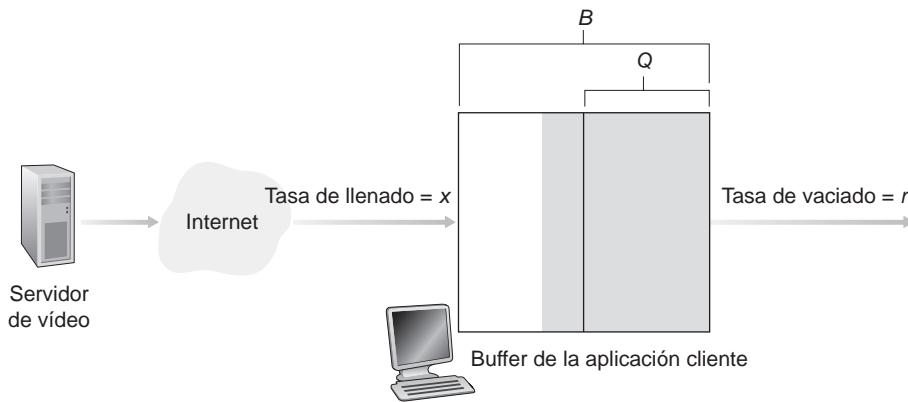


Figura 9.3 ♦ Análisis del almacenamiento en buffer en el lado del cliente durante la transmisión de un flujo de vídeo.

aplicación cliente a la velocidad x y que no se extrae *ningún* bit de dicho buffer antes de comenzar la reproducción. Entonces, la cantidad de tiempo requerida para acumular Q bits (el retardo inicial de almacenamiento en buffer) es $t_p = Q/x$.

Ahora determinemos t_f , el instante en que se llena el buffer de la aplicación cliente. Observemos primero que, si $x < r$ (es decir, si la velocidad de transmisión del servidor es inferior a la velocidad de consumo del vídeo), entonces el buffer del cliente nunca se llenará. De hecho, a partir del instante t_p , el buffer se vaciará a la velocidad r y solo se llenará a la velocidad $x < r$. En esas circunstancias, el buffer del cliente terminará por vaciarse por completo, en cuyo momento el vídeo se congelará en la pantalla mientras el buffer del cliente espera otros t_p segundos a que se acumulen Q bits de vídeo. *Por tanto, cuando la velocidad disponible en la red sea inferior a la velocidad del vídeo, la reproducción alternará entre períodos de reproducción continua y períodos de congelación de la imagen.* En uno de los problemas del capítulo se le pedirá que determine la duración de cada reproducción continua y de cada periodo de congelación, en función de Q , r y x . Ahora, vamos a determinar t_f cuando $x > r$. En este caso, a partir del instante t_p , la cantidad de vídeo en el buffer aumentará de Q a B a una velocidad $x - r$, ya que los bits se extraen a la velocidad r , mientras que llegan a la velocidad x , como se muestra en la Figura 9.3. A partir de estos datos, en uno de los problemas del capítulo se le pedirá que determine t_f , el instante en que se llena el buffer del cliente. Observe que, *cuando la velocidad disponible en la red sea mayor que la velocidad del vídeo, después del retardo inicial de almacenamiento en buffer el usuario disfrutará de una reproducción continua hasta que el vídeo termine.*

Terminación anticipada y reposicionamiento del vídeo

Los sistemas de flujos HTTP suelen hacer uso de la **cabecera de rango de bytes de HTTP** en el mensaje de solicitud GET HTTP, que especifica el rango concreto de bytes que el cliente desea extraer actualmente del vídeo solicitado. Esto resulta particularmente útil cuando el usuario quiere reposicionarse (saltar) en un punto futuro dentro del vídeo. Cuando el usuario salta a una nueva posición, el cliente envía una nueva solicitud HTTP, indicando mediante la cabecera de rango de bytes desde qué byte del archivo debe el servidor enviar los datos. Cuando el servidor recibe la nueva solicitud HTTP, puede olvidarse de cualquier solicitud anterior y empezar a enviar bytes a partir del punto indicado en la solicitud de rango de bytes.

Hablando del tema del reposicionamiento, merece la pena mencionar que, cuando un usuario salta a un punto futuro del vídeo o cancela la reproducción anticipadamente, algunos datos transmitidos por el servidor, que han sido precargados pero aún no han sido visualizados, no llegarán a verse nunca, lo cual constituye un desperdicio de ancho de banda y de recursos de servidor. Por

ejemplo, suponga que el buffer del cliente está lleno con B bits en algún instante t_0 , en mitad de la reproducción del vídeo, y que justo en ese instante el usuario salta a algún instante $t > t_0 + B/r$, y luego ve el vídeo hasta el final a partir de ese punto. En ese caso, los B bits del buffer no llegarán a verse nunca, y el ancho de banda y los recursos del servidor que se usaron para transmitir esos B bits se habrán desperdiciado completamente. Existe un desperdicio considerable de ancho de banda en Internet debido a la terminación anticipada, desperdicio que puede ser muy costoso, particularmente para los enlaces inalámbricos [Ihm 2011]. Por esta razón, muchos sistemas de transmisión de flujos utilizan buffers de la aplicación cliente de tamaño moderado, o limitan la cantidad de vídeo precargado utilizando la cabecera de rango de bytes de las solicitudes HTTP [Rao 2011].

El reposicionamiento y la terminación anticipada serían análogos a cocinar una copiosa comida, comer solo una parte de la misma y tirar el resto a la basura, desperdiando así los alimentos. Así que la próxima vez que sus padres le critiquen por desperdiciar comida al no terminarse la cena, ¡puede contestarles que ellos están desperdiciando ancho de banda y recursos del servidor cuando repositionan el vídeo mientras ven películas en Internet! Pero, por supuesto, dos cosas mal hechas no hacen una cosa bien hecha: ¡no hay que desperdiciar ni la comida, ni el ancho de banda!

En las secciones 9.2.1 y 9.2.2 hemos hablado, respectivamente, de los flujos UDP y los flujos HTTP. Un tercer tipo de transmisión de flujos es DASH (*Dynamic Adaptive Streaming over HTTP*, flujos dinámicos adaptativos sobre HTTP), que utiliza múltiples versiones del vídeo, cada una de ellas comprimida a una tasa diferente. DASH se explica en detalle en la Sección 2.6.2. A menudo se utilizan redes CDN para distribuir vídeo almacenado y en vivo. Las CDN se explican en detalle en la Sección 2.6.3.

9.3 Voz sobre IP

Las conversaciones de voz en tiempo real a través de Internet suelen denominarse **telefonía Internet**, ya que, desde la perspectiva del usuario, son similares al servicio tradicional de telefonía de conmutación de circuitos. También se las denomina comúnmente **voz sobre IP (VoIP)**. En esta sección vamos a describir los principios y protocolos que subyacen a VoIP. Las conversaciones de vídeo son similares en muchos aspectos a VoIP, salvo porque incluyen el vídeo de los participantes, además de sus voces. Para ser lo más concretos posible en nuestras explicaciones, en esta sección vamos a centrarnos en la voz, en lugar de en la combinación de voz y vídeo.

9.3.1 Limitaciones del servicio IP de entrega de mejor esfuerzo

El protocolo Internet de la capa de red, IP, proporciona un servicio de entrega de mejor esfuerzo. Esto quiere decir que el servicio trata de hacer lo que puede para transferir cada datagrama desde el origen hasta el destino de la forma más rápida posible. Sin embargo, no realiza ningún tipo de promesa en lo que se refiere a llevar el paquete a su destino dentro de un cierto límite de retardo, ni tampoco impone un límite al porcentaje de paquetes perdidos. La falta de tales garantías plantea desafíos significativos al diseño de aplicaciones de conversación en tiempo real, que son enormemente sensibles a los retardos, a las fluctuaciones y a la pérdida de paquetes.

En esta sección vamos a ver diversas formas en las que pueden mejorarse las prestaciones de VoIP sobre una red basada en un servicio de entrega de mejor esfuerzo. Nos vamos centrar en las técnicas de la capa de aplicación; es decir, en técnicas que no requieren efectuar ninguna modificación en el núcleo de la red, ni tampoco en la capa de transporte de los hosts terminales. Para ser lo más concretos posible, hablaremos de las limitaciones del servicio IP de entrega de mejor esfuerzo en el contexto de un ejemplo VoIP específico. El emisor genera bytes a una velocidad de 8.000 bytes por segundo y cada 20 milisegundos agrupa esos bytes en un fragmento. El fragmento y una cabecera especial (hablaremos de ella más adelante) se encapsulan en un segmento UDP, mediante una llamada a la interfaz de sockets. Por tanto, el número de bytes en un fragmento será

igual a $(20 \text{ milisegundos}) \cdot (8.000 \text{ bytes/segundo}) = 160 \text{ bytes}$, enviándose un segmento UDP cada 20 milisegundos.

Si cada paquete consigue llegar al receptor con un retardo extremo a extremo de duración constante, entonces los paquetes llegarán al receptor periódicamente cada 20 milisegundos. En estas condiciones ideales, el receptor puede simplemente reproducir cada fragmento en cuanto lo recibe. Pero lamentablemente, algunos paquetes pueden perderse y la mayoría de los paquetes no tendrá el mismo retardo extremo a extremo, incluso aunque Internet esté congestionada sólo ligeramente. Por esta razón, el receptor debe tener algo más de cuidado a la hora de determinar (1) cuándo hay que reproducir un fragmento y (2) qué hay que hacer cuando falta un fragmento.

Pérdida de paquetes

Considere uno de los segmentos UDP generados por nuestra aplicación VoIP. El segmento UDP está encapsulado dentro de un datagrama IP. A medida que el datagrama viaja por la red, pasa por una serie de buffers (es decir, colas) en los routers mientras espera a ser transmitido a través de los enlaces de salida. Es posible que uno o más de los buffers en la ruta existente entre el emisor y el receptor esté lleno, en cuyo caso el datagrama IP entrante será descartado, por lo que nunca llegará a la aplicación receptora.

Las pérdidas podrían eliminarse enviando los paquetes sobre TCP (que proporciona una transferencia de datos fiable) en lugar de sobre UDP. Sin embargo, los mecanismos de retransmisión se suelen considerar inaceptables para las aplicaciones de conversación audio en tiempo real, como VoIP, porque incrementan el retardo extremo a extremo [Bolot 1996]. Además, debido al control de congestión de TCP, la velocidad de transmisión en el emisor puede verse reducida después de una pérdida de paquetes, fijándose una velocidad inferior a la velocidad de consumo de paquetes en el receptor, lo que provocaría la inanición del buffer de recepción. Esto puede tener un impacto grave sobre la inteligibilidad de la voz en el receptor. Por estas razones, la mayoría de las aplicaciones de VoIP existentes se ejecutan sobre UDP de manera predeterminada. [Baset 2006] informa de que Skype utiliza UDP, a menos que un usuario se encuentre detrás de un traductor NAT o de un cortafuegos que bloquee los segmentos UDP (en cuyo caso se emplea TCP).

Pero las pérdidas de paquetes no son necesariamente tan desastrosas como podría parecer. De hecho, se pueden tolerar perfectamente tasas de pérdida de paquetes de entre el 1 y el 20 por ciento, dependiendo de cómo se codifique y transmita la voz y de cómo se oculten esas pérdidas en el receptor. Por ejemplo, los mecanismos de corrección de errores hacia adelante (FEC, *Forward Error Correction*) pueden ayudar a ocultar las pérdidas de paquetes. Veremos más adelante que con las técnicas FEC se transmite información redundante junto con la información original, de modo que una parte de los datos originales perdidos puede recuperarse a partir de esa información redundante. Sin embargo, si uno o más de los enlaces existentes entre el emisor y el receptor está severamente congestionado y la tasa de pérdida de paquetes excede del 10 o 20 por ciento (por ejemplo, en un enlace inalámbrico), entonces no hay nada que podamos hacer para conseguir una calidad de sonido aceptable. Claramente, el servicio de entrega de mejor esfuerzo tiene sus limitaciones.

Retardo extremo a extremo

El **retardo extremo a extremo** es la suma de los retardos de transmisión, de procesamiento y de puesta en cola de los routers, más los retardos de propagación de los enlaces y los retardos de procesamiento en los sistemas terminales. Para las aplicaciones de conversación en tiempo real, como VoIP, los retardos extremo a extremo inferiores a 150 milisegundos no son perceptibles por los oyentes humanos; los retardos entre 150 y 400 milisegundos son aceptables, aunque distan mucho de ser ideales, y los retardos mayores de 400 milisegundos pueden afectar seriamente a la interactividad en las conversaciones de voz. El lado receptor de una aplicación VoIP descartará normalmente todos los paquetes que estén retardados más de un cierto umbral, como por ejemplo más de 400 milisegundos. Por tanto, los paquetes cuyo retardo sea superior al umbral prefijado se perderán.

Fluctuación de los paquetes

Un componente crucial del retardo extremo a extremo son los retardos aleatorios de puesta en cola que los paquetes experimentan dentro de los routers de la red. A causa de estos retardos variables, el tiempo que transcurre desde el momento en que se genera un paquete en el origen hasta que se recibe en el destino puede fluctuar de un paquete a otro, como se muestra en la Figura 9.1. Este fenómeno se conoce como **fluctuación** o **jitter**. Por ejemplo, considere dos paquetes consecutivos dentro de nuestra aplicación VoIP. El emisor envía el segundo paquete 20 ms después de enviar el primero. Pero en el receptor, el espaciado entre estos paquetes puede ser mayor de 20 ms. Para ver por qué, suponga que el primer paquete llega a una cola prácticamente vacía dentro de un router y que, justo antes de que el segundo paquete llegue a esa misma cola, entran en la cola un gran número de paquetes procedentes de otros orígenes. Puesto que el primer paquete sufre un retardo de puesta en cola pequeño y el segundo paquete sufre un retardo de puesta en cola mucho mayor dentro de este router, el espaciado entre el primer y el segundo paquete será superior a 20 ms. De la misma manera, el espaciado entre paquetes consecutivos también podría ser inferior a 20 ms. Para ver por qué, considere de nuevo dos paquetes consecutivos. Suponga que el primer paquete se coloca al final de una cola que contiene un gran número de paquetes y que el segundo paquete llega a esa cola antes de que el primer paquete sea transmitido y antes de que otros paquetes de otros orígenes entren en la cola. En este caso, nuestros dos paquetes estarán uno detrás del otro dentro de la cola. Si el tiempo que se tarda en transmitir un paquete a través del enlace de salida del router es inferior a 20 ms, entonces el espaciado entre el primer y el segundo paquete será también inferior a 20 ms.

La situación es análoga a la que se produce cuando circulan vehículos por una carretera. Suponga que usted y un amigo viajan cada uno en su automóvil desde San Diego a Fénix. Suponga también que usted y su amigo tienen formas similares de conducir y que ambos viajan a 100 km/hora cuando el tráfico lo permite. Si su amigo comienza el viaje una hora antes que usted, entonces, dependiendo del tráfico con el que él y usted se encuentren, puede que llegue a Fénix menos de una hora o más de una hora después que su amigo.

Si el receptor ignora la presencia de las fluctuaciones y reproduce los segmentos en cuanto llegan, entonces la calidad del audio resultante puede llegar fácilmente a ser ininteligible en el receptor. Afortunadamente, las fluctuaciones pueden eliminarse a menudo utilizando **números de secuencia, marcas de tiempo** y **un retardo de reproducción**, como se explica a continuación.

9.3.2 Eliminación de las fluctuaciones al reproducir audio en el receptor

Para nuestra aplicación VoIP, en la que los paquetes se generan periódicamente, el receptor debe tratar de reproducir sincrónamente los fragmentos de voz en presencia de fluctuaciones aleatorias causadas por la red. Esto normalmente se hace combinando los siguientes tres mecanismos:

- *Precediendo cada fragmento con un número de secuencia.* El emisor incrementa el número de secuencia en una unidad por cada uno de los paquetes que genera.
- *Precediendo cada fragmento con una marca de tiempo.* El emisor marca cada fragmento con la hora a la que el fragmento fue generado.
- *Retardando la reproducción de los fragmentos en el receptor.* Como ya dijimos en nuestras explicaciones anteriores sobre la Figura 9.1, el retardo de reproducción de los fragmentos de audio recibidos debe ser lo suficientemente grande como para que la mayor parte de los paquetes se reciban antes de su instante de reproducción planificado. Este retardo de reproducción puede ser fijo a lo largo de toda la sesión de audio, o variar adaptativamente a lo largo de la misma.

Vamos a ver ahora cómo pueden aliviar o incluso eliminar los efectos de las fluctuaciones estos tres mecanismos, cuando se los combina apropiadamente. Vamos a examinar dos estrategias de reproducción: el retardo de reproducción fijo y el retardo de reproducción adaptativo.

Retardo de reproducción fijo

Con la estrategia basada en un retardo fijo, el receptor intenta reproducir cada fragmento exactamente q milisegundos después de que ese fragmento haya sido generado. Por tanto, si un fragmento tiene una marca de tiempo que indica que fue generado en el instante t , el receptor reproduce dicho fragmento en el instante $t + q$, suponiendo que el fragmento haya llegado antes de ese momento. Los paquetes que lleguen después de su instante de reproducción planificado se descartan y se consideran perdidos.

¿Qué valor sería adecuado para q ? VoIP puede permitir retardos de hasta unos 400 milisegundos, aunque se consigue una experiencia de conversación más satisfactoria con valores de q menores. Por otro lado, si hacemos q mucho menor que 400 milisegundos, entonces muchos paquetes podrían no llegar antes de su instante de reproducción planificado debido a la fluctuación de los paquetes por causa de la red. Por simplificar, digamos que si suelen experimentarse grandes variaciones en los retardos extremo a extremo, es preferible utilizar un valor de q grande; por el contrario, si el retardo es pequeño y las variaciones del mismo también, es preferible emplear un valor de q pequeño, tal vez inferior a 150 milisegundos.

En la Figura 9.4 se ilustra este compromiso entre el retardo de reproducción y la tasa de pérdida de paquetes. La figura muestra los instantes en que se generan los paquetes y los instantes en que esos paquetes se reproducen para un único periodo de conversación. En la figura se consideran dos retardos de reproducción iniciales diferentes. Como se muestra en la línea escalonada de la izquierda, el emisor genera paquetes a intervalos periódicos (por ejemplo, cada 20 milisegundos). El primer paquete de ese periodo de conversación se recibe en el instante r . Como se muestra en la figura, los instantes de llegada de los paquetes sucesivos no están espaciados de manera uniforme, debido a las fluctuaciones de la red.

Para la primera planificación de reproducción se fija un retardo de reproducción inicial constante de $p - r$. Con esta planificación, el cuarto paquete no llega antes de su instante de reproducción planificado, por lo que el reproductor considera que se ha perdido. En la segunda planificación de reproducción, el retardo de reproducción inicial fijo se hace igual a $p' - r$. Para esta planificación, todos los paquetes llegan antes de su instante de reproducción planificado, por lo que no se produce ninguna pérdida.

Retardo de reproducción adaptativo

El ejemplo anterior ilustra un importante compromiso entre el retardo y la tasa de pérdidas, que surge siempre a la hora de diseñar una estrategia de reproducción con retardos de reproducción fijos.

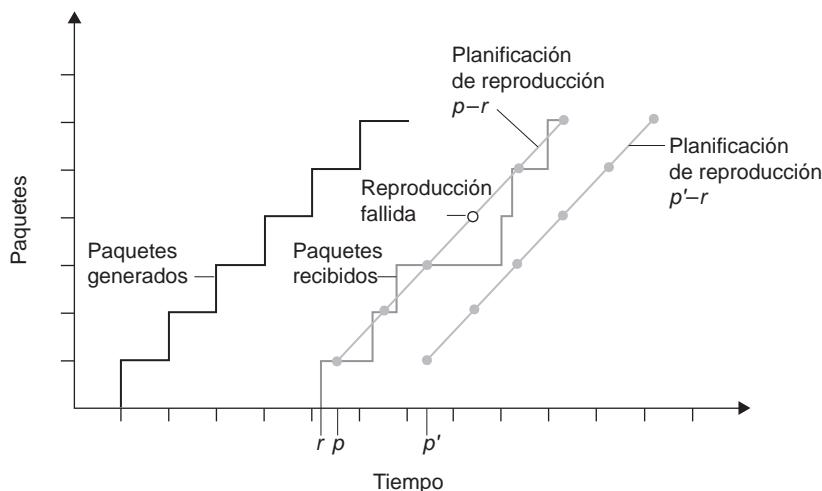


Figura 9.4 ♦ Pérdida de paquetes para diferentes retardos fijos de reproducción.

Si hacemos que el retardo inicial de reproducción sea grande, la mayoría de los paquetes llegarán a tiempo y por tanto habrá una tasa de pérdidas despreciable; sin embargo, para servicios de conversación como VoIP, esos grandes retardos pueden resultar molestos, si no intolerables. Idealmente, lo que queremos es minimizar el retardo de reproducción, bajo la restricción de que la tasa de pérdidas sea inferior a un cierto porcentaje pequeño.

La forma natural de tratar con este compromiso es estimar el retardo de la red y la varianza de ese retardo y ajustar el retardo de reproducción de acuerdo con ello al principio de cada periodo de conversación. Este ajuste adaptativo de los retardos de reproducción al principio de los periodos de conversación hará que los periodos de silencio del emisor se compriman y estiren; sin embargo, la compresión y el alargamiento de esos silencios en una proporción razonable no resulta perceptible durante la conversación.

De acuerdo con lo expuesto en [Ramjee 1994], ahora vamos a describir un algoritmo genérico que puede utilizar el receptor para ajustar adaptativamente sus retardos de reproducción. Con este fin, sean

t_i = la marca de tiempo del paquete i -ésimo, es decir, el instante en que el paquete fue generado por el emisor

r_i = el instante en el que el paquete i llega al receptor

p_i = el instante en el que el receptor reproduce el paquete i

El retardo extremo a extremo de la red para el paquete i -ésimo es $r_i - t_i$. Debido a la fluctuación inducida por la red este retardo variará de un paquete a otro. Sea d_i una estimación del retardo *promedio* de la red al recibirse el paquete i -ésimo. Esta estimación se calcula a partir de la marca de tiempo de la forma siguiente:

$$d_i = (1 - u) d_{i-1} + u (r_i - t_i)$$

donde u es una constante fija (por ejemplo, $u = 0,01$). Por tanto, d_i es una media móvil de los retardos de red observados $r_1 - t_1, \dots, r_i - t_i$. La estimación asigna un mayor peso a los retardos de red más recientemente observados que a los del pasado más distante. Esta forma de estimación no debería resultarle extraña al lector; una idea similar se emplea para estimar los tiempos de ida y vuelta en TCP, como hemos visto en el Capítulo 3. Sea v_i una estimación de la desviación media del retardo con respecto al retardo promedio estimado. Esta estimación también se construye a partir de las marcas de tiempo:

$$v_i = (1 - u) v_{i-1} + u |r_i - t_i - d_i|$$

Las estimaciones d_i y v_i se calculan para cada paquete recibido, aunque sólo se utilizan para determinar el punto de reproducción del primer paquete de cada periodo de conversación.

Una vez que se han calculado estas estimaciones, el receptor emplea el siguiente algoritmo para la reproducción de los paquetes. Si el paquete i es el primero de un periodo de conversación, su instante de reproducción p_i se calcula como:

$$p_i = t_i + d_i + Kv_i$$

donde K es una constante positiva (por ejemplo, $K = 4$). El propósito del término Kv_i es establecer el instante de reproducción lo suficientemente lejos en el futuro como para que sólo una pequeña fracción de los paquetes entrantes correspondientes a ese periodo de conversación se pierdan debido a una llegada tardía. El instante de reproducción para los siguientes paquetes de un periodo de conversación se calcula mediante un desplazamiento con respecto al instante en el que el primer paquete de ese periodo de conversación se reproduce. En particular, sea,

$$q_i = p_i - t_i$$

la diferencia temporal entre el instante en que se generó el primer paquete del periodo de conversación y el instante en se reprodujo. Si el paquete j también pertenece al mismo periodo de conversación, se reproducirá en el instante

$$p_j = t_j + q_i$$

El algoritmo recién descrito tiene mucho sentido, asumiendo que el receptor pueda determinar si un paquete es el primero de su correspondiente periodo de conversación. Esto puede realizarse examinando la energía de la señal en cada paquete recibido.

9.3.3 Recuperación frente a pérdidas de paquetes

Hemos explicado con un cierto grado de detalle cómo una aplicación VoIP puede enfrentarse a la fluctuación de los paquetes. Ahora vamos a describir brevemente diversos esquemas que tratan de preservar una calidad de audio aceptable en presencia del fenómeno de pérdida de paquetes. Dichos esquemas se denominan **esquemas de recuperación frente a pérdidas**. Aquí, definimos la pérdida de paquetes en un sentido amplio: un paquete se pierde si nunca llega al receptor o si llega después de su instante de reproducción planificado. Nuestro ejemplo de VoIP nos servirá de nuevo de contexto para describir los esquemas de recuperación frente a pérdidas.

Como hemos mencionado al principio de esta sección, la retransmisión de los paquetes perdidos puede no resultar apropiada en una aplicación de conversación en tiempo real como VoIP. De hecho, la retransmisión de un paquete que no haya llegado antes de su instante de reproducción planificado no tiene ningún sentido. Y retransmitir un paquete que ha desbordado la cola de un router no suele poder hacerse con la suficiente rapidez. Debido a estas consideraciones, las aplicaciones VoIP utilizan a menudo algún tipo de esquema de anticipación de pérdidas. Dos tipos de esquemas de anticipación de pérdidas son la **corrección de errores hacia adelante (FEC, Forward Error Correction)** y el **intercalado**.

Corrección de errores hacia adelante (FEC)

La idea básica de la técnica FEC consiste en añadir información redundante al flujo original de paquetes. A cambio de incrementar ligeramente la velocidad de transmisión, esa información redundante puede utilizarse para reconstruir aproximaciones o versiones exactas de algunos de los paquetes perdidos. Siguiendo lo expuesto en [Bolot 1996] y [Perkins 1998], vamos a esbozar aquí dos mecanismos FEC sencillos. El primero de ellos envía un fragmento redundante codificado después de cada n fragmentos. El fragmento redundante se obtiene aplicando la operación OR exclusiva a los n fragmentos originales [Shacham 1990]. De esta manera, si algún paquete del grupo de los $n + 1$ paquetes se pierde, el receptor puede reconstruir completamente el paquete perdido. Sin embargo, si se pierden dos o más paquetes de un grupo, el receptor no podrá reconstruirlos. Manteniendo pequeño el tamaño del grupo, $n + 1$, puede recuperarse un gran porcentaje de los paquetes perdidos, siempre que la tasa de pérdidas no sea excesiva. Sin embargo, cuanto más pequeño sea el tamaño del grupo, mayor será el incremento relativo de la velocidad de transmisión. En particular, la velocidad de transmisión se incrementa según un factor de $1/n$; por ejemplo, si $n = 3$, la velocidad de transmisión se incrementa en un 33 por ciento. Además, este esquema sencillo incrementa el retardo de reproducción, ya que el receptor tiene que esperar a recibir el grupo de paquetes completo antes de poder iniciar la reproducción. Para ver detalles más prácticos acerca de cómo funciona el mecanismo FEC en el transporte de datos multimedia, consulte [RFC 5109].

El segundo mecanismo FEC consiste en enviar un flujo de audio de menor resolución como información redundante. Por ejemplo, el emisor podría crear un flujo de audio nominal y otro flujo correspondiente de baja resolución y baja tasa de bits. (El flujo nominal podría ser una codificación PCM a 64 kbps y el flujo de menor calidad podría ser una codificación GSM a 13 kbps.) El flujo de baja velocidad de bits se denomina flujo redundante. Como se muestra en la Figura 9.5, el emisor

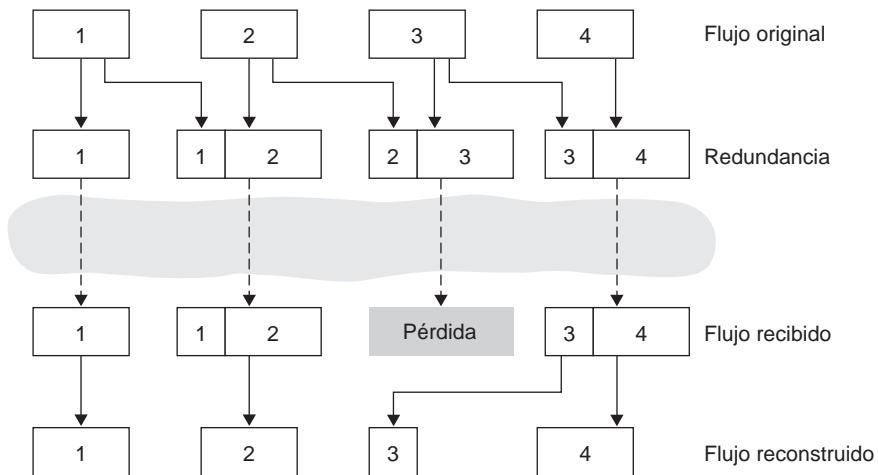


Figura 9.5 ♦ Adición de información redundante de menor calidad.

construye el n -ésimo paquete tomando el n -ésimo fragmento del flujo nominal y añadiéndole el $(n - 1)$ -ésimo fragmento del flujo redundante. De este manera, cuando existan pérdidas de paquetes no consecutivos, el receptor podrá ocultar esa pérdida reproduciendo el fragmento codificado de baja tasa de bits que llegue con el siguiente paquete. Por supuesto, los fragmentos de baja tasa de bits proporcionan una menor calidad que los fragmentos nominales. Sin embargo, un flujo compuesto por una gran mayoría de fragmentos de alta calidad, algunos fragmentos ocasionales de baja calidad y en el que no falte ningún fragmento, proporciona una buena calidad global de audio. Observe que, en este esquema, el receptor sólo tiene que recibir dos paquetes para comenzar a reproducir, por lo que el incremento en el retardo de reproducción es pequeño. Además, si la codificación a baja velocidad es muy inferior a la codificación nominal, entonces el incremento en la velocidad de transmisión será pequeño.

Para poder resolver el problema de la pérdida de paquetes consecutivos, podemos utilizar una sencilla variante. En lugar de añadir simplemente el $(n - 1)$ -ésimo fragmento de baja tasa de bits al n -ésimo fragmento nominal, el emisor puede añadir el $(n - 1)$ -ésimo y el $(n - 2)$ -ésimo fragmentos de baja tasa de bits, o añadir el $(n - 1)$ -ésimo y el $(n - 3)$ -ésimo fragmentos de baja tasa de bits, etc. Añadiendo más fragmentos de baja tasa de bits a cada fragmento nominal, la calidad de audio en el receptor será aceptable para una mayor variedad de entornos de entrega de paquetes de mejor esfuerzo en presencia de pérdidas. Por otro lado, cada fragmento adicional incrementa el ancho de banda de transmisión y el retardo de reproducción.

Intercalado

Como alternativa a la transmisión de información redundante, una aplicación VoIP puede enviar audio intercalado. Como se muestra en la Figura 9.6, el emisor reordena las unidades de datos de audio antes de su transmisión, de forma que las unidades originalmente adyacentes están separadas por una cierta distancia dentro del flujo transmitido. El intercalado puede mitigar el efecto de la pérdida de paquetes. Si, por ejemplo, las unidades tienen una longitud de 5 milisegundos y los fragmentos tienen una longitud de 20 milisegundos (es decir, hay cuatro unidades por fragmento), entonces el primer fragmento podría contener las unidades 1, 5, 9 y 13; el segundo fragmento podría contener las unidades 2, 6, 10 y 14, y así sucesivamente. La Figura 9.6 muestra que la pérdida de un único paquete en un flujo intercalado da como resultado una serie de huecos pequeños en el flujo reconstruido, en lugar del único hueco de gran tamaño que aparecería si tuviéramos un flujo no intercalado.

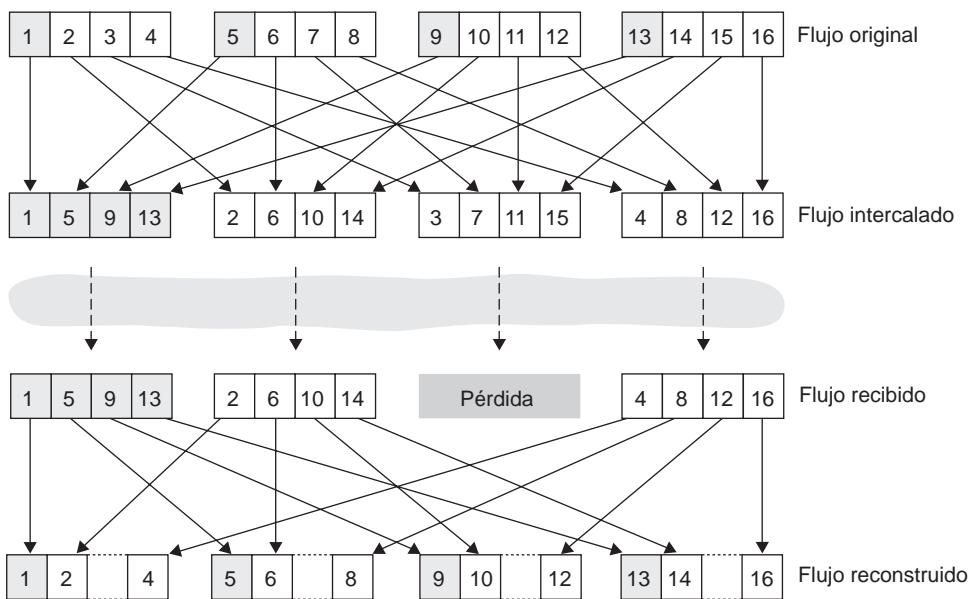


Figura 9.6 ♦ Transmisión de audio intercalado.

El intercalado puede mejorar significativamente la calidad percibida en un flujo de audio [Perkins 1998]. También tiene una baja sobrecarga de datos adicionales. La desventaja obvia del intercalado es que incrementa la latencia. Esto limita su uso en las aplicaciones de conversación, como VoIP, aunque puede funcionar bien para el envío de flujos de audio almacenado. Una de las principales ventajas del intercalado es que no incrementa los requisitos de ancho de banda de un flujo.

Ocultación de errores

Los esquemas de ocultación de errores tratan de generar un sustituto para un paquete perdido que sea similar al original. Como se explica en [Perkins 1998], esto es posible porque las señales de audio, y en particular la voz, exhiben una gran cantidad de auto-similitud a corto plazo. Por ello, estas técnicas funcionan para tasas de pérdida relativamente bajas (de menos del 15 por ciento) y para paquetes de pequeño tamaño (4–40 milisegundos). Cuando la longitud de la pérdida se approxima a la longitud de un fonema (5–100 milisegundos), estas técnicas dejan de funcionar, ya que el oyente podría perderse fonemas completos.

Quizá la forma más simple de recuperación por parte del receptor es la repetición de paquetes. La repetición de paquetes sustituye los paquetes perdidos con copias de los paquetes que hayan llegado inmediatamente antes de la pérdida. Esta técnica tiene una baja complejidad computacional y proporciona unos resultados razonablemente buenos. Otra forma de recuperación basada en el receptor es la interpolación, que utiliza el audio anterior y posterior a la pérdida para interpolar un paquete adecuado que permita cubrir el hueco. La interpolación funciona algo mejor que la repetición de paquetes, pero requiere un uso significativamente mayor de recursos de computación [Perkins 1998].

9.3.4 Caso de estudio: VoIP con Skype

Skype es una aplicación VoIP inmensamente popular, que cuenta con más de 50 millones de cuentas activas a diario. Además de proporcionar servicio VoIP entre hosts, Skype ofrece servicios

de comunicación de host a teléfono, servicios de teléfono a host y servicios de videoconferencia entre múltiples hosts. (De nuevo, un host en este contexto es cualquier dispositivo IP conectado a Internet, incluyen computadoras PC, tabletas y teléfonos inteligentes.) Skype fue adquirido por Microsoft en 2011.

Puesto que el protocolo Skype es propietario, y puesto que todos los paquetes de control y de datos de Skype están cifrados, resulta difícil determinar con precisión cómo funciona Skype. Sin embargo, a partir de la información proporcionada por el sitio web de Skype y a partir de varios estudios de medición, los investigadores han podido desentrañar cómo funciona Skype en términos generales [Baset 2006; Guha 2006; Chen 2006; Suh 2006; Ren 2006; Zhang X 2012]. Tanto para voz como para vídeo, los clientes Skype tienen a su disposición muchos códecs distintos, que son capaces de codificar la información multimedia dentro de un amplio rango de velocidades y de calidades. Por ejemplo, se ha podido medir que las tasas de vídeo de Skype van desde solo 30 kbps, para una sesión de baja calidad, hasta casi 1 Mbps, para una sesión de alta calidad [Zhang X 2012]. Normalmente, la calidad de audio de Skype es mejor que la calidad del servicio telefónico tradicional proporcionado por la red telefónica cableada. (Los códecs Skype suelen usar velocidades de muestreo de la voz de 16.000 muestras/s o superiores, lo que proporciona unos tonos más ricos que el servicio telefónico tradicional, que utiliza 8.000 muestras/s) De manera predeterminada, Skype envía los paquetes de audio y vídeo sobre UDP. Sin embargo, los paquetes de control se envían sobre TCP, y los propios paquetes de datos también se envían sobre TCP cuando los cortafuegos bloquean los flujos UDP. Skype usa técnicas FEC para la recuperación frente a pérdidas de los flujos, tanto de voz como de vídeo, enviados sobre UDP. El cliente Skype también adapta los flujos transmitidos de audio y vídeo a las condiciones actuales de la red, modificando la calidad del vídeo y la cantidad de información FEC adicional [Zhang X 2012].

Skype utiliza técnicas P2P de forma innovadora, lo que ilustra convenientemente cómo puede usarse P2P en aplicaciones que van más allá de la distribución de contenido y la compartición de archivos. Al igual que sucede con la mensajería instantánea, la telefonía Internet host-host es inherentemente P2P, ya que el núcleo de la aplicación consiste en que parejas de usuarios (es decir, pares) se comunican entre sí en tiempo real. Pero Skype también emplea técnicas P2P para otras dos importantes funciones: localización de los usuarios y NAT traversal.

Como se muestra en la Figura 9.7, los pares (hosts) en Skype están organizados en una red jerárquica paralela, en la que cada par se clasifica como un superpar o un par normal. Skype mantiene un índice que establece la correspondencia entre los nombres de usuario de Skype y las actuales direcciones IP (y números de puerto). Este índice está distribuido entre los superpares. Cuando Alicia quiere llamar a Benito, su cliente Skype busca en el índice distribuido para determinar la dirección IP actual de Benito. Como el protocolo Skype es propietario, actualmente no sabemos cómo están organizadas las entradas de índice entre los superpares, aunque es bastante posible que se emplee algún tipo de organización DHT (tablas de hash distribuidas).

También se usan técnicas P2P en los **retransmisores** Skype, que resultan útiles para establecer llamadas entre hosts dentro de redes domésticas. Muchas configuraciones de red doméstica proporcionan acceso a Internet a través de dispositivos NAT, como se explica en el Capítulo 4. Recuerde que un dispositivo NAT impide que un host situado fuera de la red doméstica inicie una conexión con un host situado dentro de la misma. Si *ambos* interlocutores Skype tienen dispositivos NAT, entonces hay un problema: ninguno puede aceptar una llamada iniciada por el otro, lo que hace que sea aparentemente imposible la conversación. El uso inteligente de los superpares y de los retransmisores permite resolver adecuadamente este problema. Suponga que cuando Alicia se registra, se le asigna un superpar que no tiene NAT y Alicia inicia una sesión con ese superpar. (Puesto que es Alicia quien inicia la sesión, su NAT permite la sesión.) Esta sesión hace posible que Alicia y su superpar intercambien mensajes de control. Lo mismo sucede con Benito cuando se registra. Ahora, cuando Alicia quiera llamar a Benito, informa a su superpar, que a su vez informa al superpar de Benito, que se encarga de informar a Benito de la llamada entrante de Alicia. Si Benito acepta la llamada, los dos superpares seleccionan un tercer superpar sin NAT

—el par retransmisor—, cuyo trabajo consistirá en retransmitir datos entre Alicia y Benito. A continuación, los superpares de Alicia y Benito les instruyen respectivamente para que inicien una sesión con el retransmisor. Como se muestra en la Figura 9.7, Alicia envía a partir de ese momento paquetes de voz al retransmisor a través de la conexión Alicia-retransmisor (que fue iniciada por Alicia) y el retransmisor reenvía esos paquetes a través de la conexión retransmisor-Benito (que fue iniciada por Benito); los paquetes que Benito envía a Alicia fluyen en sentido inverso a través de estas mismas dos conexiones de retransmisión. ¡Y voila! Benito y Alicia disponen de una conexión extremo a extremo, aun cuando ninguno de ellos puede aceptar una sesión que tenga su origen fuera de su red doméstica.

Hasta ahora, nuestro repaso de Skype se ha centrado en las llamadas en las que están involucradas dos personas. Examinemos ahora qué sucede en las llamadas de audioconferencia con múltiples interlocutores. Con $N > 2$ participantes, si cada usuario tuviera que enviar una copia de su flujo de audio a cada uno de los otros $N - 1$ usuarios, entonces haría falta enviar a la red un total de $N(N - 1)$ flujos de audio para soportar la audioconferencia. Para reducir el ancho de banda necesario, Skype emplea una inteligente técnica de distribución. En concreto, cada usuario envía su flujo de audio al iniciador de la conferencia. El iniciador de la conferencia combina entonces los flujos de audio en un único flujo (básicamente, sumando todas las señales de audio) y luego envía una copia de ese flujo combinado a cada uno de los otros $N - 1$ participantes. De esta forma, el número de flujos se reduce a $2(N - 1)$. Para las conversaciones de vídeo normales entre dos personas, Skype enruta la llamada de par a par, a menos que haga falta el NAT traversal, en cuyo caso la llamada se retransmite a través de un par que no tenga NAT, como hemos descrito anteriormente. Para una llamada de videoconferencia con $N > 2$ participantes, debido a la naturaleza de medio de vídeo, Skype no hace lo que en el caso de las llamadas de voz: combinar la llamada en un único flujo en una determinada ubicación y luego redistribuir el flujo a todos los participantes. En lugar de ello, el flujo de vídeo de cada participante se enruta hacia un cluster de servidores (que en 2011 estaba ubicado en Estonia), que a su vez retransmite a cada participante los $N - 1$ flujos de los otros $N - 1$ participantes [Zhang X 2012]. Puede que el lector se esté preguntando por qué cada participante

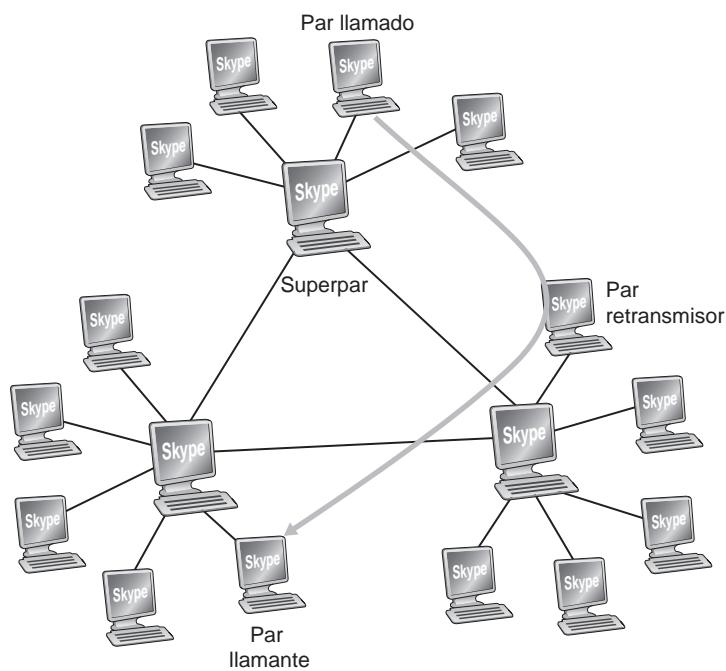


Figura 9.7 ♦ Pares Skype.

envía una copia a un servidor, en lugar de enviar directamente una copia de su flujo de vídeo a cada uno de los otros $N - 1$ participantes. Por supuesto, con ambas soluciones, el número de flujos de vídeo colectivamente recibidos por los N participantes en la conferencia será $N(N - 1)$. La razón de que se utilice la solución basada en servidor es que los anchos de banda de los enlaces de subida son significativamente más bajos que los de los enlaces de bajada en la mayoría de los enlaces de acceso, por lo que los enlaces de subida podrían no ser capaces de soportar los $N - 1$ flujos de subida que hacen falta con la solución P2P.

Los sistemas VoIP como Skype, WeChat y Google Talk, introducen nuevos problemas de privacidad. En concreto, cuando Alicia y Benito se comunican a través de VoIP, Alicia puede examinar la dirección IP de Benito y luego usar servicios de geolocalización [MaxMind 2016; Quova 2016] para determinar la posición actual y el ISP de Benito (por ejemplo, el ISP de su oficina o de su casa). De hecho, con Skype, Alicia puede bloquear la transmisión de ciertos paquetes durante el establecimiento de la llamada, para obtener la dirección IP actual de Benito cada hora, por ejemplo, sin que Benito sepa que está siendo controlado y sin aparecer en la lista de contactos de Benito. Además, la dirección IP descubierta con Skype puede ser correlacionada con las direcciones IP encontradas en BitTorrent, de modo que Alicia puede determinar los archivos que Benito se está descargando [LeBlond 2011]. Por último, resulta posible descifrar parcialmente una llamada Skype haciendo un análisis de tráfico de los tamaños de los paquetes que componen un flujo [White 2011].

9.4 Protocolos para aplicaciones de conversación en tiempo real

Las aplicaciones de conversación en tiempo real, incluidas VoIP y la videoconferencia, son atractivas y muy populares. Por tanto, no debe sorprendernos que organismos de estandarización como IETF e ITU hayan estado ocupados durante muchos años (¡y continúan estándolo!) en el establecimiento de estándares para esta clase de aplicaciones. Disponiendo de los estándares apropiados para las aplicaciones de conversación en tiempo real, empresas independientes están creando nuevos productos que interoperan entre sí. En esta sección vamos a examinar RTP y SIP para aplicaciones de conversación en tiempo real. Ambos estándares disfrutan de una amplia implementación en productos comerciales.

9.4.1 RTP

En la sección anterior hemos visto que el lado emisor de una aplicación VoIP añade campos de cabecera a los fragmentos de audio antes de pasarlos a la capa de transporte. Estos campos de cabecera incluyen números de secuencia y marcas de tiempo. Dado que la mayoría de las aplicaciones multimedia en red pueden hacer uso de los números de secuencia y de las marcas de tiempo, es conveniente disponer de una estructura de paquete estandarizada que incluya campos para los datos de audio/vídeo, los números de secuencia y las marcas de tiempo, así como otros campos potencialmente útiles. RTP, definido en el documento RFC 3550, es uno de esos estándares. RTP puede emplearse para transportar formatos comunes como PCM, ACC y MP3 para sonido, y MPEG y H.263 para vídeo. También se puede utilizar para transportar formatos de sonido y vídeo propietarios. Actualmente RTP disfruta de una amplia implementación en múltiples productos y prototipos de investigación. Además es complementario de otros importantes protocolos interactivos de tiempo real, como SIP.

En esta sección proporcionamos una introducción a RTP. Animamos a los lectores a visitar el sitio de Henning Schulzrinne dedicado a RTP [Schulzrinne- RTP 2012], que proporciona abundante información sobre el tema. También pueden visitar el sitio web de RAT [RAT 2012], que documenta una aplicación VoIP que emplea RTP.

Fundamentos de RTP

Normalmente, RTP se ejecuta sobre UDP. El lado emisor encapsula un fragmento multimedia dentro de un paquete RTP, luego encapsula ese paquete en un segmento UDP y después pasa el segmento a IP. El lado receptor extrae el paquete RTP del segmento UDP; a continuación, extrae el fragmento multimedia del paquete RTP y lo pasa al reproductor multimedia para su decodificación y reproducción.

Por ejemplo, considere el uso de RTP para transportar voz. Suponga que el origen de voz está codificado en PCM (es decir, la voz está muestreada, cuantizada y digitalizada) a 64 kbps. Suponga también que la aplicación recopila los datos codificados en fragmentos de 20 milisegundos; es decir, un fragmento tiene 160 bytes. El lado emisor precede a cada fragmento de datos de audio con una **cabecera RTP** que incluye el tipo de codificación audio, un número de secuencia y una marca de tiempo. La cabecera RTP normalmente tiene 12 bytes. El fragmento de audio y la cabecera RTP forman el **paquete RTP**. El paquete RTP se envía entonces a la interfaz de sockets UDP. En el lado receptor, la aplicación recibe el paquete RTP procedente de su interfaz de sockets. La aplicación extrae el fragmento de audio del paquete RTP y utiliza los campos de cabecera del mismo para decodificar y reproducir apropiadamente el fragmento de audio.

Si una aplicación incorpora RTP (en lugar de un esquema propietario que especifique el tipo de carga útil, los números de secuencia o las marcas de tiempo), entonces la aplicación interoperará más fácilmente con otras aplicaciones multimedia en red. Por ejemplo, si dos empresas distintas desarrollan software VoIP y ambas incorporan en su producto el protocolo RTP, existirá la posibilidad de que un usuario que emplee uno de esos productos VoIP pueda comunicarse con otro usuario que use el producto de la otra empresa. En la Sección 9.4.2 veremos que RTP suele utilizarse junto con SIP, un importante estándar para telefonía por Internet.

Debemos destacar que RTP no proporciona ningún mecanismo para garantizar la entrega a tiempo de los datos ni ninguna otra garantía de calidad del servicio (QoS, *Quality-of-Service*); ni siquiera garantiza la entrega de los paquetes, ni evita la entrega de paquetes desordenados. De hecho, la encapsulación RTP sólo se percibe en los sistemas terminales. Los routers no diferencian entre los datagramas IP que transportan paquetes RTP y los que no.

RTP permite que a cada origen (por ejemplo, una cámara o un micrófono) se le asigne su propio flujo independiente de paquetes RTP. Por ejemplo, para una videoconferencia entre dos participantes, podrían abrirse cuatro flujos RTP: dos flujos para transmitir el audio (uno en cada dirección) y dos flujos para transmitir el vídeo (también uno en cada dirección). Sin embargo, muchas técnicas de codificación populares (entre las que se incluyen MPEG 1 y MPEG 2) empaquetan el audio y el vídeo en un mismo flujo durante el proceso de codificación. Cuando el codificador empaqueta el audio y el vídeo, entonces sólo se genera un flujo RTP en cada dirección.

Los paquetes RTP no están limitados a las aplicaciones de unidifusión. También pueden enviarse a través de árboles de multidifusión uno-a-muchos y muchos-a-muchos. En una sesión de multidifusión muchos-a-muchos, normalmente todos los emisores y orígenes de la sesión utilizan el mismo grupo de multidifusión para enviar sus flujos RTP. Estos flujos multidifusión RTP conjuntos, como por ejemplo los flujos de audio y de vídeo procedentes de múltiples emisores en una aplicación de videoconferencia, pertenecen a una **sesión RTP**.

Campos de cabecera de los paquetes RTP

Como se muestra en la Figura 9.8, los cuatro campos principales de la cabecera de un paquete RTP son el tipo de carga útil, el número de secuencia, la marca de tiempo y el identificador de origen.

El campo Tipo de carga útil del paquete RTP tiene una longitud de 7 bits. En un flujo de audio, el campo Tipo de carga útil se emplea para indicar el tipo de codificación de audio (por ejemplo, PCM, modulación delta adaptativa, codificación predictiva lineal) que se está utilizando. Si un emisor decide cambiar el tipo de codificación en mitad de una sesión, puede informar al receptor de dicho cambio a través de este campo que define el tipo de carga útil. El emisor puede desear cambiar la codificación con el fin de aumentar la calidad del audio o para disminuir la tasa de bits del flujo RTP.

Tipo de carga útil	Número de secuencia	Marca de tiempo	Identificador de origen de sincronización	Campos misceláneos
--------------------	---------------------	-----------------	---	--------------------

Figura 9.8 ◆ Campos de la cabecera RTP.

En la Tabla 9.2 se enumeran algunos de los tipos de carga útil de audio a los que actualmente da soporte RTP.

Para un flujo de vídeo, el tipo de carga útil se utiliza para indicar el tipo de codificación de vídeo (por ejemplo, JPEG con movimiento, MPEG 1, MPEG 2, H.261). De nuevo, el emisor puede cambiar el tipo de codificación de vídeo sobre la marcha durante una sesión. En la Tabla 9.3 se enumeran algunos de los tipos de carga útil de vídeo soportados actualmente por RTP. Los restantes campos importantes son los siguientes:

- *Campo de número de secuencia.* El campo de número de secuencia tiene una longitud de 16 bits. El número de secuencia aumenta en una unidad para cada paquete RTP enviado y puede ser utilizado por el receptor para detectar pérdidas de paquetes y restaurar la secuencia de paquetes. Por ejemplo, si el lado receptor de la aplicación recibe un flujo de paquetes RTP con un hueco entre los números de secuencia 86 y 89, entonces el receptor sabe que faltan los paquetes 87 y 88. El receptor puede entonces intentar ocultar los datos perdidos.
- *Campo de marca de tiempo.* El campo de marca de tiempo tiene una longitud de 32 bits y refleja el instante de muestreo del primer byte del paquete de datos RTP. Como hemos visto en la sección anterior, el receptor puede utilizar las marcas de tiempo para eliminar la fluctuación de los paquetes introducida por la red y para proporcionar una reproducción síncrona en el receptor. La marca de tiempo se obtiene de una señal de reloj de muestreo del emisor. Por ejemplo, para audio, la señal de reloj de marca de tiempo se incrementa en una unidad para cada periodo de muestreo (por ejemplo, cada 125 microsegundos para una señal de reloj de muestreo de 8 kHz); si la aplicación de audio genera fragmentos que constan de 160 muestras codificadas, entonces la marca de tiempo se incrementa en 160 para cada paquete RTP cuando el origen está activo. La señal de reloj de marca de tiempo continúa aumentando a una velocidad constante incluso aunque el origen esté inactivo.
- *Identificador del origen de sincronización (SSRC, Synchronization source identifier).* El campo SSRC tiene una longitud de 32 bits. Este campo identifica el origen del flujo RTP. Normalmente, cada flujo de una sesión RTP tiene un SSRC distinto. El SSRC no es la dirección IP del emisor, sino un número que el origen asigna aleatoriamente cuando se inicia un nuevo flujo. La probabilidad de que se les asigne a dos flujos el mismo SSRC es muy pequeña. En caso de que esto ocurriera, los dos orígenes seleccionan un nuevo valor de SSRC.

Número de tipo de carga útil	Formato de audio	Frecuencia de muestreo	Velocidad
0	PCM μ-law	8 kHz	64 kbps
1	1016	8 kHz	4,8 kbps
3	GSM	8 kHz	13 kbps
7	LPC	8 kHz	2,4 kbps
9	G.722	16 kHz	48–64 kbps
14	MPEG Audio	90 kHz	—
15	G.728	8 kHz	16 kbps

Tabla 9.2 ◆ Tipos de carga útil de audio soportados por RTP.

Número de tipo de carga útil	Formato de vídeo
26	JPEG con movimiento
31	H.261
32	MPEG 1 vídeo
33	MPEG 2 vídeo

Tabla 9.3 ♦ Algunos tipos de carga útil de vídeo soportados por RTP.

9.4.2 SIP

El protocolo SIP (*Session Initiation Protocol*, protocolo de iniciación de sesión), definido en [RFC 3261; RFC 5411], es un protocolo abierto y ligero que hace lo siguiente:

- Proporciona mecanismos para establecer llamadas entre el llamante y el llamado a través de una red IP. Permite al llamante notificar al llamado que desea iniciar una comunicación. Permite a los participantes acordar los métodos de codificación de los datos multimedia, así como dar por terminadas las llamadas.
- Proporciona mecanismos al llamante para determinar la dirección IP actual del llamado. Los usuarios no tienen una única dirección IP fija, porque se les pueden asignar dinámicamente direcciones (mediante DHCP) y porque pueden tener múltiples dispositivos IP, cada uno de ellos con una dirección IP diferente.
- Proporciona mecanismos para la gestión de llamadas, tales como añadir nuevos flujos multimedia durante la llamada, cambiar el método de codificación o invitar a nuevos participantes mientras tiene lugar la llamada, además de mecanismos para la transferencia de llamadas y la puesta en espera de las mismas.

Establecimiento de una llamada con una dirección IP conocida

Para comprender la esencia de SIP, lo mejor es analizar un ejemplo concreto. En este ejemplo, Alicia se encuentra delante de su PC y desea llamar a Benito, que también está trabajando con su computadora. Los PC de Alicia y de Benito están equipados con software basado en SIP que les permite hacer y recibir llamadas telefónicas. En este ejemplo inicial, supondremos que Alicia conoce la dirección IP de la computadora de Benito. En la Figura 9.9 se ilustra el proceso de establecimiento de llamadas SIP.

En la Figura 9.9 vemos que una sesión SIP se inicia cuando Alicia envía a Benito un mensaje INVITE, que es parecido a un mensaje de solicitud HTTP. Este mensaje INVITE se envía sobre UDP al puerto bien conocido 5060 para SIP. (Los mensajes SIP también se pueden enviar sobre TCP.) El mensaje INVITE incluye un identificador para Benito (benito@193.64.210.89), una indicación de la dirección IP actual de Alicia, una indicación de que Alicia desea recibir audio, el cual debe codificarse en formato AVP 0 (codificación PCM μ-law) y encapsularse en RTP, y una indicación de que Alicia desea recibir los paquetes RTP a través del puerto 38060. Después de recibir el mensaje INVITE de Alicia, Benito envía un mensaje de respuesta SIP, que es parecido a un mensaje de respuesta HTTP. Este mensaje de respuesta SIP también se envía al puerto SIP 5060. La respuesta de Benito incluye un mensaje 200 OK y una indicación de su dirección IP, la forma de codificación y empaquetamiento que desea para recepción y su número de puerto al que deben enviarse los paquetes de audio. Observe que, en este ejemplo, Alicia y Benito van a emplear diferentes mecanismos de codificación de audio: a Alicia se le solicita que codifique su audio con GSM mientras que a Benito se le pide que codifique su audio con PCM μ-law. Una vez recibida la respuesta de Benito, Alicia envía a Benito un mensaje de confirmación SIP. Después de esta transacción SIP, Benito y Alicia pueden hablar. (Por comodidad visual, la Figura 9.9

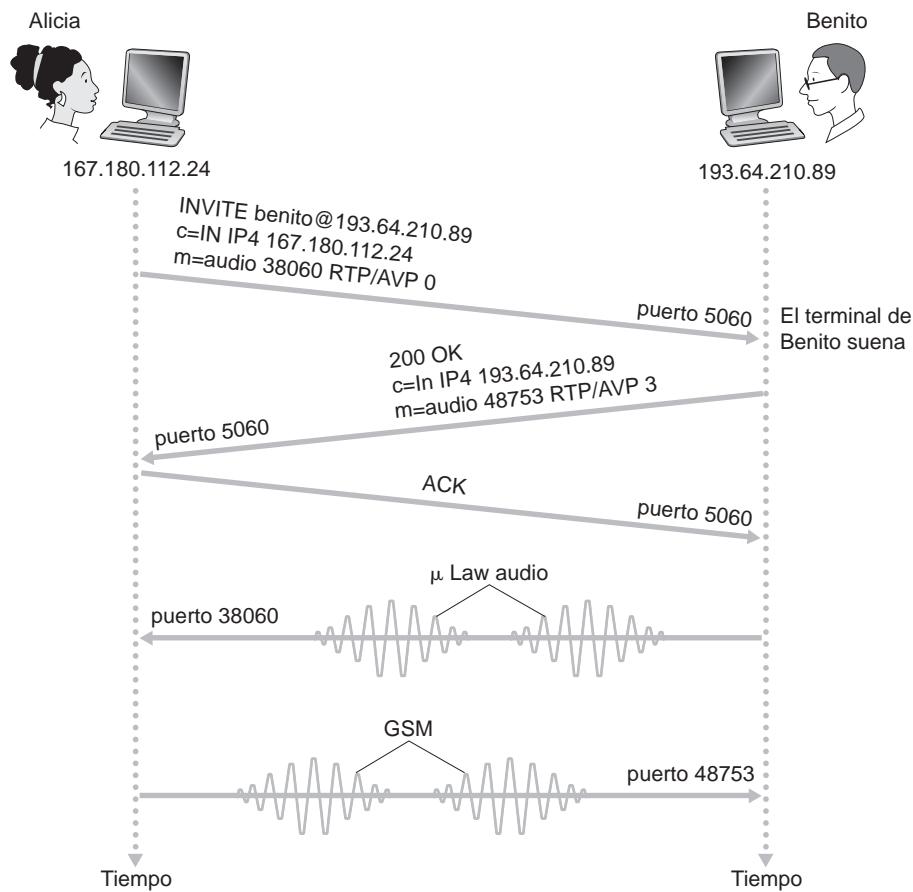


Figura 9.9 ◆ Establecimiento de una llamada SIP cuando Alicia conoce la dirección IP de Benito.

muestra que Alicia habla después de Benito, pero en la realidad normalmente hablarían al mismo tiempo.) Benito codificará y empaquetará el audio de la forma requerida y enviará los paquetes de audio al número de puerto 38060 en la dirección IP 167.180.112.24. Alicia también codificará y empaquetará el audio en el formato solicitado y enviará los paquetes de audio al número de puerto 48753 en la dirección IP 193.64.210.89.

Con este sencillo ejemplo hemos aprendido una serie de características clave de SIP. En primer lugar, SIP es un protocolo fuera de banda: los mensajes SIP se envían y se reciben a través de sockets diferentes de los utilizados para enviar y recibir los datos multimedia. En segundo lugar, los propios mensajes SIP son mensajes legibles ASCII y se parecen a los mensajes HTTP. Por último, SIP requiere que todos los mensajes sean confirmados, por lo que se puede ejecutar sobre UDP o sobre TCP.

Siguiendo con este ejemplo, ahora vamos a considerar lo que ocurriría si Benito no dispone de un codec PCM μ-law para codificar el audio. En este caso, en lugar de responder con 200 OK, Benito probablemente respondería con un mensaje 606 Not Acceptable e incluiría en el mensaje todos los codecs que puede utilizar. Alicia elegiría entonces uno de los codecs de la lista y enviaría otro mensaje INVITE, anunciando en esta ocasión el codec elegido. Benito también podría simplemente rechazar la llamada, enviando uno de los muchos posibles códigos de respuesta de rechazo. (Existen muchos códigos de este tipo, entre los que se incluyen “ocupado”, “ausente”, “pago requerido” y “prohibido”.)

Direcciones SIP

En el ejemplo anterior, la dirección SIP de Benito es `sip:benito@193.64.210.89`. Sin embargo, cabe esperar que muchas (si no la mayoría) de direcciones SIP se parezcan a direcciones de correo electrónico. Por ejemplo, la dirección de Benito podría ser `sip:benito@dominio.com`. Cuando el dispositivo SIP de Alicia envía un mensaje INVITE, el mensaje incluiría esta dirección similar a una dirección de correo electrónico; la infraestructura SIP enrutaría entonces el mensaje al dispositivo IP que Benito esté utilizando en ese momento (como veremos más adelante). Otro posible formato de la dirección SIP podría ser el número de teléfono tradicional de Benito o simplemente su nombre y apellidos (suponiendo siempre que fueran distintivos).

Una característica interesante de las direcciones SIP es que pueden incluirse en páginas web, al igual que se incluyen las direcciones de correo electrónico en las páginas web, con el URL mailto. Por ejemplo, suponga que Benito tiene una página web personal y que desea proporcionar un medio a los visitantes de su página para que le llamen. Benito puede simplemente incluir el URL `sip:benito@dominio.com` y, de este modo, cuando un visitante haga clic en el URL, la aplicación SIP del dispositivo del visitante se ejecuta y envía un mensaje INVITE a Benito.

Mensajes SIP

En esta breve introducción al protocolo SIP no vamos a tratar todas las cabeceras y tipos de mensajes SIP. En lugar de ello, vamos a examinar brevemente el mensaje SIP INVITE, junto con unas pocas líneas de cabecera comunes. Supongamos de nuevo que Alicia desea realizar una llamada VoIP a Benito y que en esta ocasión Alicia solo conoce la dirección SIP de Benito, `benito@dominio.com`, y no la dirección IP del dispositivo que Benito está utilizando actualmente. Entonces, el mensaje de Alicia sería similar al siguiente:

```
INVITE sip:benito@dominio.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alicia@hereway.com
To: sip:benito@dominio.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

La línea INVITE incluye la versión de SIP, al igual que en un mensaje de solicitud HTTP. Cada vez que un mensaje SIP atraviesa un dispositivo SIP (incluyendo el dispositivo que origina el mensaje), el dispositivo añade una línea de cabecera Via, que indica la dirección IP del dispositivo. (Veremos enseguida que el mensaje INVITE típico atraviesa muchos dispositivos SIP antes de llegar a la aplicación SIP del llamado.) De forma similar a un mensaje de correo electrónico, el mensaje SIP incluye una línea de cabecera From y una línea de cabecera To. El mensaje incluye también la línea de cabecera Call-ID, que identifica de forma única la llamada (de forma similar al message-ID de un correo electrónico). También incluye la línea de cabecera Content-Type, que define el formato utilizado para describir el contenido del mensaje SIP, e incluye la línea de cabecera Content-Length, que proporciona la longitud en bytes del contenido del mensaje. Por último, después de un retorno de carro y un salto de línea, se incluye el contenido del mensaje. En este caso, el contenido proporciona información acerca de la dirección IP de Alicia y de cómo desea ésta recibir el audio.

Traducción de nombres y localización de usuarios

En el ejemplo de la Figura 9.9 hemos supuesto que el dispositivo SIP de Alicia conocía la dirección IP en la que podría contactar a Benito. Pero esta suposición es bastante poco realista, no solo porque

las direcciones IP a menudo son asignadas dinámicamente mediante DHCP, sino también porque Benito puede disponer de varios dispositivos IP (por ejemplo, diversos dispositivos en su domicilio, en el trabajo y en el automóvil). Por tanto, vamos a suponer ahora que Alicia sólo conoce la dirección de correo electrónico de Benito, benito@dominio.com, y que esta misma dirección se utiliza para las llamadas basadas en SIP. En este caso, Alicia necesita obtener la dirección IP del dispositivo que está utilizando actualmente el usuario benito@dominio.com. Para averiguarlo, Alicia crea un mensaje INVITE que comienza con INVITE benito@dominio.com SIP/2.0 y envía este mensaje a un **proxy SIP**. El proxy contestará con una respuesta SIP que podría incluir la dirección IP del dispositivo que actualmente está utilizando benito@dominio.com. De forma alternativa, la respuesta podría incluir la dirección IP del buzón de voz de Benito, o podría incluir un URL de una página web (que diga “¡Benito está durmiendo. No molestar!”). Además, el resultado devuelto por el proxy podría depender de quién hace la llamada: si la llamada es de la esposa de Benito, podría aceptar la llamada y suministrar su dirección IP; si la llamada la hace la suegra de Benito, podría responder con el URL que apunta a la página web de ¡Estoy durmiendo!

Es posible que en este momento se esté preguntando cómo puede el servidor proxy determinar la dirección IP actual para benito@dominio.com. Para responder a esta pregunta, primero tenemos que decir algunas cosas acerca de otro dispositivo SIP: el **registrador SIP**. Todos los usuarios SIP tienen un registrador asociado. Cuando un usuario ejecuta una aplicación SIP en un dispositivo, la aplicación envía un mensaje de registro SIP al registrador, informándole de su dirección IP actual. Por ejemplo, cuando Benito ejecuta su aplicación SIP en su PDA, la aplicación enviará un mensaje similar a:

```
REGISTER sip:dominio.com SIP/2.0
Via: SIP/2.0/UDP 193.64.210.89
From: sip:benito@dominio.com
To: sip:benito@dominio.com
Expires: 3600
```

El registrador de Benito lleva la cuenta de su dirección IP actual. Cuando Benito cambia a un dispositivo SIP nuevo, éste envía un nuevo mensaje de registro, indicando la nueva dirección IP. Además, si Benito permanece en el mismo dispositivo durante un periodo de tiempo largo, el dispositivo enviará mensajes de registro de refresco, indicando que la dirección IP enviada más recientemente continúa siendo válida. (En el ejemplo anterior, los mensajes de refresco necesitan ser enviados cada 3.600 segundos con el fin de mantener la dirección en el servidor registrador.) Merece la pena comentar que el registrador es análogo a un servidor de nombres DNS autoritativo: el servidor DNS traduce los nombres fijos de host a direcciones IP fijas y el registrador SIP traduce los identificadores fijos de personas (por ejemplo, benito@dominio.com) a direcciones IP dinámicas. A menudo, los registradores SIP y los proxies SIP se ejecutan en el mismo host.

Examinemos ahora cómo el servidor proxy de Alicia obtiene la dirección IP actual de Benito. En la exposición anterior hemos visto que el servidor proxy simplemente necesita reenviar el mensaje INVITE de Alicia al registrador/proxy de Benito. El registrador/proxy podría entonces reenviar el mensaje al dispositivo SIP actual de Benito. Por último, una vez que Benito ha recibido el mensaje INVITE de Alicia, podría enviar una respuesta SIP a Alicia.

Por ejemplo, considere la Figura 9.10, en la que juan@umass.edu, que actualmente está trabajando en 217.123.56.89, desea iniciar una sesión de Voz sobre IP (VoIP) con catalina@upenn.edu, que se encuentra en este momento trabajando en 197.87.54.21. Los pasos que se llevan a cabo son los siguientes: (1) Juan envía un mensaje INVITE al proxy SIP de umass. (2) El proxy realiza una búsqueda DNS del registrador SIP upenn.edu (no mostrado en el diagrama) y luego reenvía el mensaje al servidor registrador. (3) Puesto que catalina@upenn.edu ya no está registrada en el registrador upenn, éste envía una respuesta de redirección que indica que debería probar con catalina@nyu.edu. (4) El proxy umass envía un mensaje INVITE al registrador SIP de NYU. (5) El registrador de NYU conoce la dirección IP de catalina@nyu.edu y reenvía el mensaje INVITE al host 197.87.54.21, que está ejecutando el cliente SIP de Catalina. (6-8) Se devuelve al cliente SIP

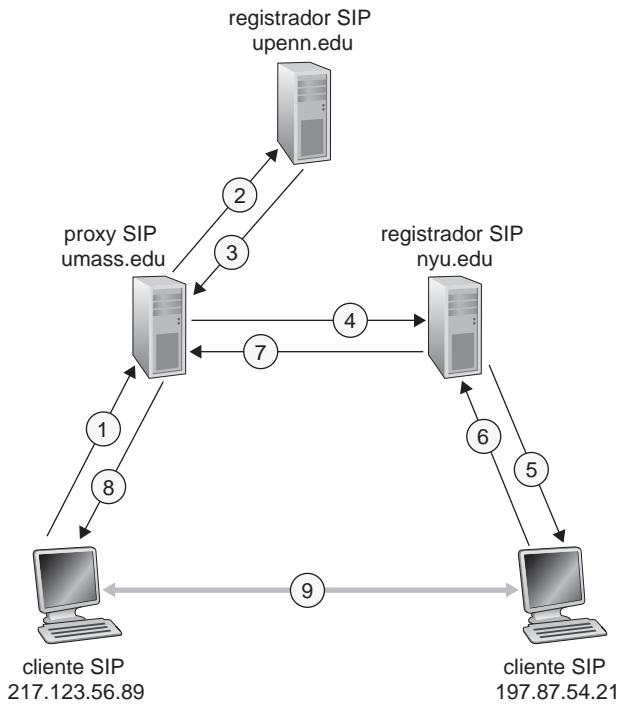


Figura 9.10 ♦ Iniciación de sesión que implica proxies y registradores SIP.

217.123.56.89 una respuesta SIP a través de los registradores/proxies. (9) Los datos multimedia se intercambian directamente entre los dos clientes. (También existe un mensaje de confirmación SIP, que no se muestra en la figura.)

Nuestra exposición acerca de SIP se ha centrado en el proceso de iniciación de llamada para el caso de llamadas de voz. SIP, al ser un protocolo de señalización para el inicio y la terminación de llamadas en general, puede utilizarse tanto para videoconferencias, como para sesiones basadas en texto. De hecho, SIP se ha convertido en un componente fundamental en muchas aplicaciones de mensajería instantánea. Animamos a los lectores que deseen aprender más acerca de SIP a visitar el sitio web de Henning Schulzrinne [Schulzrinne-SIP 2016]. En particular, en este sitio encontrará software de código abierto para clientes y servidores SIP [SIP Software 2016].

9.5 Soporte de red para aplicaciones multimedia

En las Secciones 9.2 a 9.4, hemos visto que las aplicaciones multimedia pueden utilizar mecanismos de nivel de aplicación —como los buffers de cliente, la precarga, la adaptación de la calidad de los datos multimedia al ancho de banda disponible, la reproducción adaptativa y las técnicas de mitigación de pérdidas— para mejorar su propio rendimiento. También vimos que pueden utilizarse redes de distribución de contenido y redes P2P solapadas para proporcionar una solución de *nivel de sistema* para la distribución de contenido multimedia. Todas estas técnicas y soluciones están diseñadas para usarse en la Internet actual, con su servicio de entrega de mejor esfuerzo. De hecho, si se utilizan hoy en día es, precisamente, porque Internet solo proporciona una única clase de servicio, con entrega de mejor esfuerzo. Pero como diseñadores de redes de computadoras, no podemos evitar preguntarnos si la *red* (en vez de solo las aplicaciones o la infraestructura de nivel de aplicación) podría proporcionar mecanismos para soportar la distribución de contenido multimedia. Como pronto veremos, la respuesta es, por supuesto, “¡Sí!”. Pero también veremos que aun no están

ampliamente implantados varios de estos nuevos mecanismos de nivel de red. Esto puede deberse a su complejidad y al hecho de que las técnicas de nivel de aplicación, junto con el servicio de entrega de mejor esfuerzo y unos recursos de red adecuadamente dimensionados (por ejemplo, el ancho de banda), pueden ciertamente proporcionar un servicio de entrega multimedia extremo a extremo “suficientemente bueno” (aunque no siempre sea perfecto).

La Tabla 9.4 resume tres enfoques generales para proporcionar soporte de nivel de red a las aplicaciones multimedia.

- *Sacar el máximo partido del servicio de entrega de mejor esfuerzo.* Los mecanismos de nivel de aplicación y la infraestructura que hemos estudiado en las Secciones 9.2 a 9.4, pueden usarse satisfactoriamente en una red bien dimensionada, en la que solo se produzcan raramente pérdidas de paquetes y retardos extremo a extremo excesivos. Cuando se prevén incrementos de la demanda, los ISP implantan ancho de banda y capacidad de commutación adicionales, para continuar garantizando unas prestaciones satisfactorias en lo referente a los retardos y a la pérdida de paquetes [Huang 2005]. Hablaremos más en detalle de ese **dimensionamiento de la red** en la Sección 9.5.1.
- *Servicio diferenciado.* Desde los primeros tiempos de Internet, ha estado rondando la idea de proporcionar diferentes clases de servicio a los distintos tipos de tráfico (por ejemplo, según se indique en el campo Tipo de servicio de la cabecera de los paquetes IPv4), en lugar de proporcionar un único servicio común de entrega de mejor esfuerzo. Con el **servicio diferenciado**, se puede dar prioridad estricta a un tipo de tráfico con respecto a otra clase de tráfico, cuando ambos tipos de tráfico estén en cola en un router. Por ejemplo, los paquetes pertenecientes a una aplicación de conversación en tiempo real podrían ser prioritarios con respecto a otros paquetes, debido a sus estrictas restricciones de retardo. Introducir servicios diferenciados en la red requeriría nuevos mecanismos para el marcado de paquetes (para indicar la clase de servicio del paquete), para la planificación de paquetes y otros. Hablaremos del servicio diferenciado, y de los nuevos mecanismos de red necesarios para implementar este servicio, en las Secciones 9.5. y 9.5.3.
- *Garantías de calidad de servicio (QoS) por conexión.* Con las garantías de calidad de servicio (QoS) por conexión, cada instancia de una aplicación reserva explícitamente ancho de banda extremo a extremo y goza, por tanto, de unas prestaciones extremo a extremo garantizadas. Una **garantía estricta** quiere decir que la aplicación recibirá la calidad de servicio (QoS) solicitada con absoluta seguridad. Una **garantía parcial** quiere decir que la aplicación recibirá su calidad de servicio solicitada con una alta probabilidad. Por ejemplo, si un usuario desea hacer una llamada VoIP desde el Host A al Host B, entonces la aplicación VoIP del usuario reserva ancho de banda de forma explícita en cada uno de los enlaces de una ruta que conecta a ambos hosts. Pero permitir

Enfoque	Granularidad	Garantía	Mecanismos	Complejidad	Implantación hasta la fecha
Sacar el máximo partido del servicio de entrega de mejor esfuerzo	Todo el tráfico se trata igual	Ninguna o parcial	Soporte de la capa de aplicación, redes CDN, redes solapadas, provisión de recursos de nivel de red	Mínima	En todas partes
Servicio diferenciado	Se trata de forma diferente a las distintas clases de tráfico	Ninguna o parcial	Marcado de paquetes, vigilancia, planificación	Media	Alguna
Garantías de calidad de servicio (QoS) por conexión	Se trata de forma diferente a cada flujo origen-destino	Parcial o estricta, una vez admitido el flujo	Marcado de paquetes, vigilancia, planificación; admisión y señalización de llamadas	Alta	Poca

Tabla 9.4 ♦ Tres soluciones de nivel de red para dar soporte a las aplicaciones multimedia.

que las aplicaciones hagan reservas y exigir a la red que de satisfacción a esas reservas requiere algunos grandes cambios. En primer lugar, necesitamos un protocolo que, en nombre de las aplicaciones, reserve ancho de banda en los enlaces que componen las rutas desde los emisores hasta sus respectivos receptores. En segundo lugar, habrá que modificar las políticas de planificación en las colas de los routers, de modo que se cumpla con las reservas de ancho de banda de cada conexión. Finalmente, para hacer una reserva, las aplicaciones tienen que proporcionar a la red una descripción del tráfico que pretenden enviar, y la red tendrá entonces que vigilar el tráfico de cada aplicación para garantizar que cumple con la descripción proporcionada. Estos mecanismos, cuando se combinan, requieren un software nuevo y complejo tanto en los hosts como en los routers. Puesto que el servicio de QoS garantizada por conexión todavía no se ha implantado de manera significativa, solo veremos estos mecanismos brevemente en la Sección 9.5.4.

9.5.1 Dimensionamiento de las redes con servicio de entrega de mejor esfuerzo

Fundamentalmente, las dificultades para dar soporte a las aplicaciones multimedia surgen de sus estrictos requisitos de rendimiento (bajo retardo de paquetes extremo a extremo, baja fluctuación del retardo y baja tasa de pérdidas) y del hecho de que el retardo de los paquetes, la fluctuación de los retardos y las pérdidas se producen cuando la red está congestionada. Una primera técnica para mejorar la calidad de las aplicaciones multimedia (una técnica que a menudo puede emplearse para resolver prácticamente cualquier problema provocado por la restricción de los recursos) es simplemente “meter dinero” y evitar desde el principio que exista una contienda por los recursos. En el caso de las aplicaciones multimedia en red, esto quiere decir proporcionar suficiente capacidad de enlace por toda la red, de modo que nunca se produzca (o sólo se produzca muy raramente) una congestión en la red, con los consiguientes retardos y pérdidas de paquetes. Con suficiente capacidad de enlace, los paquetes podrían atravesar la actual red Internet sin retardos de puesta en cola ni pérdidas. Desde muchos puntos de vista, ésta es una situación ideal: las aplicaciones multimedia funcionarían perfectamente, los usuarios estarían contentos y todo esto podría conseguirse sin efectuar ningún cambio en la arquitectura Internet, basada en un servicio de entrega de mejor esfuerzo.

Por supuesto, la cuestión es cuánta capacidad resulta “suficiente” para conseguir esta situación paradisiaca y si los costes de proporcionar un ancho de banda “suficiente” resultan prácticos desde el punto de vista empresarial para los ISP. La cuestión de cuánta capacidad proporcionar en los enlaces de la red para una cierta topología dada, con el fin de conseguir un determinado nivel de rendimiento, se suele denominar **provisión de ancho de banda**. El problema, todavía más complicado, de cómo diseñar una topología de red (dónde colocar los routers, cómo interconectar los routers mediante enlaces y qué capacidad asignar a los enlaces) para conseguir un nivel prefijado de rendimiento extremo a extremo, es un problema de diseño de redes que a menudo se denomina **dimensionamiento de la red**. Tanto la provisión de ancho de banda como el dimensionamiento de la red son temas complejos, que caen fuera del alcance de este libro. Sin embargo, conviene resaltar que es necesario resolver los siguientes problemas para poder predecir el rendimiento de nivel de aplicación entre dos puntos terminales de la red, y así poder provisionar una capacidad suficiente como para satisfacer los requisitos de rendimiento de una aplicación.

- *Modelos de demanda de tráfico entre puntos terminales de la red.* Los modelos pueden tener que especificarse tanto en el nivel de llamada (por ejemplo, usuarios “que llegan” a la red e inician aplicaciones extremo a extremo), como en el nivel de paquetes (por ejemplo, paquetes generados por las aplicaciones activas). Observe que la carga de trabajo puede variar a lo largo del tiempo.
- *Requisitos de rendimiento bien definidos.* Por ejemplo, un requisito para el soporte de tráfico sensible al retardo (como el de una aplicación de conversación multimedia) podría ser que la probabilidad de que el retardo extremo a extremo del paquete supere un determinado umbral máximo tolerable sea inferior a un cierto valor pequeño [Fraleigh 2003].

- *Modelos para predecir el rendimiento extremo a extremo para un determinado modelo de carga de trabajo, junto con técnicas para encontrar una asignación de coste mínimo del ancho de banda que permita satisfacer todos los requisitos de los usuarios.* En este aspecto, los investigadores están trabajando arduamente para desarrollar modelos de rendimiento que permitan cuantificar el rendimiento para una carga de trabajo determinada, junto con técnicas de optimización para hallar las asignaciones de coste mínimo del ancho de banda que satisfagan los requisitos de rendimiento.

Dado que la red Internet actual, basada en un servicio de entrega de mejor esfuerzo, podría (desde el punto de vista tecnológico) soportar tráfico multimedia con un nivel de rendimiento apropiado, si estuviera dimensionada para hacerlo, la pregunta natural es por qué la red Internet de hoy día no lo hace. Las respuestas son principalmente económicas y organizativas. Desde el punto de vista económico, ¿estarían dispuestos los usuarios a pagar a sus ISP el suficiente dinero como para que los ISP instalaran un ancho de banda suficiente para soportar aplicaciones multimedia sobre la actual Internet? Las cuestiones organizativas son quizás aún más difíciles. Observe que una ruta extremo a extremo entre dos puntos terminales de una comunicación multimedia tendrá que pasar por las redes de múltiples ISP. Desde el punto de vista organizativo, ¿estarían dispuestos estos ISP a cooperar (tal vez compartiendo los ingresos) para garantizar que la ruta extremo a extremo esté adecuadamente dimensionada como para soportar las aplicaciones multimedia? Para ver un análisis desde la perspectiva de estos problemas económicos y organizativos, consulte [Davies 2005]. Para ver un análisis de la provisión de redes troncales de nivel 1 con el fin de dar soporte a tráfico sensible al retardo, consulte [Fraleigh 2003].

9.5.2 Provisión de múltiples clases de servicio

Quizá la mejora más simple al servicio “de talla única” de la actual Internet, basado en la entrega de mejor esfuerzo, consistiría en dividir el tráfico en clases y proporcionar diferentes niveles de servicio a esas distintas clases de tráfico. Por ejemplo, un ISP puede querer proporcionar una clase de servicio mejor (y cobrar más por ese servicio!) al tráfico de teleconferencia o de Voz sobre IP (que es sensible al retardo), que al tráfico elástico, como el de FTP o HTTP. Alternativamente, un ISP puede simplemente querer proporcionar una mejor calidad de servicio a los clientes que estén dispuestos a pagar más por ese servicio mejorado. Diversos ISP de acceso residencial cableado y de acceso celular inalámbrico han adoptado ese tipo de niveles de servicio diferenciados: los abonados al servicio platino disfrutan de un mejor rendimiento que los abonados al servicio oro o plata.

Todos estamos familiarizados con las clases de servicio diferenciadas en nuestra vida cotidiana: los pasajeros de primera clase de una línea aérea obtienen un mejor servicio que los pasajeros de la clase business, quienes a su vez obtienen un mejor servicio que el que se proporciona a los pasajeros de la clase turista; las personas VIP pueden entrar de forma inmediata a los eventos, mientras que los demás tienen que esperar en fila; en algunos países, las personas mayores son reverenciadas y se les ofrecen los asientos de honor y lo más exquisito de los alimentos en la mesa. Es importante observar que tal servicio diferenciado se proporciona entre agregados de tráfico; es decir, entre clases de tráfico, no entre conexiones individuales. Por ejemplo, todos los pasajeros de primera clase son tratados igual (ningún pasajero de primera clase recibe un tratamiento mejor que cualquier otro pasajero también de primera clase), al igual que todos los paquetes VoIP recibirán el mismo tratamiento dentro de la red, independientemente de la conexión extremo a extremo concreta a la que pertenezcan. Como veremos, al tratar con un número pequeño de agregados de tráfico, en lugar de con una gran cantidad de conexiones individuales, los nuevos mecanismos de red requeridos para proporcionar un servicio de entrega más efectivo que el de mejor esfuerzo pueden ser relativamente simples.

Evidentemente, los primeros diseñadores de Internet tenían esta idea de múltiples clases de servicio en mente. Recuerde el campo Tipo de servicio (ToS) de la cabecera IPv4 que analizamos en el Capítulo 4. IEN123 [ISI 1979] describe el campo ToS también presente en un antecesor del

datagrama IPv4 de la siguiente manera: “El [campo] Tipo de servicio proporciona una indicación de los parámetros abstractos de la calidad de servicio deseada. Estos parámetros están pensados para guiar la selección de los parámetros reales del servicio al transmitir un datagrama a través de una red concreta. Varias redes ofrecen mecanismos de precedencia de servicio, que tratan en cierto modo el tráfico con alta precedencia como si fuera más importante que el resto del tráfico.” ¡Hace más de cuatro décadas, la visión de proporcionar diferentes niveles de servicio a las diferentes clases de tráfico estaba clara! Sin embargo, nos ha llevado todo ese tiempo el conseguir poner en práctica dicha visión.

Escenarios de ejemplo

Comencemos con unos cuantos ejemplos ilustrativos nuestra exposición sobre los mecanismos de red necesarios para proporcionar múltiples clases de servicio.

La Figura 9.11 muestra un escenario de red simple, en el que dos flujos de paquetes de aplicación tienen su origen en los hosts H1 y H2 de una LAN y están destinados a los hosts H3 y H4 de otra LAN. Los routers de las dos redes LAN están conectados mediante un enlace a 1,5 Mbps. Supongamos que las velocidades de las redes LAN son significativamente mayores de 1,5 Mbps y vamos a fijarnos en la cola de salida del router R1; es aquí donde se producirá el retardo y la pérdida de paquetes si la velocidad agregada de transmisión de H1 y H2 excede los 1,5 Mbps. Supongamos también que una aplicación de audio de 1 Mbps (por ejemplo, una llamada de audio con calidad de CD) comparte el enlace a 1,5 Mbps entre R1 y R2 con una aplicación HTTP de navegación web que está descargando una página desde H2 a H4.

Con el servicio de entrega del mejor esfuerzo de Internet, los paquetes de audio y HTTP se mezclan en la cola de salida de R1 y (normalmente) se transmiten siguiendo el orden de llegada (FIFO, *First-In-First-Out*). En este escenario, una ráfaga de paquetes procedentes del servidor web podría potencialmente llenar la cola, haciendo que los paquetes de audio IP se retardaran excesivamente o se perdieran a causa de un desbordamiento del buffer de R1. ¿Cómo podemos resolver este potencial problema? Dado que la aplicación HTTP de navegación web no tiene restricciones de tiempo, nuestra intuición puede llevarnos a pensar que debería proporcionarse una prioridad estricta a los paquetes de audio en R1. Aplicando esta disciplina de planificación con prioridad estricta, un paquete de audio que se encontrara en el buffer de salida de R1 siempre se transmitiría antes que cualquier paquete HTTP que se encontrara en ese mismo buffer. El enlace entre R1 y R2 sería entonces como un enlace dedicado a 1,5 Mbps para el tráfico de audio, y el tráfico HTTP emplearía dicho enlace sólo cuando no hubiera tráfico de audio en la cola. Con el fin de que R1 distinga los paquetes de

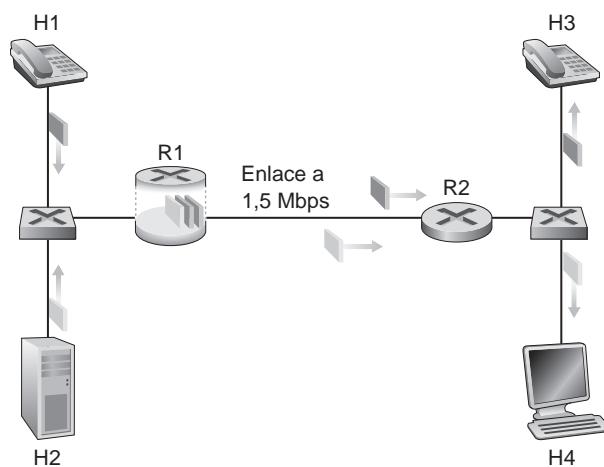


Figura 9.11 ♦ Aplicaciones de audio y HTTP compitiendo.

audio de los paquetes HTTP que tiene en su cola, cada uno de los paquetes debe marcarse como perteneciente a una de esas dos clases de tráfico. Éste era el objetivo original del campo Tipo de servicio (ToS) de IPv4. Aunque pueda parecer obvio, éste es nuestro primer principio básico en el que se fundamentan los mecanismos necesarios para proporcionar múltiples clases de tráfico:

Principio 1: El marcado de los paquetes permite a un router diferenciar entre paquetes pertenecientes a distintas clases de tráfico.

Observe que, aunque en nuestro ejemplo estamos considerando el caso de un flujo multimedia y otro elástico compitiendo entre sí, las mismas conclusiones se aplicarían en el caso de que se implementaran las clases de servicio platino, oro y plata: sigue siendo necesario un mecanismo de marcado de paquetes, para indicar la clase de servicio a la que cada paquete pertenece.

Suponga ahora que el router está configurado para dar prioridad a los paquetes marcados como pertenecientes a la aplicación de audio a 1 Mbps. Puesto que la velocidad del enlace de salida es de 1,5 Mbps, aun cuando los paquetes HTTP tengan una prioridad menor, todavía recibirán, como promedio, un servicio de transmisión de 0,5 Mbps. Pero, ¿qué ocurre si la aplicación de audio comienza a enviar paquetes a una velocidad de 1,5 Mbps o superior (bien maliciosamente o debido a un error de la aplicación)? En este caso, los paquetes HTTP sufrirán inanición, es decir, no recibirán ningún servicio en el enlace R1-R2. Podrían producirse problemas similares si varias aplicaciones (por ejemplo, varias llamadas de audio), todas ellas con la misma clase de servicio que nuestra aplicación de audio original, tuvieran que compartir el ancho de banda del enlace; también podrían provocar, colectivamente, la inanición de la sesión HTTP. Idealmente, sería deseable un cierto grado de aislamiento entre las distintas clases de tráfico, con el fin de proteger a una clase de tráfico de la otra. Esta protección podrían implementarse en diferentes lugares de la red: en todos y cada uno de los routers, en el momento de entrar en la red o en las fronteras entre dominios de la red. Nuestro segundo principio sería, por tanto:

Principio 2: Es deseable proporcionar un cierto grado de **aislamiento del tráfico** entre las distintas clases, de manera que ninguna clase se vea afectada negativamente por otra clase de tráfico que exhiba un comportamiento erróneo.

Más adelante examinaremos varios mecanismos específicos para proporcionar este aislamiento entre clases de tráfico. Debemos comentar aquí que es posible adoptar dos enfoques generales. En primer lugar, como se muestra en la Figura 9.12, es posible realizar una **vigilancia del tráfico**. Si una clase de tráfico o flujo tiene que satisfacer ciertos criterios (por ejemplo, que el flujo de audio no exceda una velocidad de pico de 1 Mbps), entonces puede utilizarse un mecanismo de vigilancia con el fin de garantizar que esos criterios son, en efecto, respetados. Si la aplicación que se está monitorizando presenta un mal comportamiento, entonces el mecanismo de vigilancia llevará a cabo una cierta acción (por ejemplo, descartar o retardar los paquetes que están violando los criterios), de modo que el tráfico que realmente entre en la red cumpla los criterios escrupulosamente. El mecanismo de goteo (llamado también de cubeta con pérdidas) que examinaremos en breve es quizás el mecanismo de vigilancia más ampliamente utilizado. En la Figura 9.12, el mecanismo de clasificación y marcado de paquetes (Principio 1) y el mecanismo de vigilancia (Principio 2) están implementados conjuntamente en la frontera de la red, bien en el sistema terminal o bien en un router de frontera.

Un enfoque complementario para proporcionar aislamiento entre clases de tráfico es que el mecanismo de planificación de paquetes a nivel de enlace asigne explícitamente a cada clase una cantidad fija de ancho de banda del enlace. Por ejemplo, a la clase de audio podría asignársele 1 Mbps en R1 y a la clase HTTP se le podría asignar 0,5 Mbps. En este caso, los flujos de audio y HTTP ven un enlace lógico con una capacidad de 1,0 y 0,5 Mbps, respectivamente, como se muestra en la Figura 9.13. Con una imposición estricta de la asignación de ancho de banda a nivel de enlace, una clase sólo puede usar la cantidad de ancho de banda que le haya sido asignada; en concreto, no puede utilizar ancho de banda que no esté siendo actualmente empleado por otros. Por ejemplo, si

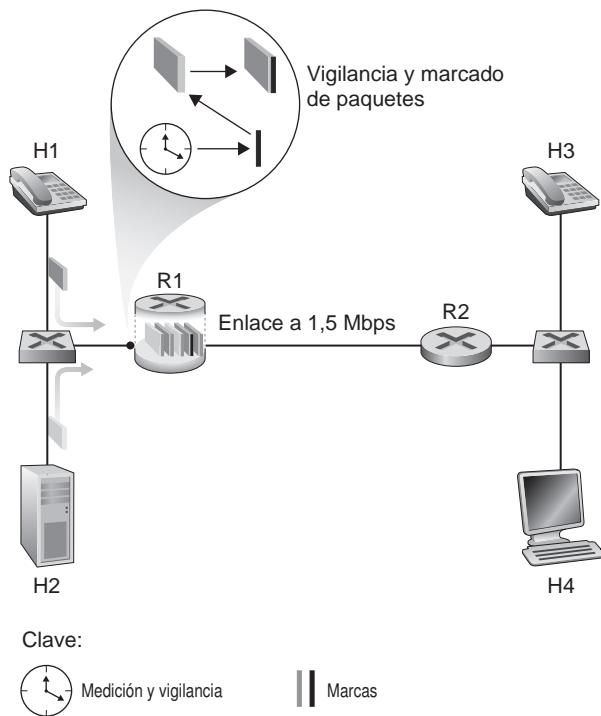


Figura 9.12 ♦ Vigilancia (y marcado) de las clases de tráfico de audio y HTTP.

el flujo de audio se silencia (por ejemplo, si el que habla hace una pausa y no genera paquetes de audio), el flujo HTTP seguirá sin poder transmitir a más de 0,5 Mbps a través del enlace R1-R2, incluso aunque la asignación de ancho de banda de 1 Mbps del flujo de audio no esté siendo utilizada en ese momento. Como el ancho de banda es un tipo de recurso que “si no se usa, se pierde”, no hay ninguna razón para impedir al tráfico HTTP utilizar el ancho de banda no utilizado por el tráfico de audio. Lo que queremos es emplear el ancho de banda de la forma más eficiente posible, sin desperdiciarlo cuando pueda ser aprovechado para otras cosas. Estas consideraciones nos llevan a nuestro tercer principio:

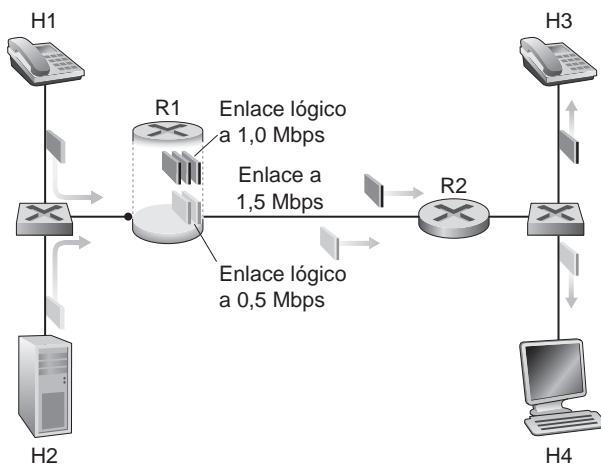


Figura 9.13 ♦ Aislamiento lógico de las clases de tráfico de audio y HTTP.

Principio 3: Mientras se proporciona aislamiento entre clases o flujos, es deseable utilizar los recursos (por ejemplo, el ancho de banda del enlace y los buffers) de la forma más eficiente posible.

Recuerde de las Secciones 1.3 y 4.2 que los paquetes pertenecientes a varios flujos de red se multiplexan y se ponen en cola para su transmisión en los buffers de salida asociados con un enlace. La forma en que los paquetes puestos en cola son seleccionados para su transmisión a través del enlace se conoce como **disciplina de planificación de enlace**, y ya hablamos de ella detalladamente en la Sección 4.2. Recuerde que en aquella sección explicamos tres disciplinas de planificación del enlace: FIFO, colas con prioridad y colas equitativas ponderadas (WFQ). Como pronto veremos, WFQ juega un papel particularmente importante a la hora de aislar las distintas clases de tráfico.

El mecanismo de goteo

Uno de los principios que establecimos anteriormente era que la vigilancia, es decir, la regulación de la velocidad a la que una clase o flujo puede injectar paquetes en la red (en la exposición que sigue supondremos que la unidad de vigilancia es un flujo), es un importante mecanismo de QoS. Pero, ¿qué aspectos de la tasa de paquetes de un flujo deberían ser vigilados? Podemos identificar tres importantes criterios de vigilancia, cada uno de ellos diferente con respecto a la escala de tiempo a lo largo de la cual se vigila el flujo de paquetes:

- *Tasa promedio.* La red puede querer limitar la tasa promedio a largo plazo (paquetes por intervalo de tiempo) a la que los paquetes de un flujo pueden ser enviados a la red. Un problema crucial en este caso es el intervalo de tiempo a lo largo del cual se vigilará la tasa promedio. Un flujo cuya tasa promedio esté limitada a 100 paquetes por segundo está más restringido que un origen que está limitado a 6.000 paquetes por minuto, incluso aunque ambos tengan la misma tasa promedio a lo largo de un intervalo de tiempo lo suficientemente largo. Por ejemplo, esta última restricción permitiría a un flujo enviar 1.000 paquetes en un determinado intervalo de un segundo, mientras que la primera restricción impediría este comportamiento de envío.
- *Tasa de pico.* Mientras que la restricción de la tasa promedio limita la cantidad de tráfico que puede ser enviada a la red para un periodo de tiempo relativamente largo, la restricción de la tasa de pico limita el número máximo de paquetes que pueden ser enviados en un periodo de tiempo más corto. Siguiendo con el ejemplo anterior, la red puede vigilar un flujo con una tasa promedio de 6.000 paquetes por minuto, a la vez que limita la tasa de pico del flujo a 1.500 paquetes por segundo.
- *Tamaño de la ráfaga.* La red también puede desear limitar el número máximo de paquetes (la “ráfaga” de paquetes) que pueden ser enviados a la red en un intervalo de tiempo extremadamente corto. En el límite, cuando la longitud del intervalo tiende a cero, el tamaño de la ráfaga limita el número de paquetes que pueden ser enviados a la red de forma instantánea. Incluso aunque físicamente sea imposible enviar instantáneamente varios paquetes a la red (después de todo, los enlaces tienen una velocidad de transmisión física que no puede excederse), la abstracción de un tamaño máximo de ráfaga es muy útil.

El mecanismo de goteo (también llamado de la cubeta con pérdidas) es una abstracción que puede utilizarse para caracterizar estos límites de vigilancia. Como se muestra en la Figura 9.14, una cubeta con pérdidas es una cubeta que puede almacenar hasta b fichas. Las fichas se van añadiendo a esta cubeta de la forma siguiente: las nuevas fichas que potencialmente pueden añadirse a la cubeta siempre se generan a una velocidad de r fichas por segundo. (Para simplificar, supondremos que la unidad de tiempo en este caso es el segundo.) Si la cubeta contiene menos de b fichas cuando se genera una ficha, la ficha que se acaba de generar se añade a la cubeta; en caso contrario, dicha ficha recién generada se ignora y la cubeta sigue estando llena, con b fichas.

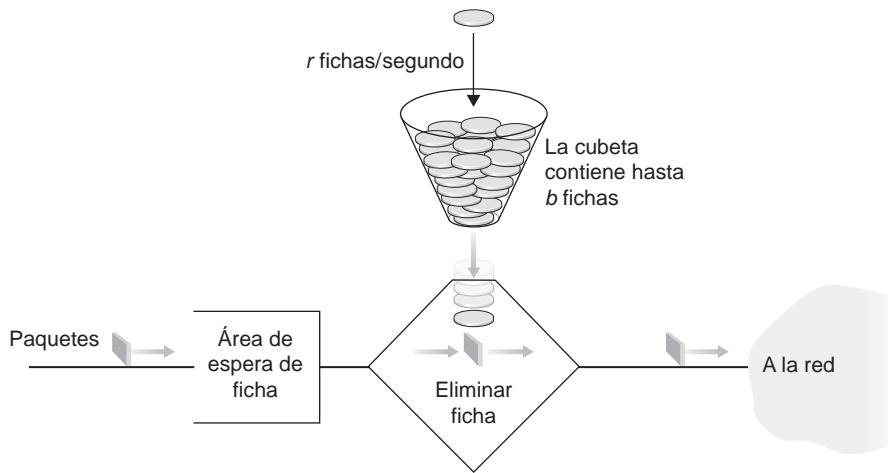


Figura 9.14 ♦ El controlador de la cubeta con pérdidas.

Veamos ahora cómo se puede utilizar una cubeta con pérdidas para vigilar un flujo de paquetes. Supongamos que antes de transmitir un paquete a la red es necesario eliminar una ficha de la cubeta. En este caso, si la cubeta está vacía, el paquete tendrá que esperar a que haya una ficha. (Una alternativa sería eliminar el paquete, aunque esta opción no vamos a considerarla aquí.) Consideremos ahora cómo este comportamiento permite vigilar un flujo de tráfico. Puesto que como máximo puede haber b fichas en la cubeta, el tamaño máximo de ráfaga para un flujo controlado mediante una cubeta con pérdidas es de b fichas. Además, dado que la velocidad de generación de fichas es r , el número máximo de paquetes que pueden entrar en la red en cualquier intervalo de tiempo de longitud t es $rt + b$. Por tanto, la tasa de generación de fichas r sirve para limitar la tasa promedio a largo plazo a la que los paquetes pueden acceder a la red. También pueden utilizarse varias cubetas con pérdidas (en concreto, dos cubetas de este tipo en serie) para controlar la tasa de pico de un flujo, además de la tasa promedio a largo plazo (consulte los problemas de repaso incluidos al final del capítulo).

Cubeta con pérdidas + cola WFQ = retardo máximo demostrable en una cola

Vamos a concluir nuestro análisis del mecanismo de vigilancia mostrando cómo combinar la cubeta con pérdidas y WFQ para proporcionar un límite al retardo de puesta en cola en un router. (Los lectores que hayan olvidado cómo funciona WFQ pueden repasar la Sección 4.2.) Consideremos el enlace de salida de un router que multiplexa n flujos, estando cada uno de ellos controlado mediante una cubeta con pérdidas cuyos parámetros son b_i y r_i , $i = 1, \dots, n$, y utilizando la disciplina de planificación WFQ. Vamos a emplear el término *flujo* aquí en un sentido laxo, para hacer referencia al conjunto de paquetes que el planificador no distingue entre sí. En la práctica, un flujo puede comprender tráfico procedente de una única conexión extremo a extremo o de una colección de conexiones de este tipo (véase la Figura 9.15).

Recuerde, de nuestras explicaciones sobre WFQ, que a cada flujo i se le garantiza que reciba una cuota del ancho de banda del enlace igual a, como mínimo, $R \cdot w_i / (\sum w_j)$, donde R es la velocidad de transmisión del enlace en paquetes/segundo. En este caso, ¿cuál es el retardo máximo que experimentará un paquete mientras espera para recibir servicio en la cola WFQ (es decir, después de atravesar la cubeta con pérdidas)? Vamos a centrarnos en el flujo 1. Suponga que la cubeta del flujo 1 está inicialmente llena y que entonces llega una ráfaga de b_1 paquetes al controlador de la cubeta del flujo 1. Estos paquetes extraerán todas las fichas de la cubeta (sin esperar) y pasarán al área de espera WFQ correspondiente al flujo 1. Puesto que estos b_1 paquetes reciben servicio a una tasa de

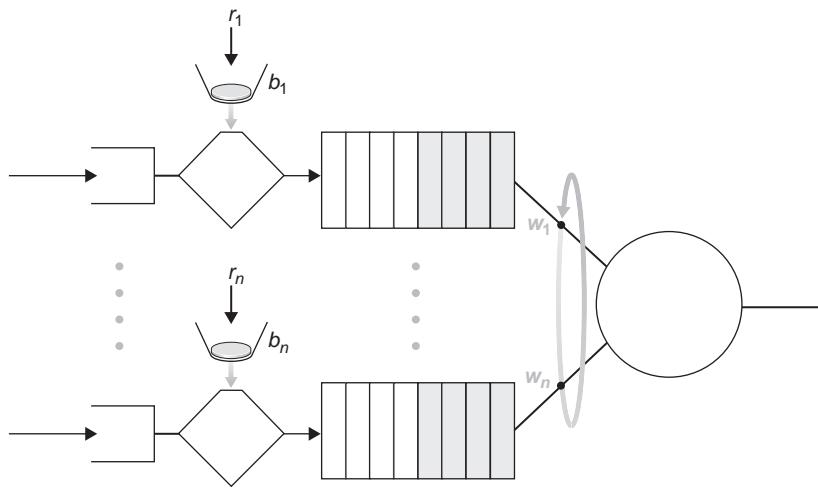


Figura 9.15 \diamond n flujos multiplexados con cubetas con pérdidas y planificación WFQ.

como mínimo $R \cdot w_i / (\sum w_j)$ paquetes/segundo, el último de estos paquetes sufrirá un retardo máximo d_{\max} hasta que se complete su transmisión, donde

$$d_{\max} = \frac{b_i}{R \cdot w_i / \sum w_j}$$

La fundamentación de esta fórmula es que si hay b_1 paquetes en la cola y se les está dando servicio (se les está eliminando de la cola) a una tasa de como mínimo $R \cdot w_1 / (\sum w_j)$ paquetes por segundo, entonces la cantidad de tiempo transcurrido hasta que se transmite el último bit del último paquete no puede ser mayor que $b_1 / (R \cdot w_1 / (\sum w_j))$. Uno de los problemas de repaso le pide que demuestre que, siempre y cuando $r_1 < R \cdot w_1 / (\sum w_j)$, entonces d_{\max} es el retardo máximo que cualquier paquete del flujo 1 experimentará en la cola WFQ.

9.5.3 Diffserv

Habiendo visto la motivación, los principios y los mecanismos específicos para proporcionar múltiples clases de servicio, concluyamos nuestro estudio de las soluciones existentes con un ejemplo: la arquitectura Diffserv de Internet [RFC 2475; Kilkki 1999]. Diffserv proporciona una diferenciación de servicio, es decir, la capacidad de manejar diferentes clases de tráfico de formas distintas dentro de Internet, de una manera escalable. La necesidad de escalabilidad surge del hecho de que en un router troncal de Internet pueden existir millones de flujos simultáneos de tráfico origen-destino. Veremos en breve que esta necesidad se cubre incluyendo solamente una simple funcionalidad dentro del núcleo de la red, implementándose las operaciones de control más complejas en la frontera de la red.

Comencemos con la sencilla red mostrada en la Figura 9.16. Aquí vamos a describir un posible uso de Diffserv; como se describe en el documento RFC 2475, son posibles otras variantes. La arquitectura Diffserv consta de dos conjuntos de elementos funcionales:

- *Funciones de frontera: clasificación de paquetes y acondicionamiento del tráfico.* En la frontera de entrada de la red (es decir, en cualquier host compatible con Diffserv que genere tráfico o en el primer router compatible con Diffserv a través del cual pase el tráfico), se marcan los paquetes que llegan. Más específicamente, se asigna un cierto valor al campo Servicio diferenciado (DS, *Differentiated Service*) de la cabecera del paquete IPv4 o IPv6 [RFC 3260]. La definición del campo DS pretende sustituir las definiciones anteriores del campo Tipo de servicio de IPv4 y de

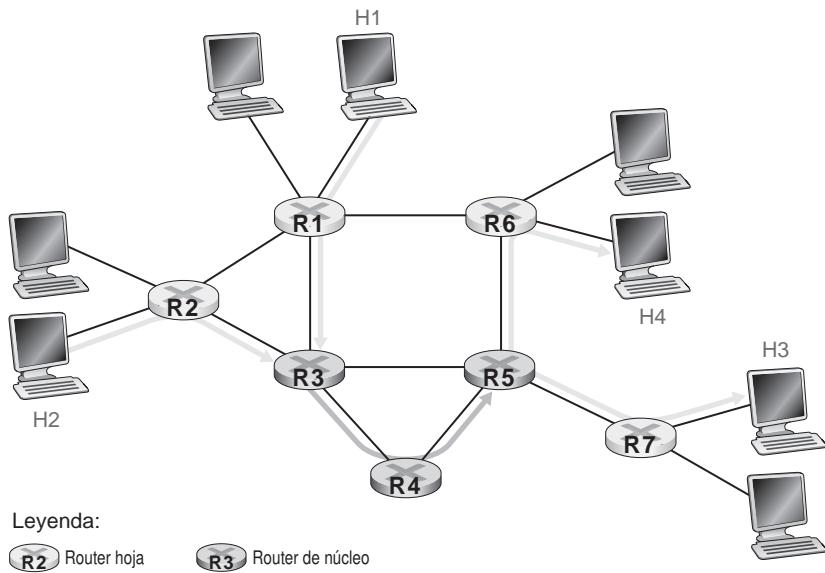


Figura 9.16 ♦ Un ejemplo de red Diffserv simple.

los campos de clase de tráfico de IPv6, de los que hemos hablado en el Capítulo 4. Por ejemplo, en la Figura 9.16, los paquetes que están siendo transmitidos de H1 a H3 pueden marcarse en R1, mientras que los paquetes que se envían de H2 a H4 pueden marcarse en R2. La marca que recibe un paquete identifica la clase de tráfico a la que pertenece. Las distintas clases de tráfico recibirán entonces un servicio diferente en el núcleo de la red.

- **Función del núcleo: reenvío.** Cuando un paquete marcado con DS llega a un router compatible con Diffserv, el paquete es reenviado al siguiente salto de acuerdo con el denominado comportamiento por salto (PHB, *Per-Hop Behavior*) asociado con dicha clase de paquete. El comportamiento por salto influye en cómo las clases de tráfico en competencia comparten los buffers de un router y el ancho de banda del enlace. Un principio fundamental de la arquitectura Diffserv es que el comportamiento por salto de un router se basará únicamente en las marcas de los paquetes, es decir, en la clase de tráfico a la que pertenece el paquete. Por tanto, si los paquetes que se están enviando de H1 a H3 en la Figura 9.16 reciben la misma marca que los paquetes que están siendo enviados de H2 a H4, entonces los routers de la red tratan estos paquetes como un agregado, sin distinguir si los paquetes fueron originados en H1 o en H2. Por ejemplo, R3 no diferenciaría entre los paquetes de H1 y H2 al reenviarlos hacia R4. De este modo, la arquitectura Diffserv elimina la necesidad de mantener el estado del router para cada pareja individual origen-destino (una consideración crítica para poder hacer Diffserv escalable).

En este momento puede resultarnos útil presentar una analogía. En muchos actos sociales a gran escala (por ejemplo, una importante recepción pública, una sala de baile o discoteca, un concierto o un partido de fútbol), las personas que asisten al acto reciben distintos tipos de entradas o pases: las personalidades reciben entradas VIP; los mayores de 18 años disponen de entradas de adulto (por ejemplo, en el caso de que se vayan a servir bebidas alcohólicas); los pases para estar entre bastidores en los conciertos; los pases de prensa para los periodistas, incluso un pase ordinario para las personas normales. Normalmente, estos diversos tipos de entradas o pases se distribuyen a la entrada del acto, es decir, en la frontera que da paso al acto. Es precisamente en la frontera donde se llevan a cabo las operaciones que requieren una computación intensiva, tales como abonar una entrada, comprobar que el tipo de invitación es el apropiado y comprobar que la invitación se corresponde con algún tipo de identificación personal. Además, puede existir un límite relativo al número de personas de un determinado tipo que pueden asistir al acto. Si existe tal límite, es posible que la gente tenga

que esperar antes de entrar en el recinto. Una vez que se ha accedido al acto, el pase permite que cada persona reciba un servicio diferenciado en las distintas zonas que forman el evento; por ejemplo, un VIP puede obtener bebidas gratis, una mesa mejor, comida gratis, acceso a salas exclusivas y un servicio de atenciones. Por el contrario, una persona normal queda excluida de determinadas áreas, tendrá que abonar las bebidas y sólo recibirá un servicio básico. En ambos casos, el servicio recibido dentro del recinto depende únicamente del tipo de pase del que disponga la persona. Además, todas las personas pertenecientes a una misma clase son tratadas de igual forma.

La Figura 9.17 proporciona una visión lógica de las funciones de clasificación y marcado en el router de frontera. En primer lugar, los paquetes que llegan al router de frontera se clasifican. El clasificador selecciona los paquetes basándose en los valores de uno o más campos de la cabecera del paquete (por ejemplo, dirección de origen, dirección de destino, puerto de origen, puerto de destino e ID de protocolo) y dirige al paquete a la función de marcado apropiada. Como ya hemos dicho, la marca de un paquete se transporta dentro del campo DS de la cabecera del paquete.

En algunos casos, un usuario final puede estar de acuerdo en limitar su velocidad de transmisión de paquetes con el fin de cumplir un **perfil de tráfico** previamente declarado. El perfil de tráfico puede contener un límite para la tasa de pico, así como para el tamaño de ráfaga del flujo de paquetes, como hemos visto anteriormente con el mecanismo de la cubeta con pérdidas. Siempre y cuando el usuario envíe paquetes a la red cumpliendo con el perfil de tráfico negociado, los paquetes reciben su marca de prioridad y son reenviados a lo largo de su ruta hasta alcanzar el destino. Por el contrario, si se viola el perfil de tráfico, los paquetes que no cumplen los límites impuestos por dicho perfil pueden marcarse de forma diferente, pueden conformarse (por ejemplo, ser retardados de manera que se cumpla una restricción de tasa máxima) o pueden ser eliminados en la frontera de la red. El papel de la **función de medida**, mostrada en la Figura 9.17, es comparar el flujo de paquetes entrantes con el perfil de tráfico negociado y determinar si un paquete cumple con dicho perfil. La decisión real acerca de si volver a marcar de forma inmediata, reenviar, retardar o eliminar un paquete es una cuestión de política que debe determinar el administrador de la red y *no* está especificada en la arquitectura Diffserv.

Hasta el momento, nos hemos centrado en las funciones de marcado y vigilancia de la arquitectura Diffserv. El segundo componente clave de dicha arquitectura tiene que ver con el comportamiento por salto (PHB, *Per-Hop Behavior*) de los routers compatibles con Diffserv. El PHB se define de manera críptica, pero precisa, como “una descripción del comportamiento de reenvío externamente observable de un nodo Diffserv, aplicado a un agregado concreto de comportamiento Diffserv” [RFC 2475]. Profundizando un poco en esta definición, podemos ver que contiene varias consideraciones importantes:

- Un PHB puede dar lugar a que diferentes clases de tráfico obtengan distinto rendimiento (es decir, diferentes comportamientos de reenvío externamente observables).

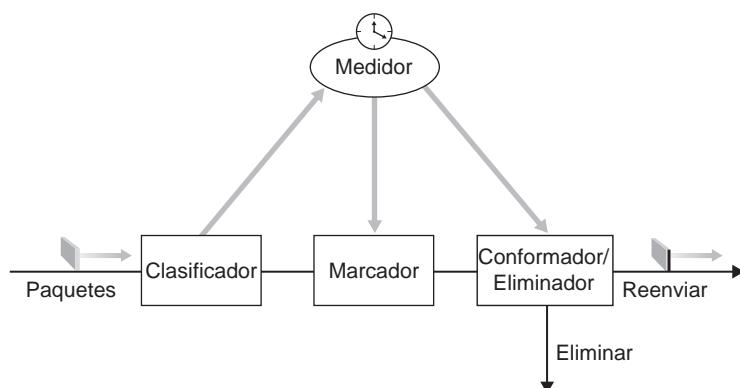


Figura 9.17 ♦ Visión lógica de las funciones de clasificación de paquetes y acondicionamiento de tráfico en el router terminal.

- Aunque un PHB define diferencias de rendimiento (comportamientos) entre clases, no impone ningún mecanismo concreto para conseguir estos comportamientos. Siempre y cuando se cumplan los criterios de rendimiento externamente observables, puede aplicarse cualquier mecanismo de implementación y puede usarse cualquier política de asignación de buffer/ancho de banda. Por ejemplo, un PHB no requiere que se emplee una disciplina de colas de paquetes concreta (por ejemplo, una cola con prioridad, una cola WFQ o una cola FCFS) para conseguir un determinado comportamiento. El PHB es el fin, mientras que la asignación de recursos y los mecanismos de implementación son los medios.
- Las diferencias de rendimiento deben ser observables y por tanto mensurables.

Actualmente están definidos dos PHB: un PHB de reenvío expedido (EF, *Expedited Forwarding*) [RFC 3246] y un PHB de reenvío garantizado (AF, *Assured Forwarding*) [RFC 2597]. El PHB de **reenvío expedido** (EF) especifica que la tasa de salida de un router para una clase de tráfico tiene que ser igual o mayor que una tasa configurada. El PHB de **reenvío garantizado** (AF) divide el tráfico en cuatro clases, y a cada una de las clases se le garantiza que recibirá cierta cantidad mínima de ancho de banda y de buffer.

Vamos a cerrar nuestra exposición sobre Diffserv con unas cuantas observaciones sobre su modelo de servicio. En primer lugar, hemos asumido implícitamente que Diffserv se implanta dentro de un único dominio administrativo, pero lo normal es que un servicio extremo a extremo tenga que ser establecido entre varios ISP que se encuentran entre los sistemas terminales que se están comunicando. Para proporcionar un servicio diferenciado extremo a extremo, todos los ISP existentes entre los sistemas terminales no sólo tienen que proporcionar dicho servicio, sino que también tienen que cooperar y establecer acuerdos para ofrecer al usuario final un verdadero servicio extremo a extremo. Sin esta clase de cooperación, los ISP que venden directamente el servicio diferenciado a los clientes tendrían que decir continuamente: “Sí, sabemos que usted ha pagado un extra, pero no tenemos un acuerdo de servicio con el ISP que ha perdido y retardado su tráfico. ¡Sentimos que se hayan producido muchos huecos en su llamada VoIP!”. En segundo lugar, si Diffserv estuviera realmente implementada y la red operara con una carga sólo moderada, la mayor parte del tiempo no se percibiría ninguna diferencia entre un servicio con entrega de mejor esfuerzo y un servicio Diffserv. De hecho, el retardo extremo a extremo normalmente está dominado por las velocidades de acceso y los saltos de router, en lugar de por los retardos de cola en los routers. ¡Imagine al infeliz cliente de Diffserv que ha pagado por un servicio premium, sólo para descubrir que el servicio de entrega de mejor esfuerzo que se está proporcionando a otros clientes ofrece casi el mismo rendimiento que el servicio premium!

9.5.4 Garantías de calidad de servicio (QoS) por conexión: reserva de recursos y admisión de llamadas

En la sección anterior hemos visto que el marcado y la vigilancia de paquetes, el aislamiento del tráfico y la planificación en el nivel de enlace pueden proporcionar una clase de servicio con mejor rendimiento que otra. Con ciertas disciplinas de planificación, como la planificación con prioridad, las clases de tráfico inferiores son prácticamente “invisibles” para la clase de tráfico con prioridad más alta. Con un dimensionamiento apropiado de la red, la clase de servicio de más alta prioridad puede, de hecho, conseguir tasas de pérdida de paquetes y retardos extremadamente bajos, con un rendimiento esencialmente idéntico al de las redes de conmutación de circuitos. ¿Pero puede la red *garantizar* que un flujo activo perteneciente a una clase de tráfico de alta prioridad continuará recibiendo dicho servicio mientras dure el flujo, utilizando únicamente los mecanismos que hemos descrito hasta el momento? En realidad no. En esta sección veremos por qué hacen falta otros mecanismos de red y protocolos adicionales cuando se proporciona una garantía de servicio estricta a ciertas conexiones individuales.

Volvamos a nuestro escenario de la Sección 9.5.2 y consideremos dos aplicaciones de audio a 1 Mbps que transmiten sus paquetes a través del enlace de 1,5 Mbps, como se muestra en la

Figura 9.18. La velocidad combinada de datos de los dos flujos (2 Mbps) excede la capacidad del enlace. Incluso utilizando los mecanismos de clasificación y marcado, el aislamiento de flujos y la compartición del ancho de banda no utilizado (que en este caso es igual a cero), es obvio que no podemos alcanzar nuestro objetivo. Simplemente, no existe el suficiente ancho de banda como para satisfacer las necesidades de ambas aplicaciones al mismo tiempo. Si las dos aplicaciones comparten equitativamente el ancho de banda, cada una de ellas perderá el 25% de sus paquetes transmitidos. Ésta es una calidad de servicio tan inaceptablemente baja que ambas aplicaciones de audio serán completamente inutilizables; de hecho, ni siquiera merece la pena transmitir ningún paquete de audio.

Dado que no se puede satisfacer simultáneamente a las dos aplicaciones de la Figura 9.18, ¿qué debería hacer la red? Permitir que ambas continúen con una QoS inaceptable implica desperdiciar los recursos de la red en una serie de flujos de aplicación que, en último extremo, no tienen ninguna utilidad para el usuario final. La respuesta es bastante simple: habrá que bloquear uno de los flujos de aplicación (es decir, denegarle el acceso a la red) mientras que se permite al otro continuar utilizando el 1 Mbps completo que la aplicación necesita. La red telefónica sería un ejemplo de red en la que se efectúa ese tipo de bloqueo de llamadas: si no se pueden asignar a la llamada los recursos requeridos (un circuito extremo a extremo, en el caso de la red telefónica), se bloquea la llamada (se la impide cursarse a través de la red) y se devuelve una señal de ocupado al usuario. En nuestro ejemplo, no se gana nada permitiendo que un flujo entre en la red si no va a recibir una QoS suficiente como para poder considerarlo utilizable. De hecho, existe un coste asociado a la admisión de un flujo que no vaya a recibir su QoS necesaria, ya que se estarán empleando recursos de la red para dar soporte a un flujo que no proporciona ninguna utilidad al usuario final.

Admitiendo o bloqueando explícitamente los flujos según sus requisitos de recursos y los de los flujos ya admitidos, la red puede garantizar que los flujos admitidos reciban la QoS solicitada. En la necesidad de proporcionar una QoS garantizada a un cierto flujo, está implícita la necesidad de que ese flujo declare sus requisitos de QoS. Este proceso de hacer que un flujo declare sus requisitos de QoS y que luego la red acepte el flujo (con la QoS requerida) o lo bloquee se denomina proceso de **admisión de llamada**. Éste es, por tanto, el cuarto de nuestros principios fundamentales (de los otros tres hemos hablado en la Sección 9.5.2) de los mecanismos necesarios para proporcionar calidad de servicio (QoS).

Principio 4: Si no siempre van a estar disponibles los recursos suficientes y es necesario garantizar la calidad de servicio, se necesita un proceso de admisión de llamadas en el que los flujos declaren sus requisitos de QoS y, o bien sean admitidos en la red (con la QoS requerida), o bien sean bloqueados (si la red no puede proporcionar la QoS requerida).

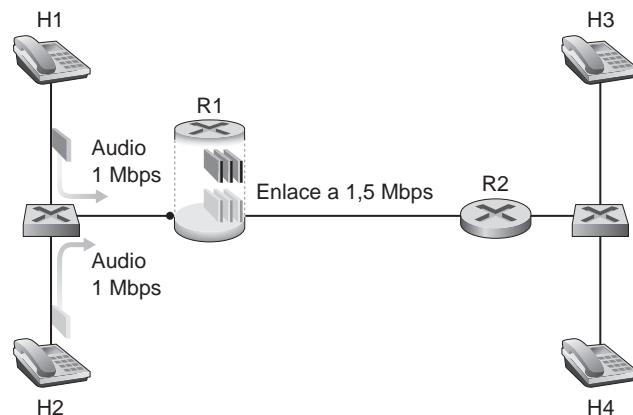


Figura 9.18 ♦ Dos aplicaciones de audio competidoras sobrecargan el enlace de R1 a R2.

Nuestro ejemplo introductorio de la Figura 9.18 resalta la necesidad de diversos nuevos mecanismos y protocolos de red, para el caso en que haya que garantizar a una llamada (a un flujo extremo) una determinada calidad de servicio una vez que la llamada se ha establecido:

- *Reserva de recursos.* La única forma de garantizar que una llamada dispondrá de los recursos (ancho de banda de enlace, buffers) necesarios para obtener su calidad de servicio deseada, consiste en asignar explícitamente dichos recursos a esa llamada; a este proceso se le conoce en la jerga del mundo de las redes con el nombre de **reserva de recursos**. Una vez reservados los recursos, la llamada podrá, mientras dure, acceder bajo demanda a dichos recursos, independientemente de las demandas de todas las restantes llamadas. Si una llamada reserva y recibe una garantía de x Mbps de ancho de banda de enlace y nunca transmite a una velocidad superior a x , podrá disfrutar de unas comunicaciones sin pérdidas y sin retardos.
- *Admisión de llamadas.* Si hay que reservar recursos, entonces la red tiene que disponer de un mecanismo para que las llamadas puedan solicitar y reservar los recursos. Dado que los recursos no son infinitos, cuando una llamada realiza una solicitud de admisión de llamada, esa admisión será denegada (es decir, la llamada será bloqueada) si no están disponibles los recursos solicitados. Este tipo de admisión de llamadas es realizado por las redes telefónicas, en las que solicitamos los recursos en el momento de marcar un número. Si están disponibles los circuitos (particiones TDMA) necesarios para completar la llamada, se asignarán los circuitos y la llamada podrá completarse. Si los circuitos no están disponibles, entonces se bloqueará la llamada y recibiremos la señal de ocupado. Una llamada bloqueada puede volver a intentar que la admitan en la red, pero no se le permitirá enviar tráfico hacia la red hasta que haya completado con éxito el proceso de admisión de llamada. Por supuesto, un router que realiza asignaciones del ancho de banda del enlace no debe asignar más ancho de banda del que esté disponible en dicho enlace. Normalmente una llamada sólo podrá reservar una fracción del ancho de banda del enlace, por lo que el router puede asignar ancho de banda del enlace a más de una llamada. Sin embargo, la suma del ancho de banda asignado a todas las llamadas debe ser inferior a la capacidad del enlace, si es que queremos ser capaces de proporcionar garantías estrictas de calidad de servicio.
- *Señalización del establecimiento de llamada.* El proceso de admisión de llamadas descrito más arriba requiere que las llamadas sean capaces de reservar los recursos suficientes en cada uno de los routers de la red que formen parte de la ruta entre el origen y el destino, con el fin de asegurarse de satisfacer sus requisitos de QoS extremo a extremo. Cada router deberá determinar los recursos locales requeridos por la sesión, tener en cuenta la cantidad de recursos que ya han sido comprometidos con otras sesiones activas y determinar si dispone de los suficientes recursos como para satisfacer los requisitos de QoS por salto que esa sesión tiene en dicho router, sin violar las garantías de QoS locales que ya se hayan concedido a otras sesiones ya admitidas. Hace falta un protocolo de señalización para coordinar estas diversas actividades: la asignación de recursos locales en cada salto, así como la decisión global extremo a extremo de si la llamada ha sido o no capaz de reservar los recursos suficientes en cada uno de los routers de la ruta entre los dos sistemas terminales. Este es el trabajo del **protocolo de establecimiento de llamada**, como se muestra en la Figura 9.19. Con este objetivo se propuso el **protocolo RSVP** [Zhang 1993, RFC 2210] dentro de una arquitectura Internet para proporcionar garantías de calidad de servicio. En las redes ATM, el protocolo Q2931b [Black 1995] transporta esta información entre los conmutadores de la red ATM y el punto terminal.

A pesar de los tremendos esfuerzos de investigación y desarrollo, y a pesar de que existen productos que proporcionan garantías de calidad de servicio por conexión, no ha habido apenas implantación de ese tipo de servicios. Son muchas las posibles razones. En primer lugar, es posible que los mecanismos simples de nivel de aplicación que hemos estudiado en las secciones 9.2 a 9.4, combinados con un dimensionamiento adecuado de la red (Sección 9.5.1), proporcionen un servicio de red con entrega de mejor esfuerzo “suficientemente bueno” para las aplicaciones multimedia. Además, la complejidad añadida y el coste de implantar y gestionar una red que proporcione

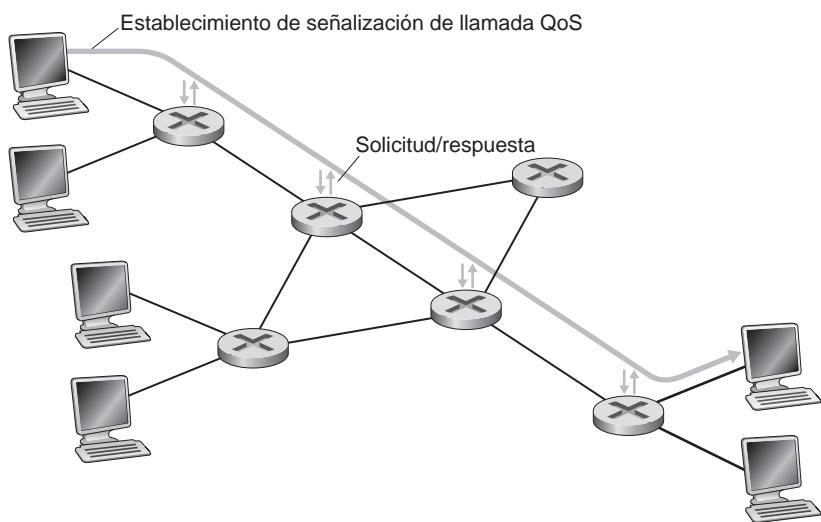


Figura 9.19 ♦ Proceso de establecimiento de llamada.

garantías de calidad de servicio por conexión pueden ser considerados por los ISP simplemente como demasiado altos, dadas las previsiones de ingresos para ese tipo de servicio.

9.6 Resumen

El campo de las redes multimedia constituye uno de los desarrollos más atractivos en la red Internet actual. Personas de todo el mundo pasan cada vez menos tiempo delante de su televisión y utilizan en su lugar sus teléfonos inteligentes y otros dispositivos para recibir transmisiones de audio y de vídeo, tanto en vivo como pregrabadas. Y con sitios como YouTube, los usuarios se han convertido en productores, además de consumidores de contenido Internet multimedia. Además de para la distribución de vídeo, Internet también se está empleando para transportar llamadas telefónicas. De hecho, a lo largo de la próxima década, Internet (junto con el acceso inalámbrico a Internet) puede llegar a hacer que el sistema telefónico tradicional de conmutación de circuitos quede obsoleto. VoIP no solo proporciona servicios telefónicos baratos, sino que también proporciona numerosos servicios de valor añadido, como videoconferencia, servicios de directorio en línea, mensajería de voz e integración en redes sociales como Facebook y WeChat.

En la Sección 9.1 hemos descrito las características intrínsecas del vídeo y la voz, y luego hemos clasificado las aplicaciones multimedia en tres categorías diferentes: (i) flujos de audio/vídeo almacenado, (ii) aplicaciones de conversación voz/vídeo sobre IP y (iii) flujos de audio/vídeo en vivo.

En la Sección 9.2 hemos estudiado con un cierto grado de detalles la transmisión de flujos de vídeo almacenado. Para las aplicaciones de flujos de vídeo, los vídeos pregrabados se almacenan en servidores y los usuarios envían solicitudes a esos servidores con el fin de ver los vídeos a la carta. Allí dijimos que los sistemas de flujos de vídeo pueden clasificarse en dos categorías: flujos UDP y HTTP. Vimos que la medida de rendimiento más importante en el caso de los flujos de vídeo es la tasa media de transferencia.

En la Sección 9.3 hemos examinado cómo pueden diseñarse aplicaciones de conversación multimedia, como VoIP, para ejecutarse sobre una red con servicio de entrega de mejor esfuerzo. Para las conversaciones multimedia, las consideraciones de temporización son importantes, porque las aplicaciones de conversación son extremadamente sensibles al retardo. Por otro lado, las aplicaciones de conversación multimedia son tolerantes a las pérdidas: las pérdidas ocasionales solo

causan cortes ocasionales en la reproducción del audio/vídeo, y estas pérdidas pueden a menudo ocultarse total o parcialmente. Vimos cómo una combinación de buffers de cliente, números de secuencia de los paquetes y marcas de tiempo, pueden aliviar enormemente los efectos de las fluctuaciones inducidas por la red. También repasamos la tecnología en la que se basa Skype, una de las empresas punteras en voz y vídeo sobre IP. En la Sección 9.4, examinamos dos de los más importantes protocolos estandarizados para VoIP, concretamente RTP y SIP.

En la Sección 9.5 hemos visto cómo pueden utilizarse diversos mecanismos de red (disciplinas de planificación de nivel de enlace y mecanismos de vigilancia del tráfico) para proporcionar un servicio diferenciado a distintas clases de tráfico.

Problemas y cuestiones de repaso

Capítulo 9 Cuestiones de repaso

SECCIÓN 9.1

- R1. Reconstruya la Tabla 9.1 para el caso de que Víctor este viendo un vídeo a 4 Mbps, Francisco esté examinando una nueva imagen de 100 Kbytes cada 20 segundos y Marta esté escuchando un flujo de audio a 200 kbps.
- R2. Hay dos tipos de redundancia en el vídeo. Describalos y explique cómo pueden aprovecharse para conseguir una compresión eficiente.
- R3. Suponga que muestreamos una señal analógica de audio 16.000 veces por segundo y que cada muestra se cuantiza en uno de 1024 niveles. ¿Cuál será la tasa de bits resultante de la señal PCM de audio digital?
- R4. Las aplicaciones multimedia se pueden clasificar en tres categorías. Enumere y describa cada una de ellas.

SECCIÓN 9.2

- R5. Los sistemas de flujos de vídeo pueden clasificarse en tres categorías. Enumérelas y describa brevemente cada una de ellas.
- R6. Indique tres desventajas de los flujos UDP.
- R7. Con los flujos HTTP, ¿son la misma cosa el buffer de recepción TCP y el buffer de la aplicación cliente? Si la respuesta es negativa, ¿cómo interaccionan?
- R8. Considere el modelo simple de flujos HTTP. Suponga que el servidor envía bits a una tasa constante de 2 Mbps y que la reproducción comienza cuando se han recibido 8 millones de bits. ¿Cuál es el retardo inicial de almacenamiento en buffer t_p ?

SECCIÓN 9.3

- R9. ¿Cuál es la diferencia entre el retardo extremo a extremo y la fluctuación de paquetes? ¿Cuáles son las causas de la fluctuación de paquetes?
- R10. ¿Por qué un paquete que se recibe después de su instante de reproducción planificado se considera un paquete perdido?
- R11. En la Sección 9.3 se han descrito dos esquemas FEC. Resúmalos brevemente. Ambos esquemas incrementan la velocidad de transmisión del flujo mediante la adición de más sobrecarga. ¿El intercalado aumenta también la velocidad de transmisión?

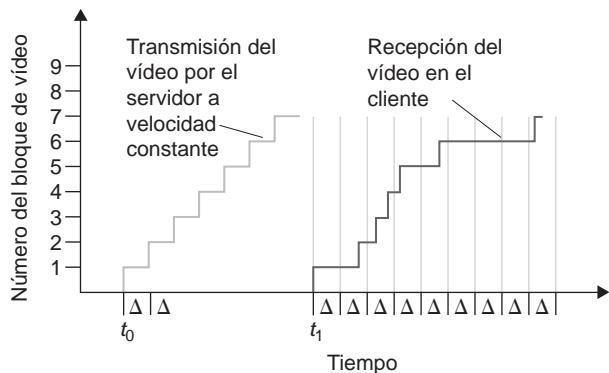
SECCIÓN 9.4

- R12. ¿Cómo identifica un receptor los distintos flujos RTP de sesiones diferentes? ¿Cómo se identifican los diferentes flujos dentro de la misma sesión?

- R13. ¿Cuál es el papel de un registrador SIP? ¿En qué se diferencia el papel de un registrador SIP del de un agente propio en IP móvil?

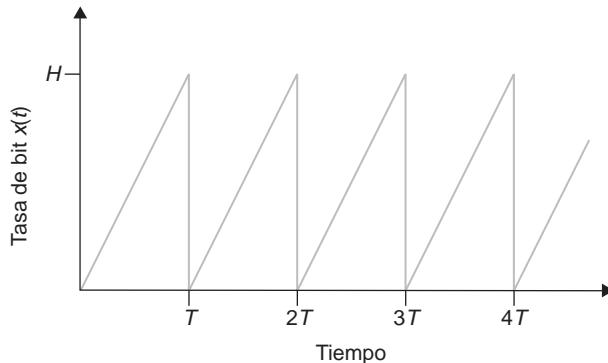
Problemas

- P1. Considere la siguiente figura. De forma similar a nuestro análisis de la Figura 9.1, suponga que el vídeo está codificado con una tasa de bits constante y que, por tanto, cada bloque de vídeo contiene fotogramas que hay que reproducir a lo largo de una misma cantidad fija de tiempo, Δ . El servidor transmite el primer bloque de vídeo en t_0 , el segundo bloque en $t_0 + \Delta$, el tercer bloque en $t_0 + 2\Delta$, etc. Una vez que el cliente comienza la reproducción, cada bloque debe reproducirse Δ unidades de tiempo después del bloque precedente.



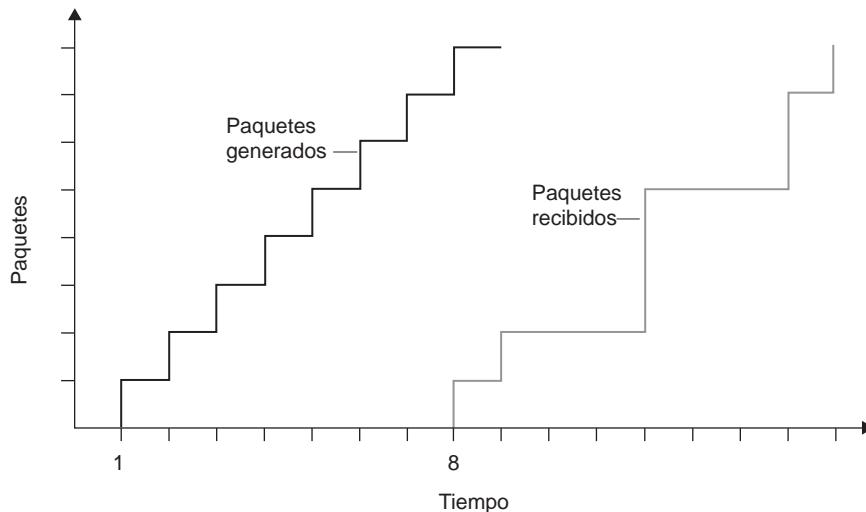
- Suponga que el cliente comienza la reproducción en cuanto llega el primer bloque en t_1 . Consulte la figura y determine cuántos bloques de vídeo (incluyendo el primero) llegarán al cliente a tiempo para ser reproducidos. Explique cómo ha llegado a esa respuesta.
 - Suponga que el cliente comienza ahora la reproducción en $t_1 + \Delta$. ¿Cuántos bloques de vídeo (incluyendo el primero) llegarán al cliente a tiempo para ser reproducidos? Explique cómo ha llegado a esa respuesta.
 - En el mismo escenario del apartado (b) anterior, ¿cuál es el máximo número de bloques que llegarán a estar almacenados en el buffer del cliente, a la espera de ser reproducidos? Explique cómo ha llegado a esa respuesta.
 - ¿Cuál será el mínimo retardo de reproducción en el cliente que garantice que todos los bloques de vídeo lleguen a tiempo para ser reproducidos? Explique cómo ha llegado a esa respuesta.
- P2. Recuerde el modelo simple de flujo HTTP mostrado en la Figura 9.3. Recuerde que B designa el tamaño del buffer de la aplicación cliente, y que Q designa el número de bits que hay que almacenar en el buffer antes de que la aplicación cliente comience la reproducción. Asimismo, r designa la velocidad de consumo del vídeo. Suponga que el servidor transmite los bits a una velocidad constante x mientras el buffer del cliente no esté lleno.
- Suponga que $x < r$. Como se explica en el texto, en este caso la reproducción alternará entre períodos de reproducción continua y períodos de congelación de la imagen. Determine la duración de cada período de reproducción continua y de congelación de la imagen, en función de Q , r y x .
 - Ahora suponga que $x > r$. ¿En qué momento $t = t_f$ se llenará el buffer de la aplicación cliente?
- P3. Recuerde el modelo simple de flujo HTTP mostrado en la Figura 9.3. Suponga que el tamaño del buffer es infinito, pero que el servidor transmite los bits con una velocidad variable $x(t)$.

Específicamente, suponga que $x(t)$ tiene la siguiente forma, en dientes de sierra. La velocidad es inicialmente cero en el instante $t = 0$ y aumenta linealmente hasta valer H en el instante $t = T$. Después se repite una y otra vez este patrón, como se muestra en la siguiente figura.



- ¿Cuál es la velocidad media de transmisión del servidor?
 - Suponga que $Q = 0$, de modo que el cliente comienza la reproducción en cuanto recibe el primer fotograma. ¿Qué sucederá?
 - Ahora suponga que $Q > 0$ y $HT/2 \geq Q$. Determine el instante en que comenzará la reproducción, en función de Q, H y T .
 - Suponga que $H > 2r$ y $Q = HT/2$. Demuestre que no se producirá ninguna congelación de la imagen después del retardo inicial de reproducción.
 - Suponga que $H > 2r$. Determine el valor mínimo de Q que garanticé que no se producirá ninguna congelación de la imagen después del retardo inicial de reproducción.
 - Ahora suponga que el tamaño de buffer B es finito y que $H > 2r$. Determine el instante $t = t_f$ en el que el buffer de la aplicación cliente se llena por primera vez, en función de Q, B, T y H .
- P4. Recuerde el modelo simple de flujo HTTP mostrado en la Figura 9.3. Suponga que el buffer de la aplicación cliente es infinito, que el servidor transmite a la velocidad constante x y que la velocidad de consumo del vídeo es r , con $r < x$. Suponga también que la reproducción comienza inmediatamente. Suponga que el usuario termina el vídeo anticipadamente, en el instante $t = E$. En el momento de cancelarse la reproducción, el servidor deja de enviar bits (si no ha enviado ya todos los bits contenidos en el vídeo).
- Suponga que el vídeo es infinitamente largo. ¿Cuántos bits se desperdiciarán (es decir, cuántos se habrán transmitido, pero no habrán llegado a reproducirse)?
 - Suponga que el vídeo tiene T segundos de duración, con $T > E$. ¿Cuántos bits se desperdiciarán (es decir, cuántos se transmitirán, pero no llegarán a reproducirse)?
- P5. Considere un sistema DASH (como el explicado en la Sección 2.6) para el que hay N versiones de vídeo (con N diferentes tasas y calidades) y N versiones de audio (con N diferentes tasas y calidades). Suponga que queremos permitir que el reproductor seleccione, en cualquier momento, cualquiera de las N versiones de vídeo y cualquiera de las N versiones de audio.
- Si creamos archivos en los que el audio esté mezclado con el vídeo, de modo que el servidor sólo envíe un flujo multimedia en cualquier momento dado, ¿cuántos archivos necesitará almacenar el servidor (cada uno con un URL distinto)?
 - Si, en lugar de ello, el servidor envía los flujos de audio y de vídeo por separado y deja que el cliente sincronice los flujos, ¿cuántos archivos necesitará almacenar el servidor?
- P6. En el ejemplo sobre VoIP de la Sección 9.3, sea h el número total de bytes de cabecera añadidos a cada fragmento, incluyendo las cabeceras UDP e IP.

- a. Suponiendo que se transmite un datagrama IP cada 20 milisegundos, calcule la velocidad de transmisión en bits por segundo para los datagramas generados por uno de los lados de esta aplicación.
- b. ¿Cuál es el valor típico de h cuando se utiliza RTP?
- P7. Considere el procedimiento descrito en la Sección 9.3 para estimar el retardo promedio d_r . Suponga que $u = 0,1$. Sea $r_1 - t_1$ el retardo de la muestra más reciente, sea $r_2 - t_2$ el retardo de la siguiente muestra más reciente, etc.
- Para una aplicación de audio dada, suponga que han llegado cuatro paquetes al receptor con los retardos de muestra $r_4 - t_4$, $r_3 - t_3$, $r_2 - t_2$ y $r_1 - t_1$. Exprese la estimación del retardo d en función de las cuatro muestras.
 - Generalice la fórmula para n retardos de muestra.
 - Para la fórmula del apartado (b), obtenga la fórmula resultante cuando n tiende a infinito. Comente por qué este procedimiento para obtener la media se denomina media móvil exponencial.
- P8. Repita los apartados (a) y (b) del problema anterior para obtener la estimación de la desviación media del retardo.
- P9. En el ejemplo sobre VoIP de la Sección 9.3, hemos presentado un procedimiento en línea (media móvil exponencial) para estimar el retardo. En este problema vamos a examinar un procedimiento alternativo. Sea t_i la marca de tiempo del paquete i -ésimo recibido; sea r_i el instante en el que el paquete i -ésimo es recibido. Sea d_n nuestra estimación del retardo medio después de recibir el paquete n -ésimo. Después de recibir el primer paquete, establecemos que la estimación del retardo es igual a $d_1 = r_1 - t_1$.
- Suponga que deseamos que $d_n = (r_1 - t_1 + r_2 - t_2 + \dots + r_n - t_n)/n$ para todo n . Proporcione una fórmula recursiva para d_n en función de d_{n-1} , r_n y t_n .
 - Describa por qué en la telefonía por Internet la estimación del retardo descrita en la Sección 9.3 es más apropiada que la estimación dada en el apartado (a).
- P10. Compare el procedimiento descrito en la Sección 9.3 para estimar el retardo medio con el procedimiento dado en la Sección 3.5 para estimar el tiempo de ida y vuelta. ¿Qué tienen en común ambos procedimientos? ¿En qué se diferencian?
- P11. Considere la siguiente figura (que es similar a la Figura 9.3). Un emisor comienza a enviar audio empaquetado periódicamente en $t = 1$. El primer paquete llega al receptor en $t = 8$.



- a. ¿Cuáles son los retardos (del emisor al receptor, ignorando cualquier retardo de reproducción) de los paquetes 2 a 8? Observe que cada segmento de línea vertical y horizontal de la figura tiene una longitud de 1, 2 o 3 unidades de tiempo.
- b. Si la reproducción del audio se inicia tan pronto como llega el primer paquete al receptor en $t = 8$, ¿cuáles de los ocho primeros paquetes enviados no llegarán a tiempo para la reproducción?
- c. Si la reproducción del audio comienza en $t = 9$, ¿cuáles de los ocho primeros paquetes enviados no llegarán a tiempo para la reproducción?
- d. ¿Cuál es el retardo mínimo de reproducción en el receptor que hace que los ocho primeros paquetes lleguen a tiempo para la reproducción?
- P12. Considere de nuevo la figura del Problema P11, que muestra los tiempos de transmisión y de recepción de los paquetes.
- Calcule el retardo estimado para los paquetes 2 a 8 utilizando la fórmula para d_i de la Sección 9.3.2. Utilice un valor de $u = 0,1$.
 - Calcule la desviación estimada del retardo respecto del promedio estimado para los paquetes 2 a 8, utilizando la fórmula para v_i de la Sección 9.3.2. Utilice un valor de $u = 0,1$.
- P13. Recuerde los dos esquemas FEC para VoIP descritos en la Sección 9.3. Suponga que el primer esquema genera un fragmento redundante por cada cuatro fragmentos originales. Suponga que el segundo esquema utiliza una codificación con una baja tasa de bit cuya velocidad de transmisión es el 25 por ciento de la velocidad de transmisión del flujo nominal.
- ¿Cuánto ancho de banda adicional requiere cada esquema? ¿Cuánto retardo de reproducción añade cada esquema?
 - ¿Cómo funciona cada uno de los dos esquemas cuando se pierde el primer paquete de cada grupo de cinco paquetes? ¿Qué esquema proporcionará una mejor calidad de audio?
 - ¿Cómo funciona cada uno de los dos esquemas cuando se pierde el primer paquete de cada grupo de dos paquetes? ¿Qué esquema proporcionará una mejor calidad de audio?
- P14. a. Considere una llamada de audioconferencia en Skype con $N > 2$ participantes. Suponga que cada participante genera un flujo constante con una tasa igual a r bps. ¿Cuántos bits por segundo necesitará enviar el iniciador de la llamada? ¿Cuántos bits por segundo necesitará enviar cada uno de los otros $N - 1$ participantes? ¿Cuál es la tasa de envío total, agregando la de todos los participantes?
- b. Repita el apartado (a) para una videoconferencia Skype que utilice un servidor central.
- c. Repita el apartado (b), pero ahora para el caso de que cada par envíe una copia de su flujo de vídeo a cada uno de los otros $N - 1$ pares.
- P15. a. Suponga que enviamos a Internet dos datagramas IP, transportando cada uno de ellos un segmento UDP diferente. El primer datagrama tiene una dirección IP de origen A1, una dirección IP de destino B, un puerto de origen P1 y un puerto de destino T. El segundo datagrama tiene la dirección IP de origen A2, la dirección IP de destino B, el puerto de origen P2 y el puerto de destino T. Suponga que A1 es diferente de A2 y que P1 es diferente de P2. Suponiendo que ambos datagramas llegan a su destino final, ¿serán recibidos los dos datagramas UDP por el mismo socket? ¿Por qué?
- b. Suponga que Alicia, Benito y Clara desean mantener una audioconferencia utilizando SIP y RTP. Para que Alicia pueda intercambiar paquetes RTP con Benito y Clara, ¿basta con un socket UDP (además del socket necesario para los mensajes SIP)? En caso afirmativo, ¿cómo distingue el cliente SIP de Alicia los paquetes RTP recibidos de Benito de los procedentes de Clara?

P16. Verdadero o falso:

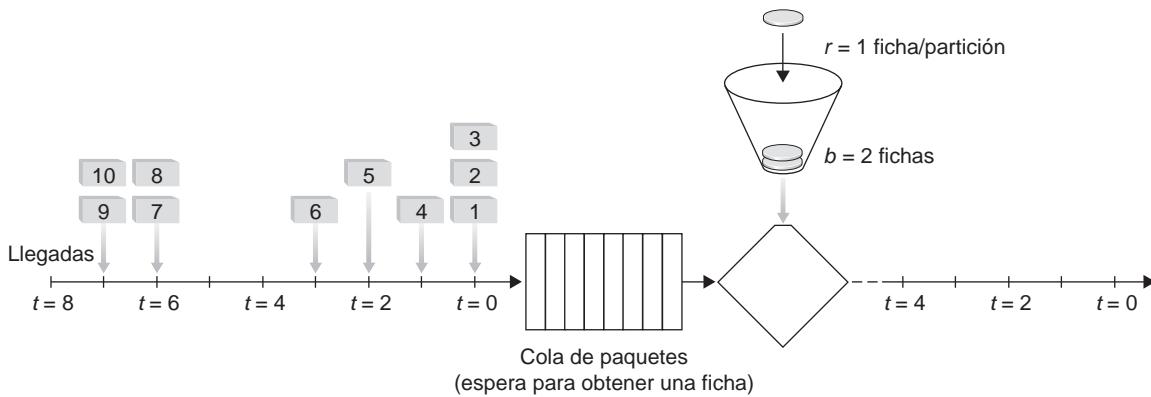
- Si un flujo de vídeo almacenado se descarga directamente de un servidor web a un reproductor multimedia, entonces la aplicación está usando TCP como protocolo de transporte subyacente.
- Cuando se está usando RTP, un emisor puede cambiar la codificación en mitad de una sesión.
- Todas las aplicaciones que usan RTP deben emplear el puerto 87.
- Si una sesión RTP tiene un flujo separado de audio y de vídeo para cada emisor, entonces los flujos de audio y de vídeo usan el mismo SSRC.
- En los servicios diferenciados, aunque el comportamiento por salto define diferencias de rendimiento entre las clases, no impone ningún mecanismo concreto para conseguir ese rendimiento.
- Suponga que Alicia desea establecer una sesión SIP con Benito. En su mensaje INVITE, Alicia incluye la línea: m=audio 48753 RTP/AVP 3 (AVP 3 indica audio GSM). Alicia ha indicado, por tanto, en este mensaje que desea enviar audio GSM.
- Respecto a la afirmación anterior, Alicia ha indicado en su mensaje INVITE que enviará el audio al puerto 48753.
- Normalmente, los mensajes SIP se envían entre entidades SIP utilizando un número de puerto SIP predeterminado.
- Para mantener el registro, los clientes SIP tienen que enviar periódicamente mensajes REGISTER.
- SIP obliga a que todos los clientes SIP soporten la codificación de audio G.711.

P17. Considere la figura de la página siguiente, la cual muestra una cubeta con pérdidas que está siendo alimentada con un flujo de paquetes. El buffer de fichas puede almacenar como máximo dos fichas e, inicialmente, en $t = 0$ está lleno. Las nuevas fichas llegan con una tasa igual a una ficha por partición de tiempo. La velocidad del enlace de salida es tal que si dos paquetes obtienen fichas al principio de una partición de tiempo, ambos pueden dirigirse al enlace de salida en la misma partición de tiempo. Los detalles de temporización del sistema son los siguientes:

- Los paquetes (si los hay) llegan al principio de la partición de tiempo. Así, en la figura, los paquetes 1, 2 y 3 llegan en la partición 0. Si ya existen paquetes en la cola, entonces los paquetes que llegan se colocan al final de la cola. Los paquetes avanzan hacia el principio de la cola siguiendo una planificación FIFO.
- Una vez que se han añadido a la cola las fichas que llegan, si existe algún paquete en la cola, uno o dos de esos paquetes (dependiendo del número de fichas disponibles) eliminarán cada uno de ellos una ficha del buffer de fichas y pasarán al enlace de salida durante dicha partición. Por tanto, los paquetes 1 y 2 eliminan cada uno una ficha del buffer (puesto que inicialmente existen dos fichas) y pasan al enlace de salida durante la partición de tiempo 0.
- Si el buffer de fichas no está lleno, se añade una nueva ficha, ya que la velocidad de generación de fichas es $r = 1$ ficha/partición.
- El tiempo avanza entonces a la siguiente partición de tiempo y estos pasos se repiten.

Responda a las siguientes preguntas:

- Para cada partición de tiempo, identifique los paquetes que se encuentran en la cola y el número de fichas que hay en la cubeta inmediatamente después de que las llegadas hayan sido procesadas (paso 1 anterior), pero antes de que cualquiera de los paquetes haya atravesado la cola y eliminado una ficha. Así, para la partición $t = 0$ del ejemplo anterior, los paquetes 1, 2 y 3 están en la cola y en el buffer hay dos fichas.



- b. Para cada partición de tiempo, indique qué paquetes aparecen en la salida después de que se hayan eliminado de la cola la o las fichas. Así, para la partición $t = 0$ del ejemplo anterior, los paquetes 1 y 2 aparecen en el enlace de salida del buffer con pérdidas durante la partición 0.
- P18. Repita el Problema P17, pero suponiendo ahora que $r = 2$. Suponga de nuevo que la cubeta está inicialmente llena.
- P19. Considere el Problema P18 y suponga en este caso que $r = 3$ y que $b = 2$, al igual que antes. ¿Cambiaría esto su respuesta al problema anterior?
- P20. Considere el mecanismo de vigilancia de la cubeta con pérdidas que monitoriza la velocidad promedio y el tamaño de ráfaga de un flujo de paquetes. Suponga ahora que también deseamos monitorizar la velocidad de pico, p . Explique cómo la salida de este mecanismo de vigilancia de la cubeta con pérdidas puede alimentar a un segundo mecanismo del mismo tipo, de modo que las dos cubetas con pérdidas conectadas en serie monitoricen la velocidad media, la velocidad de pico y el tamaño de ráfaga. Asegúrese de determinar el tamaño de la cubeta y la velocidad de generación de fichas del segundo mecanismo de vigilancia.
- P21. Se dice que un flujo de paquetes cumple una especificación de cubeta con pérdidas (r, b) con un tamaño de ráfaga igual a b y una velocidad media r si el número de paquetes que llega a dicha cubeta es menor que $rt + b$ paquetes en todo intervalo de tiempo de duración t , para todo t . Un flujo de paquetes que cumpla con una especificación de cubeta con pérdidas (r, b) , ¿tendrá que esperar alguna vez en un controlador de cubeta con pérdidas con parámetros r y b ? Justifique su respuesta.
- P22. Demuestre que siempre y cuando $r_1 < R w_1 / (\sum w_i)$, entonces d_{\max} es el retardo máximo que cualquier paquete del flujo 1 puede experimentar en la cola WFQ.

Tarea de programación

En esta práctica de laboratorio tendrá que implementar un servidor y un cliente de flujos de vídeo. El cliente utilizará el protocolo RTSP (*Real-Time Streaming Protocol*) para controlar las acciones del servidor. El servidor utilizará el protocolo de tiempo real RTP para empaquetar el vídeo y transportarlo sobre UDP. Se le proporcionará código Python que implementa parcialmente RTSP y RTP en el cliente y en el servidor. Su trabajo consistirá en completar tanto el código del cliente como el del servidor. Cuando haya terminado, habrá creado una aplicación cliente-servidor que hará lo siguiente:

- El cliente envía comandos SETUP, PLAY, PAUSE y TEARDOWN RTSP, y el servidor responde a los comandos.

- Cuando el servidor se encuentra en el estado de reproducción, periódicamente captura una trama JPEG almacenada, la empaqueta con RTP y envía el paquete RTP a un socket UDP.
- El cliente recibe los paquetes RTP, extrae las tramas JPEG, descomprime las tramas y las reproduce en el monitor del cliente.

El código que le proporcionamos implementa el protocolo RTSP en el servidor y el desempaquetamiento RTP en el cliente. El código también se ocupa de mostrar el vídeo transmitido. Tendrá que implementar RTSP en el cliente y el servidor RTP. Esta tarea de programación mejorará significativamente la compresión del estudiante sobre RTP, RTSP y la transmisión de los flujos de vídeo, por lo que le recomendamos que la realice. La tarea también sugiere una serie de ejercicios opcionales, incluyendo la implementación del comando RTSP DESCRIBE tanto en el cliente como en el servidor. Puede encontrar todos los detalles acerca de la tarea, así como una panorámica general del protocolo RTSP en el sitio web www.pearsonhighered.com/cs-resources.

UNA ENTREVISTA CON...

Henning Schulzrinne

Henning Schulzrinne es profesor jefe del Departamento de Ciencias de la Computación y director del Internet Real-Time Laboratory en la Universidad de Columbia. Es coautor de RTP, RTSP, SIP y GIST, protocolos clave para las comunicaciones de audio y vídeo a través de Internet. Henning obtuvo su título de grado en Ingeniería Eléctrica e Industrial en la Universidad Técnica de Darmstadt, Alemania, su máster en Ingeniería Eléctrica y de Computadoras en la Universidad de Cincinnati y es doctor en Ingeniería Eléctrica por la Universidad de Massachusetts, Amherst.



¿Qué le hizo especializarse en redes multimedia?

Ocurrió casi por casualidad. Como estudiante de doctorado, estuve involucrado en DARTnet, una red experimental que cubría los Estados Unidos mediante líneas T1. DARTnet se utilizó como campo de pruebas para la multidifusión y las herramientas en tiempo real de Internet. Esto me llevó a escribir mi primera herramienta para audio, NeVoT. A través de algunos de los participantes de DARTnet, me involucré en el IETF, dentro del entonces naciente grupo de trabajo de Transporte de Audio y Video. Este grupo terminó más tarde estandarizando RTP.

¿Cuál fue su primer trabajo en la industria de las computadoras? ¿Qué implicó?

Mi primer trabajo en la industria de las computadoras fue el de soldar un kit de computadora Altair cuando era estudiante de bachillerato en Livermore, California. Cuando volví a Alemania, comencé con una empresa pequeña de consultoría que diseñó un programa de gestión de direcciones para una agencia de viajes, almacenando los datos en cintas de casete para nuestro TRS-80 y utilizando como impresora una máquina de escribir IBM Selectric con una interfaz hardware casera.

Mi primer trabajo de verdad fue en los laboratorios AT&T Bell Laboratories, donde desarrollé un emulador de red para construir redes experimentales en un entorno de laboratorio.

¿Cuáles son los objetivos del Internet Real-Time Lab?

Nuestro objetivo es proporcionar componentes y piezas para la red Internet, como única infraestructura de comunicaciones en el futuro. Esto incluye el desarrollo de nuevos protocolos, como GIST (para la señalización de la capa de red) y LoST (para localizar recursos por su ubicación), o la mejora de protocolos con los que hemos trabajado anteriormente, como SIP, con investigaciones en sistemas de enriquecimiento de presencia, P2P, llamadas de emergencia de nueva generación y herramientas de creación de servicios. Recientemente, nos hemos centrado especialmente también en los sistemas inalámbricos para VoIP, ya que es probable que las redes 802.11b y 802.11n, y también posiblemente WiMax, cobren importancia como tecnologías de conexión de abonado en el campo de la telefonía. También estamos intentando mejorar de forma considerable la capacidad de los usuarios para diagnosticar los fallos en la complicada maraña de proveedores y equipos, utilizando un sistema de diagnóstico de fallos entre pares conocido como DYSWIS (*Do You See What I See, ves lo que yo veo*).

Intentamos llevar a cabo trabajos relevantes desde el punto de vista práctico, construyendo prototipos y sistemas de código abierto, midiendo el rendimiento de sistemas reales y contribuyendo a los estándares del IETF.

¿Cuál es su visión del futuro de las redes multimedia?

Ahora nos encontramos en una fase de transición; estamos sólo a unos pocos años de que IP sea la plataforma universal para los servicios multimedia, desde IPTV a VoIP. Todos confiamos en que la radio, el teléfono y la

televisión estén disponibles incluso durante las tormentas de nieve y los terremotos; por tanto, cuando Internet asuma el papel de estas redes dedicadas, los usuarios esperarán el mismo nivel de fiabilidad.

Tendremos que aprender a diseñar tecnologías de red para un ecosistema de operadoras, servicios y proveedores de contenido competidores, que darán servicio a muchos usuarios sin formación técnica, a los que tendrán que defender de un conjunto pequeño, pero destructivo, de usuarios maliciosos y criminales. Modificar los protocolos está comenzando a ser cada más complicado. Además, cada vez son más complejos, ya que tienen que tener en cuenta los intereses de negocios competidores, los temas de seguridad y confidencialidad y la falta de transparencia de las redes debida al uso de cortafuegos y traductores de direcciones de red.

Dado que las redes multimedia se están convirtiendo en la base de casi todas las aplicaciones de entretenimiento de gran consumo, tendrá que hacerse hincapié en la gestión de redes de muy gran tamaño, con un coste bajo. Los usuarios esperarán disponer de una gran facilidad de uso, para poder localizar, por ejemplo, el mismo contenido en todos sus dispositivos.

¿Por qué tiene SIP un futuro prometedor?

A medida que se actualicen las redes inalámbricas actuales convirtiéndose en redes 3G, es de esperar que un único mecanismo de señalización multimedia se extienda a todos los tipos de redes, desde los modems por cable a las redes telefónicas corporativas y las redes inalámbricas públicas. Junto con las radios software, esto hará posible que en el futuro pueda utilizarse un único dispositivo en una red doméstica, como por ejemplo un teléfono inalámbrico BlueTooth, en una red corporativa a través de 802.11 y en una red de área extensa mediante 3G. Incluso antes de que dispongamos de tal dispositivo inalámbrico universal y único, los mecanismos de movilidad personal permiten ocultar las diferencias entre redes. Un identificador se convierte en el medio universal de localización de una persona, en lugar de tener que recordar o comunicar media docena de números de teléfono específicos de cada tecnología o de cada ubicación.

SIP también rompe el monopolio de la provisión de transporte de voz (bits), que hasta ahora tenían los servicios de voz. Ahora es técnicamente posible romper el monopolio de la telefonía local, de modo que una empresa proporcione un transporte de bits neutral, mientras que otros proporcionan “tono de marcado” IP y los servicios telefónicos clásicos, como las pasarelas, el reenvío de llamadas y el identificador del llamante.

Más allá de la señalización multimedia, SIP ofrece un nuevo servicio que no existía en Internet: la notificación de sucesos. Se han hecho aproximaciones a tales servicios con técnicas HTTP y el correo electrónico, pero nunca han sido muy satisfactorias. Dado que los sucesos son una abstracción común en los sistemas distribuidos, esto puede simplificar la construcción de nuevos servicios.

¿Tiene algún consejo para los estudiantes que ahora se inician en el campo de las redes?

El campo de las redes abarca múltiples disciplinas. Aprovecha los resultados de la ingeniería eléctrica, las ciencias de la computación, la investigación operativa, la estadística, la economía y otras disciplinas. Por tanto, los investigadores de redes tienen que estar familiarizados con temas que van bastante más allá de los protocolos y los algoritmos de enrutamiento. Dado que las redes se están convirtiendo en una parte tan importante de la vida cotidiana, los estudiantes que deseen sobresalir en este campo deben pensar en las nuevas restricciones de recursos que afectarán a las redes: el tiempo y el esfuerzo de las personas, en lugar de pensar sólo en el ancho de banda o en la capacidad de almacenamiento.

Trabajar en el campo de investigación de las redes puede resultar tremadamente satisfactorio, ya que es algo que permite a las personas comunicarse e intercambiar ideas, una de las esencias de los seres humanos. Internet se ha convertido en la tercera infraestructura global más importante, después de los sistemas de transporte y de distribución de energía. Prácticamente ninguna parte de la economía puede funcionar sin redes de altas prestaciones, por lo que el futuro próximo está repleto de oportunidades.

Referencias

Una nota sobre las direcciones URL. En las referencias proporcionada a continuación, hemos incluido el URL de páginas web, documentos web y otros materiales que no se han publicado en una revista o conferencia (cuando hemos sido capaces de localizar un URL para ese material). No hemos incluido el URL de las publicaciones en conferencias o revistas, ya que esos documentos suelen poder ser localizados mediante un buscador en el sitio web de la propia conferencia (por ejemplo, los artículos de todas las conferencias y reuniones *ACM SIGCOMM* pueden localizarse a través de <http://www.acm.org/sigcomm>), o mediante una suscripción a una biblioteca digital. Aunque todos los URL incluidos a continuación eran válidos (y fueron comprobados) en enero de 2016, los URL pueden desactualizarse. En la versión en línea de este libro (www.pearsonhighered.com/cs-resources) podrá encontrar una bibliografía actualizada.

Una nota sobre los documentos RFC: Hay disponibles copias de los RFC de Internet en muchos sitios web. El Editor de documentos RFC de Internet Society (el organismo que se encarga de los RFC) mantiene el sitio web <http://www.rfc-editor.org>. Este sitio permite buscar un RFC específico por su título, su número o sus autores, y muestra las actualizaciones de todos los RFC enumerados. Los RFC de Internet pueden ser actualizados o convertidos en obsoletos por otros documentos RFC posteriores. Nuestro sitio web favorito para obtener documentos RFC es la fuente original: <http://www.rfc-editor.org>.

[3GPP 2016] Página web del Third Generation Partnership Project, <http://www.3gpp.org/>

[Abramson 1970] N. Abramson, “The Aloha System—Another Alternative for Computer Communications”, *Proc. 1970 Fall Joint Computer Conference, AFIPS Conference*, p. 37, 1970.

[Abramson 1985] N. Abramson, “Development of the Alohanet”, *IEEE Transactions on Information Theory*, Vol. IT-31, Nº 3 (marzo 1985), págs. 119–123.

[Abramson 2009] N. Abramson, “The Alohanet—Surfing for Wireless Data”, *IEEE Communications Magazine*, Vol. 47, Nº 12, págs. 21–25.

[Adhikari 2011a] V. K. Adhikari, S. Jain, Y. Chen, Z. L. Zhang, “Vivisecting YouTube: An Active Measurement Study”, Technical Report, University of Minnesota, 2011.

[Adhikari 2012] V. K. Adhikari, Y. Gao, F. Hao, M. Varvello, V. Hilt, M. Steiner, Z. L. Zhang, “Unreeling Netflix: Understanding and Improving Multi-CDN Movie Delivery”, Technical Report, University of Minnesota, 2012.

[Afanasyev 2010] A. Afanasyev, N. Tilley, P. Reiher, L. Kleinrock, “Host-to-Host Congestion Control for TCP”, *IEEE Communications Surveys & Tutorials*, Vol. 12, Nº 3, págs. 304–342.

[Agarwal 2009] S. Agarwal, J. Lorch, “Matchmaking for Online Games and Other Latency-sensitive P2P Systems”, *Proc. 2009 ACM SIGCOMM*.

[Ager 2012] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, W. Willinger, “Anatomy of a Large European ISP”, *Sigcomm*, 2012.

[Ahn 1995] J. S. Ahn, P. B. Danzig, Z. Liu, and Y. Yan, “Experience with TCP Vegas: Emulation and Experiment”, *Proc. 1995 ACM SIGCOMM* (Boston, MA, agosto 1995), págs. 185–195.

[Akamai 2016] Akamai homepage, <http://www.akamai.com>

[Akella 2003] A. Akella, S. Seshan, A. Shaikh, “An Empirical Evaluation of Wide-Area Internet Bottlenecks”, *Proc. 2003 ACM Internet Measurement Conference* (Miami, FL, noviembre 2003).

- [Akhshabi 2011]** S. Akhshabi, A. C. Begen, C. Dovrolis, “An Experimental Evaluation of Rate-Adaptation Algorithms in Adaptive Streaming over HTTP”, *Proc. 2011 ACM Multimedia Systems Conf.*
- [Akyildiz 2010]** I. Akyildiz, D. Gutierrez-Estevez, E. Reyes, “The Evolution to 4G Cellular Systems, LTE Advanced”, *Physical Communication*, Elsevier, 3 (2010), 217–244.
- [Albitz 1993]** P. Albitz and C. Liu, *DNS and BIND*, O'Reilly & Associates, Petaluma, CA, 1993.
- [Al-Fares 2008]** M. Al-Fares, A. Loukissas, A. Vahdat, “A Scalable, Commodity Data Center Network Architecture”, *Proc. 2008 ACM SIGCOMM*.
- [Amazon 2014]** J. Hamilton, “AWS: Innovation at Scale”, YouTube video, https://www.youtube.com/watch?v=JIQETrFC_SQ
- [Anderson 1995]** J. B. Andersen, T. S. Rappaport, S. Yoshida, “Propagation Measurements and Models for Wireless Communications Channels”, *IEEE Communications Magazine*, (enero 1995), págs. 42–49.
- [Alizadeh 2010]** M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan. “Data center TCP (DCTCP)”, *ACM SIGCOMM 2010 Conference*, ACM, Nueva York, NY, USA, págs. 63–74.
- [Allman 2011]** E. Allman, “The Robustness Principle Reconsidered: Seeking a Middle Ground”, *Communications of the ACM*, Vol. 54, Nº 8 (agosto 2011), págs. 40–45.
- [Appenzeller 2004]** G. Appenzeller, I. Keslassy, N. McKeown, “Sizing Router Buffers”, *Proc. 2004 ACM SIGCOMM* (Portland, OR, agosto 2004).
- [ASO-ICANN 2016]** The Address Supporting Organization homepage, <http://www.aso.icann.org>
- [AT&T 2013]** “AT&T Vision Alignment Challenge Technology Survey”, AT&T Domain 2.0 Vision White Paper, 13 de noviembre de 2013.
- [Atheros 2016]** Atheros Communications Inc., “Atheros AR5006 WLAN Chipset Product Bulletins”, <http://www.atheros.com/pt/AR5006Bulletins.htm>
- [Ayanoglu 1995]** E. Ayanoglu, S. Paul, T. F. La Porta, K. K. Sabnani, R. D. Gitlin, “AIRMAIL: A Link-Layer Protocol for Wireless Networks”, *ACM ACM/Baltzer Wireless Networks Journal*, 1: 47–60, febrero 1995.
- [Bakre 1995]** A. Bakre, B. R. Badrinath, “I-TCP: Indirect TCP for Mobile Hosts”, *Proc. 1995 Int. Conf. on Distributed Computing Systems (ICDCS)* (mayo 1995), págs. 136–143.
- [Balakrishnan 1997]** H. Balakrishnan, V. Padmanabhan, S. Seshan, R. Katz, “A Comparison of Mechanisms for Improving TCP Performance Over Wireless Links”, *IEEE/ACM Transactions on Networking* Vol. 5, Nº 6 (diciembre 1997).
- [Balakrishnan 2003]** H. Balakrishnan, F. Kaashoek, D. Karger, R. Morris, I. Stoica, “Looking Up Data in P2P Systems”, *Communications of the ACM*, Vol. 46, Nº 2 (febrero 2003), págs. 43–48.
- [Baldauf 2007]** M. Baldauf, S. Dustdar, F. Rosenberg, “A Survey on Context-Aware Systems”, *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 2, Nº 4 (2007), págs. 263–277.
- [Baran 1964]** P. Baran, “On Distributed Communication Networks”, *IEEE Transactions on Communication Systems*, marzo 1964. Rand Corporation Technical report with the same title (Memorandum RM-3420-PR, 1964). <http://www.rand.org/publications/RM/RM3420/>
- [Bardwell 2004]** J. Bardwell, “You Believe You Understand What You Think I Said . . . The Truth About 802.11 Signal and Noise Metrics: A Discussion Clarifying Often-Misused 802.11 WLAN Terminologies”, http://www.connect802.com/download/techpubs/2004/you_believe_D100201.pdf

- [Barford 2009]** P. Barford, N. Duffield, A. Ron, J. Sommers, “Network Performance Anomaly Detection and Localization”, *Proc. 2009 IEEE INFOCOM* (abril 2009).
- [Baronti 2007]** P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, Y. Hu, “Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards”, *Computer Communications*, Vol. 30, Nº 7 (2007), págs. 1655–1695.
- [Baset 2006]** S. A. Basset and H. Schulzrinne, “An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol”, *Proc. 2006 IEEE INFOCOM* (Barcelona, España, abril 2006).
- [BBC 2001]** BBC news online “A Small Slice of Design”, abril 2001, <http://news.bbc.co.uk/2/hi/science/nature/1264205.stm>
- [Beheshti 2008]** N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, G. Salmon, “Experimental Study of Router Buffer Sizing”, *Proc. ACM Internet Measurement Conference* (octubre 2008, Vouliagmeni, Greece).
- [Bender 2000]** P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, A. Viterbi, “CDMA/HDR: A Bandwidth-Efficient High-Speed Wireless Data Service for Nomadic Users”, *IEEE Commun. Mag.*, Vol. 38, Nº 7 (julio 2000), págs. 70–77.
- [Berners-Lee 1989]** T. Berners-Lee, CERN, “Information Management: A Proposal”, marzo 1989, mayo 1990. <http://www.w3.org/History/1989/proposal.html>
- [Berners-Lee 1994]** T. Berners-Lee, R. Cailliau, A. Luotonen, H. Frystyk Nielsen, A. Secret, “The World-Wide Web”, *Communications of the ACM*, Vol. 37, Nº 8 (agosto 1994), págs. 76–82.
- [Bertsekas 1991]** D. Bertsekas, R. Gallagher, *Data Networks, 2nd Ed.*, Prentice Hall, Englewood Cliffs, NJ, 1991.
- [Biersack 1992]** E. W. Biersack, “Performance Evaluation of Forward Error Correction in ATM Networks”, *Proc. 1999 ACM SIGCOMM* (Baltimore, MD, agosto 1992), págs. 248–257.
- [BIND 2016]** Internet Software Consortium page on BIND, <http://www.isc.org/bind.html>
- [Bisdikian 2001]** C. Bisdikian, “An Overview of the Bluetooth Wireless Technology”, *IEEE Communications Magazine*, Nº 12 (diciembre 2001), págs. 86–94.
- [Bishop 2003]** M. Bishop, *Computer Security: Art and Science*, Boston: Addison Wesley, Boston MA, 2003.
- [Black 1995]** U. Black, *ATM Volume I: Foundation for Broadband Networks*, Prentice Hall, 1995.
- [Black 1997]** U. Black, *ATM Volume II: Signaling in Broadband Networks*, Prentice Hall, 1997.
- [Blumenthal 2001]** M. Blumenthal, D. Clark, “Rethinking the Design of the Internet: The End-to-end Arguments vs. the Brave New World”, *ACM Transactions on Internet Technology*, Vol. 1, Nº 1 (agosto 2001), págs. 70–109.
- [Bochman 1984]** G. V. Bochmann, C. A. Sunshine, “Formal Methods in Communication Protocol Design”, *IEEE Transactions on Communications*, Vol. 28, Nº 4 (abril 1980) págs. 624–631.
- [Bolot 1996]** J-C. Bolot, A. Vega-Garcia, “Control Mechanisms for Packet Audio in the Internet”, *Proc. 1996 IEEE INFOCOM*, págs. 232–239.
- [Bosshart 2013]** P. Bosshart, G. Gibb, H. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, M. Horowitz, “Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN”, *ACM SIGCOMM Comput. Commun. Rev.* 43, 4 (agosto 2013), 99–110.
- [Bosshart 2014]** P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, D. Walker, “P4: Programming Protocol-Independent Packet Processors”, *ACM SIGCOMM Comput. Commun. Rev.* 44, 3 (julio 2014), págs. 87–95.

- [Brakmo 1995]** L. Brakmo, L. Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet”, *IEEE Journal of Selected Areas in Communications*, Vol. 13, Nº 8 (octubre 1995), págs. 1465–1480.
- [Bryant 1988]** B. Bryant, “Designing an Authentication System: A Dialogue in Four Scenes”, <http://web.mit.edu/kerberos/www/dialogue.html>
- [Bush 1945]** V. Bush, “As We May Think”, *The Atlantic Monthly*, julio 1945. <http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm>
- [Byers 1998]** J. Byers, M. Luby, M. Mitzenmacher, A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data”, *Proc. 1998 ACM SIGCOMM* (Vancouver, Canadá, agosto 1998), págs. 56–67.
- [Caesar 2005a]** M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, J. van der Merwe, “Design and implementation of a Routing Control Platform”, *Proc. Networked Systems Design and Implementation* (mayo 2005).
- [Caesar 2005b]** M. Caesar, J. Rexford, “BGP Routing Policies in ISP Networks”, *IEEE Network Magazine*, Vol. 19, Nº 6 (noviembre 2005).
- [Caldwell 2012]** C. Caldwell, “The Prime Pages”, <http://www.utm.edu/research/primes/prove>
- [Cardwell 2000]** N. Cardwell, S. Savage, T. Anderson, “Modeling TCP Latency”, *Proc. 2000 IEEE INFOCOM* (Tel-Aviv, Israel, marzo 2000).
- [Casado 2007]** M. Casado, M. Freedman, J. Pettit, J. Luo, N. McKeown, S. Shenker, “Ethane: Taking Control of the Enterprise”, *Proc. ACM SIGCOMM '07*, Nueva York, págs. 1–12. Véase también *IEEE/ACM Trans. Networking*, 17, 4 (agosto 2007), págs. 270–1283.
- [Casado 2009]** M. Casado, M. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, S. Shenker, “Rethinking Enterprise Network Control”, *IEEE/ACM Transactions on Networking (ToN)*, Vol. 17, Nº 4 (agosto 2009), págs. 1270–1283.
- [Casado 2014]** M. Casado, N. Foster, A. Guha, “Abstractions for Software-Defined Networks”, *Communications of the ACM*, Vol. 57 Nº 10, (octubre 2014), págs. 86–95.
- [Cerf 1974]** V. Cerf, R. Kahn, “A Protocol for Packet Network Interconnection”, *IEEE Transactions on Communications Technology*, Vol. COM-22, Nº 5, págs. 627–641.
- [CERT 2001–09]** CERT, “Advisory 2001–09: Statistical Weaknesses in TCP/IP Initial Sequence Numbers”, <http://www.cert.org/advisories/CA-2001-09.html>
- [CERT 2003–04]** CERT, “CERT Advisory CA-2003-04 MS-SQL Server Worm”, <http://www.cert.org/advisories/CA-2003-04.html>
- [CERT 2016]** CERT, <http://www.cert.org>
- [CERT Filtering 2012]** CERT, “Packet Filtering for Firewall Systems”, http://www.cert.org/tech_tips/packet_filtering.html
- [Cert SYN 1996]** CERT, “Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks”, <http://www.cert.org/advisories/CA-1998-01.html>
- [Chandra 2007]** T. Chandra, R. Greisemer, J. Redstone, “Paxos Made Live: an Engineering Perspective”, Proc. of 2007 ACM Symposium on Principles of Distributed Computing (PODC), págs. 398–407.
- [Chao 2001]** H. J. Chao, C. Lam, E. Oki, *Broadband Packet Switching Technologies—A Practical Guide to ATM Switches and IP Routers*, John Wiley & Sons, 2001.

[Chao 2011] C. Zhang, P. Dunghel, D. Wu, K. W. Ross, “Unraveling the BitTorrent Ecosystem”, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 22, Nº 7 (julio 2011).

[Chen 2000] G. Chen, D. Kotz, “A Survey of Context-Aware Mobile Computing Research”, *Technical Report TR2000-381*, Dept. of Computer Science, Dartmouth College, noviembre 2000. <http://www.cs.dartmouth.edu/reports/TR2000-381.pdf>

[Chen 2006] K.-T. Chen, C.-Y. Huang, P. Huang, C.-L. Lei, “Quantifying Skype User Satisfaction”, *Proc. 2006 ACM SIGCOMM* (Pisa, Italia, septiembre 2006).

[Chen 2011] Y. Chen, S. Jain, V. K. Adhikari, Z. Zhang, “Characterizing Roles of Front-End Servers in End-to-End Performance of Dynamic Content Distribution”, *Proc. 2011 ACM Internet Measurement Conference* (Berlín, Alemania, nov. 2011).

[Cheswick 2000] B. Cheswick, H. Burch, S. Branigan, “Mapping and Visualizing the Internet”, *Proc. 2000 Usenix Conference* (San Diego, CA, junio 2000).

[Chiu 1989] D. Chiu, R. Jain, “Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks”, *Computer Networks and ISDN Systems*, Vol. 17, Nº 1, págs. 1–14. http://www.cs.wustl.edu/~jain/papers/cong_av.htm

[Christiansen 2001] M. Christiansen, K. Jeffay, D. Ott, F. D. Smith, “Tuning Red for Web Traffic”, *IEEE/ACM Transactions on Networking*, Vol. 9, Nº 3 (junio 2001), págs. 249–264.

[Chuang 2005] S. Chuang, S. Iyer, N. McKeown, “Practical Algorithms for Performance Guarantees in Buffered Crossbars”, *Proc. 2005 IEEE INFOCOM*.

[Cisco 802.11ac 2014] Cisco Systems, “802.11ac: The Fifth Generation of Wi-Fi”, Technical White Paper, marzo 2014.

[Cisco 7600 2016] Cisco Systems, “Cisco 7600 Series Solution and Design Guide”, http://www.cisco.com/en/US/products/hw/routers/ps368/prod_technical_reference09186a0080092246.html

[Cisco 8500 2012] Cisco Systems Inc., “Catalyst 8500 Campus Switch Router Architecture”, http://www.cisco.com/univercd/cc/td/doc/product/l3sw/8540/rel_12_0/w5_6f/softcnfg/1cfg8500.pdf

[Cisco 12000 2016] Cisco Systems Inc., “Cisco XR 12000 Series and Cisco 12000 Series Routers”, <http://www.cisco.com/en/US/products/ps6342/index.html>

[Cisco 2012] Cisco 2012, Data Centers, <http://www.cisco.com/go/dce>

[Cisco 2015] Cisco Visual Networking Index: Forecast and Methodology, 2014–2019, White Paper, 2015.

[Cisco 6500 2016] Cisco Systems, “Cisco Catalyst 6500 Architecture White Paper”, http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/prod_white_paper0900aecd80673385.html

[Cisco NAT 2016] Cisco Systems Inc., “How NAT Works”, http://www.cisco.com/en/US/tech/tk648/tk361/technologies_tech_note09186a0080094831.shtml

[Cisco QoS 2016] Cisco Systems Inc., “Advanced QoS Services for the Intelligent Internet”, http://www.cisco.com/warp/public/cc/pd/iosw/isoft/ioqo/tech/qos_wp.htm

[Cisco Queue 2016] Cisco Systems Inc., “Congestion Management Overview”, http://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfconmg.html

[Cisco SYN 2016] Cisco Systems Inc., “Defining Strategies to Protect Against TCP SYN Denial of Service Attacks”, http://www.cisco.com/en/US/tech/tk828/technologies_tech_note09186a00800f67d5.shtml

- [Cisco TCAM 2014]** Cisco Systems Inc., “CAT 6500 and 7600 Series Routers and Switches TCAM Allocation Adjustment Procedures”, <http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/117712-problemsolution-cat6500-00.html>
- [Cisco VNI 2015]** Cisco Systems Inc., “Visual Networking Index”, http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html
- [Clark 1988]** D. Clark, “The Design Philosophy of the DARPA Internet Protocols”, *Proc. 1988 ACM SIGCOMM* (Stanford, CA, agosto 1988).
- [Cohen 1977]** D. Cohen, “Issues in Transnet Packetized Voice Communication”, *Proc. Fifth Data Communications Symposium* (Snowbird, UT, septiembre 1977), págs. 6–13.
- [Cookie Central 2016]** Cookie Central homepage, http://www.cookiecentral.com/n_cookie_faq.htm
- [Cormen 2001]** T. H. Cormen, *Introduction to Algorithms*, 2nd Ed., MIT Press, Cambridge, MA, 2001.
- [Crow 1997]** B. Crow, I. Widjaja, J. Kim, P. Sakai, “IEEE 802.11 Wireless Local Area Networks”, *IEEE Communications Magazine* (septiembre 1997), págs. 116–126.
- [Cusumano 1998]** M. A. Cusumano, D. B. Yoffie, *Competing on Internet Time: Lessons from Netscape and Its Battle with Microsoft*, Free Press, Nueva York, NY, 1998.
- [Czyz 2014]** J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, M. Bailey, “Measuring IPv6 Adoption”, *Proc. ACM SIGCOMM 2014*, ACM, Nueva York, NY, USA, págs. 87–98.
- [Dahlman 1998]** E. Dahlman, B. Gudmundson, M. Nilsson, J. Sköld, “UMTS/IMT-2000 Based on Wideband CDMA”, *IEEE Communications Magazine* (septiembre 1998), págs. 70–80.
- [Daigle 1991]** J. N. Daigle, *Queueing Theory for Telecommunications*, Addison-Wesley, Reading, MA, 1991.
- [DAM 2016]** Digital Attack Map, <http://www.digitalattackmap.com>
- [Davie 2000]** B. Davie and Y. Rekhter, *MPLS: Technology and Applications*, Morgan Kaufmann Series in Networking, 2000.
- [Davies 2005]** G. Davies, F. Kelly, “Network Dimensioning, Service Costing, and Pricing in a Packet-Switched Environment”, *Telecommunications Policy*, Vol. 28, Nº 4, págs. 391–412.
- [DEC 1990]** Digital Equipment Corporation, “In Memoriam: J. C. R. Licklider 1915–1990”, SRC Research Report 61, agosto 1990. <http://www.memex.org/licklider.pdf>
- [DeClercq 2002]** J. DeClercq, O. Paridaens, “Scalability Implications of Virtual Private Networks”, *IEEE Communications Magazine*, Vol. 40, Nº 5 (mayo 2002), págs. 151–157.
- [Demers 1990]** A. Demers, S. Keshav, S. Shenker, “Analysis and Simulation of a Fair Queueing Algorithm”, *Internetworking: Research and Experience*, Vol. 1, Nº 1 (1990), págs. 3–26.
- [dhc 2016]** IETF Dynamic Host Configuration working group homepage, <http://www.ietf.org/html.charters/dhc-charter.html>
- [Dhungel 2012]** P. Dhungel, K. W. Ross, M. Steiner., Y. Tian, X. Hei, “Xunlei: Peer-Assisted Download Acceleration on a Massive Scale”, *Passive and Active Measurement Conference (PAM) 2012*, Viena, 2012.
- [Diffie 1976]** W. Diffie, M. E. Hellman, “New Directions in Cryptography”, *IEEE Transactions on Information Theory*, Vol IT-22 (1976), págs. 644–654.

[Diggavi 2004] S. N. Diggavi, N. Al-Dahir, A. Stamoulis, R. Calderbank, “Great Expectations: The Value of Spatial Diversity in Wireless Networks”, *Proceedings of the IEEE*, Vol. 92, Nº 2 (febrero 2004).

[Dilley 2002] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, B. Weihl, “Globally Distributed Content Deliver”, *IEEE Internet Computing* (septiembre–octubre 2002).

[Diot 2000] C. Diot, B. N. Levine, B. Lyles, H. Kassem, D. Balensiefen, “Deployment Issues for the IP Multicast Service and Architecture”, *IEEE Network*, Vol. 14, Nº 1 (enero/febrero 2000) págs. 78–88.

[Dischinger 2007] M. Dischinger, A. Haeberlen, K. Gummadi, S. Saroiu, “Characterizing residential broadband networks”, *Proc. 2007 ACM Internet Measurement Conference*, págs. 24–26.

[Dimitriopoulos 2007] X. Dimitriopoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. C. Claffy, G. Riley, “AS Relationships: Inference and Validation”, *ACM Computer Communication Review* (enero 2007).

[DOCSIS 2011] Data-Over-Cable Service Interface Specifications, DOCSIS 3.0: MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.0-I16-110623, 2011.

[Dodge 2016] M. Dodge, “An Atlas of Cyberspaces”, http://www.cybergeography.org/atlas/isp_maps.html

[Donahoo 2001] M. Donahoo, K. Calvert, *TCP/IP Sockets in C: Practical Guide for Programmers*, Morgan Kaufman, 2001.

[DSL 2016] DSL Forum homepage, <http://www.dslforum.org/>

[Dhungel 2008] P. Dhungel, D. Wu, B. Schonhorst, K.W. Ross, “A Measurement Study of Attacks on BitTorrent Leechers”, *7th International Workshop on Peer-to-Peer Systems (IPTPS 2008)* (Tampa Bay, FL, febrero 2008).

[Droms 2002] R. Droms, T. Lemon, *The DHCP Handbook* (2nd Edition), SAMS Publishing, 2002.

[Edney 2003] J. Edney and W. A. Arbaugh, *Real 802.11 Security: Wi-Fi Protected Access and 802.11i*, Addison-Wesley Professional, 2003.

[Edwards 2011] W. K. Edwards, R. Grinter, R. Mahajan, D. Wetherall, “Advancing the State of Home Networking”, *Communications of the ACM*, Vol. 54, Nº 6 (junio 2011), págs. 62–71.

[Ellis 1987] H. Ellis, “The Story of Non-Secret Encryption”, <http://jya.com/ellisdoc.htm>

[Erickson 2013] D. Erickson, “The Beacon Openflow Controller”, 2nd *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (HotSDN ’13). ACM, Nueva York, NY, USA, págs. 13–18.

[Ericsson 2012] Ericsson, “The Evolution of Edge”, http://www.ericsson.com/technology/whitepapers/broadband/evolution_of_EDGE.shtml

[Facebook 2014] A. Andreyev, “Introducing Data Center Fabric, the Next-Generation Facebook Data Center Network”, <https://code.facebook.com/posts/360346274145943/introducing-data-center-fabric-the-next-generation-facebook-data-center-network>

[Faloutsos 1999] C. Faloutsos, M. Faloutsos, P. Faloutsos, “What Does the Internet Look Like? Empirical Laws of the Internet Topology”, *Proc. 1999 ACM SIGCOMM* (Boston, MA, agosto 1999).

[Farrington 2010] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, “Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers”, *Proc. 2010 ACM SIGCOMM*.

- [Fteamster 2004]** N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, K. van der Merwe, “The Case for Separating Routing from Routers”, *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, septiembre 2004.
- [Fteamster 2004]** N. Feamster, J. Winick, J. Rexford, “A Model for BGP Routing for Network Engineering”, *Proc. 2004 ACM SIGMETRICS* (Nueva York, NY, junio 2004).
- [Fteamster 2005]** N. Feamster, H. Balakrishnan, “Detecting BGP Configuration Faults with Static Analysis”, *NSDI* (mayo 2005).
- [Fteamster 2013]** N. Feamster, J. Rexford, E. Zegura, “The Road to SDN”, *ACM Queue*, Volume 11, Issue 12, (diciembre 2013).
- [Feldmeier 1995]** D. Feldmeier, “Fast Software Implementation of Error Detection Codes”, *IEEE/ACM Transactions on Networking*, Vol. 3, Nº 6 (diciembre 1995), págs. 640–652.
- [Ferguson 2013]** A. Ferguson, A. Guha, C. Liang, R. Fonseca, S. Krishnamurthi, “Participatory Networking: An API for Application Control of SDNs”, *Proceedings ACM SIGCOMM 2013*, págs. 327–338.
- [Fielding 2000]** R. Fielding, “Architectural Styles and the Design of Network-based Software Architectures”, 2000. PhD Thesis, UC Irvine, 2000.
- [FIPS 1995]** Federal Information Processing Standard, “Secure Hash Standard”, FIPS Publication 180-1. <http://www.itl.nist.gov/fipspubs/fip180-1.htm>
- [Floyd 1999]** S. Floyd, K. Fall, “Promoting the Use of End-to-End Congestion Control in the Internet”, *IEEE/ACM Transactions on Networking*, Vol. 6, Nº 5 (octubre 1998), págs. 458–472.
- [Floyd 2000]** S. Floyd, M. Handley, J. Padhye, J. Widmer, “Equation-Based Congestion Control for Unicast Applications”, *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, agosto 2000).
- [Floyd 2001]** S. Floyd, “A Report on Some Recent Developments in TCP Congestion Control”, *IEEE Communications Magazine* (abril 2001).
- [Floyd 2016]** S. Floyd, “References on RED (Random Early Detection) Queue Management”, <http://www.icir.org/floyd/red.html>
- [Floyd Synchronization 1994]** S. Floyd, V. Jacobson, “Synchronization of Periodic Routing Messages”, *IEEE/ACM Transactions on Networking*, Vol. 2, Nº 2 (abril 1997) págs. 122–136.
- [Floyd TCP 1994]** S. Floyd, “TCP and Explicit Congestion Notification”, *ACM SIGCOMM Computer Communications Review*, Vol. 24, Nº 5 (octubre 1994), págs. 10–23.
- [Fluhrer 2001]** S. Fluhrer, I. Mantin, A. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4”, *Eighth Annual Workshop on Selected Areas in Cryptography* (Toronto, Canadá, agosto 2002).
- [Fortz 2000]** B. Fortz, M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights”, *Proc. 2000 IEEE INFOCOM* (Tel Aviv, Israel, abril 2000).
- [Fortz 2002]** B. Fortz, J. Rexford, M. Thorup, “Traffic Engineering with Traditional IP Routing Protocols”, *IEEE Communication Magazine* (octubre 2002).
- [Fraleigh 2003]** C. Fraleigh, F. Tobagi, C. Diot, “Provisioning IP Backbone Networks to Support Latency Sensitive Traffic”, *Proc. 2003 IEEE INFOCOM* (San Francisco, CA, marzo 2003).
- [Frost 1994]** J. Frost, “BSD Sockets: A Quick and Dirty Primer”, <http://world.std.com/~jimf/papers/sockets/sockets.html>

[**FTC 2015**] Internet of Things: Privacy and Security in a Connected World, Federal Trade Commission, 2015, <https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf>

[**FTTH 2016**] Fiber to the Home Council, <http://www.ftthcouncil.org/>

[**Gao 2001**] L. Gao, J. Rexford, “Stable Internet Routing Without Global Coordination”, *IEEE/ACM Transactions on Networking*, Vol. 9, Nº 6 (diciembre 2001), págs. 681–692.

[**Gartner 2014**] Gartner report on Internet of Things, <http://www.gartner.com/technology/research/internet-of-things>

[**Gauthier 1999**] L. Gauthier, C. Diot, and J. Kurose, “End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet”, *Proc. 1999 IEEE INFOCOM* (Nueva York, NY, abril 1999).

[**Gember-Jacobson 2014**] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, A. Akella, “OpenNF: Enabling Innovation in Network Function Control”, *Proc. ACM SIGCOMM 2014*, págs. 163–174.

[**Goodman 1997**] David J. Goodman, *Wireless Personal Communications Systems*, Prentice-Hall, 1997.

[**Google IPv6 2015**] Google Inc. “IPv6 Statistics”, <https://www.google.com/intl/en/ipv6/statistics.html>

[**Google Locations 2016**] Google data centers. <http://www.google.com/corporate/datacenter/locations.html>

[**Goralski 1999**] W. Goralski, *Frame Relay for High-Speed Networks*, John Wiley, Nueva York, 1999.

[**Greenberg 2009a**] A. Greenberg, J. Hamilton, D. Maltz, P. Patel, “The Cost of a Cloud: Research Problems in Data Center Networks”, *ACM Computer Communications Review* (enero 2009).

[**Greenberg 2009b**] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, S. Sengupta, “VL2: A Scalable and Flexible Data Center Network”, *Proc. 2009 ACM SIGCOMM*.

[**Greenberg 2011**] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, S. Sengupta, “VL2: A Scalable and Flexible Data Center Network”, *Communications of the ACM*, Vol. 54, Nº 3 (marzo 2011), págs. 95–104.

[**Greenberg 2015**] A. Greenberg, “SDN for the Cloud”, Sigcomm 2015 Keynote Address, <http://conferences.sigcomm.org/sigcomm/2015/pdf/papers/keynote.pdf>

[**Griffin 2012**] T. Griffin, “Interdomain Routing Links”, <http://www.cl.cam.ac.uk/~tgg22/interdomain/>

[**Gude 2008**] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “NOX: Towards an Operating System for Networks”, *ACM SIGCOMM Computer Communication Review*, julio 2008.

[**Guha 2006**] S. Guha, N. Daswani, R. Jain, “An Experimental Study of the Skype Peer-to-Peer VoIP System”, *Proc. Fifth Int. Workshop on P2P Systems* (Santa Barbara, CA, 2006).

[**Guo 2005**] L. Guo, S. Chen, Z. Xiao, E. Tan, X. Ding, X. Zhang, “Measurement, Analysis, and Modeling of BitTorrent-Like Systems”, *Proc. 2005 ACM Internet Measurement Conference*.

[**Guo 2009**] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, “BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers”, *Proc. 2009 ACM SIGCOMM*.

- [Gupta 2001]** P. Gupta, N. McKeown, “Algorithms for Packet Classification”, *IEEE Network Magazine*, Vol. 15, Nº 2 (marzo/abril 2001), págs. 24–32.
- [Gupta 2014]** A. Gupta, L. Vanbever, M. Shahbaz, S. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, E. Katz-Bassett, “SDX: A Software Defined Internet Exchange”, *Proc. ACM SIGCOMM 2014* (agosto 2014), págs. 551–562.
- [Ha 2008]** S. Ha, I. Rhee, L. Xu, “CUBIC: A New TCP-Friendly High-Speed TCP Variant”, *ACM SIGOPS Operating System Review*, 2008.
- [Halabi 2000]** S. Halabi, *Internet Routing Architectures*, 2nd Ed., Cisco Press, 2000.
- [Hanabali 2005]** A. A. Hanbali, E. Altman, P. Nain, “A Survey of TCP over Ad Hoc Networks”, *IEEE Commun. Surveys and Tutorials*, Vol. 7, Nº 3 (2005), págs. 22–36.
- [Hei 2007]** X. Hei, C. Liang, J. Liang, Y. Liu, K. W. Ross, “A Measurement Study of a Large-scale P2P IPTV System”, *IEEE Trans. on Multimedia* (diciembre 2007).
- [Heidemann 1997]** J. Heidemann, K. Obraczka, J. Touch, “Modeling the Performance of HTTP over Several Transport Protocols”, *IEEE/ACM Transactions on Networking*, Vol. 5, Nº 5 (octubre 1997), págs. 616–630.
- [Held 2001]** G. Held, *Data Over Wireless Networks: Bluetooth, WAP, and Wireless LANs*, McGraw-Hill, 2001.
- [Holland 2001]** G. Holland, N. Vaidya, V. Bahl, “A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks”, *Proc. 2001 ACM Int. Conference of Mobile Computing and Networking (Mobicom01)* (Roma, Italia, julio 2001).
- [Hollot 2002]** C.V. Hollot, V. Misra, D. Towsley, W. Gong, “Analysis and Design of Controllers for AQM Routers Supporting TCP Flows”, *IEEE Transactions on Automatic Control*, Vol. 47, Nº 6 (junio 2002), págs. 945–959.
- [Hong 2013]** C. Hong, S. Kandula, R. Mahajan, M.Zhang, V. Gill, M. Nanduri, R. Wattenhofer, “Achieving High Utilization with Software-driven WAN”, *ACM SIGCOMM Conference* (agosto 2013), págs.15–26.
- [Huang 2002]** C. Haung, V. Sharma, K. Owens, V. Makam, “Building Reliable MPLS Networks Using a Path Protection Mechanism”, *IEEE Communications Magazine*, Vol. 40, Nº 3 (marzo 2002), págs. 156–162.
- [Huang 2005]** Y. Huang, R. Guerin, “Does Over-Provisioning Become More or Less Efficient as Networks Grow Larger?”, *Proc. IEEE Int. Conf. Network Protocols (ICNP)* (Boston MA, noviembre 2005).
- [Huang 2008]** C. Huang, J. Li, A. Wang, K. W. Ross, “Understanding Hybrid CDN-P2P: Why Limelight Needs Its Own Red Swoosh”, *Proc. 2008 NOSSDAV*, Braunschweig, Alemania.
- [Huitema 1998]** C. Huitema, *IPv6: The New Internet Protocol*, 2nd Ed., Prentice Hall, Englewood Cliffs, NJ, 1998.
- [Huston 1999a]** G. Huston, “Interconnection, Peering, and Settlements—Part I”, *The Internet Protocol Journal*, Vol. 2, Nº 1 (marzo 1999).
- [Huston 2004]** G. Huston, “NAT Anatomy: A Look Inside Network Address Translators”, *The Internet Protocol Journal*, Vol. 7, Nº 3 (septiembre 2004).
- [Huston 2008a]** G. Huston, “Confronting IPv4 Address Exhaustion”, <http://www.potaroo.net/ispcol/2008-10/v4depletion.html>

[**Huston 2008b**] G. Huston, G. Michaelson, “IPv6 Deployment: Just where are we?” <http://www.potaroo.net/ispcol/2008-04/ipv6.html>

[**Huston 2011a**] G. Huston, “A Rough Guide to Address Exhaustion”, *The Internet Protocol Journal*, Vol. 14, Nº 1 (marzo 2011).

[**Huston 2011b**] G. Huston, “Transitioning Protocols”, *The Internet Protocol Journal*, Vol. 14, Nº 1 (marzo 2011).

[**IAB 2016**] Internet Architecture Board homepage, <http://www.iab.org/>

[**IANA Protocol Numbers 2016**] Internet Assigned Numbers Authority, Protocol Numbers, <http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>

[**IBM 1997**] IBM Corp., *IBM Inside APPN - The Essential Guide to the Next-Generation SNA*, SG24-3669-03, junio 1997.

[**ICANN 2016**] The Internet Corporation for Assigned Names and Numbers homepage, <http://www.icann.org>

[**IEEE 802 2016**] IEEE 802 LAN/MAN Standards Committee homepage, <http://www.ieee802.org/>

[**IEEE 802.11 1999**] IEEE 802.11, “1999 Edition (ISO/IEC 8802-11: 1999) IEEE Standards for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Network—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification”, <http://standards.ieee.org/getieee802/download/802.11-1999.pdf>

[**IEEE 802.11ac 2013**] IEEE, “802.11ac-2013—IEEE Standard for Information technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Amendment 4: Enhancements for Very High Throughput for Operation in Bands Below 6 GHz.”

[**IEEE 802.11n 2012**] IEEE, “IEEE P802.11—Task Group N—Meeting Update: Status of 802.11n”, http://grouper.ieee.org/groups/802/11/Reports/tgn_update.htm

[**IEEE 802.15 2012**] IEEE 802.15 Working Group for WPAN homepage, <http://grouper.ieee.org/groups/802/15/>.

[**IEEE 802.15.4 2012**] IEEE 802.15 WPAN Task Group 4, <http://www.ieee802.org/15/pub/TG4.html>

[**IEEE 802.16d 2004**] IEEE, “IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems”, <http://standards.ieee.org/getieee802/download/802.16-2004.pdf>

[**IEEE 802.16e 2005**] IEEE, “IEEE Standard for Local and Metropolitan Area Networks, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1”, <http://standards.ieee.org/getieee802/download/802.16e-2005.pdf>

[**IEEE 802.1q 2005**] IEEE, “IEEE Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks”, <http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf>

[**IEEE 802.1X**] IEEE Std 802.1X-2001 Port-Based Network Access Control, http://standards.ieee.org/reading/ieee/std_public/description/lanman/802.1x-2001_desc.html

[**IEEE 802.3 2012**] IEEE, “IEEE 802.3 CSMA/CD (Ethernet)”, <http://grouper.ieee.org/groups/802/3/>

[**IEEE 802.5 2012**] IEEE, IEEE 802.5 homepage, <http://www.ieee802.org/5/www8025org/>

- [IETF 2016]** Internet Engineering Task Force homepage, <http://www.ietf.org>
- [Ihm 2011]** S. Ihm, V. S. Pai, “Towards Understanding Modern Web Traffic”, *Proc. 2011 ACM Internet Measurement Conference* (Berlin).
- [IMAP 2012]** The IMAP Connection, <http://www imap.org/>
- [Intel 2016]** Intel Corp., “Intel 710 Ethernet Adapter”, <http://www.intel.com/content/www/us/en/ethernet-products/converged-network-adapters/ethernet-xl710.html>
- [Internet2 Multicast 2012]** Internet2 Multicast Working Group homepage, <http://www.internet2.edu/multicast/>
- [ISC 2016]** Internet Systems Consortium homepage, <http://www.isc.org>
- [ISI 1979]** Information Sciences Institute, “DoD Standard Internet Protocol”, Internet Engineering Note 123 (diciembre 1979), <http://www.isi.edu/in-notes/ien/ien123.txt>
- [ISO 2016]** International Organization for Standardization homepage, International Organization for Standardization, <http://www.iso.org/>
- [ISO X.680 2002]** International Organization for Standardization, “X.680: ITU-T Recommendation X.680 (2002) Information Technology—Abstract Syntax Notation One (ASN.1): Specification of Basic Notation”, <http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf>
- [ITU 1999]** Asymmetric Digital Subscriber Line (ADSL) Transceivers. ITU-T G.992.1, 1999.
- [ITU 2003]** Asymmetric Digital Subscriber Line (ADSL) Transceivers—Extended Bandwidth ADSL2 (ADSL2Plus). ITU-T G.992.5, 2003.
- [ITU 2005a]** International Telecommunication Union, “ITU-T X.509, The Directory: Public-key and attribute certificate frameworks” (agosto 2005).
- [ITU 2006]** ITU, “G.993.1: Very High Speed Digital Subscriber Line Transceivers (VDSL)”, <https://www.itu.int/rec/T-REC-G.993.1-200406-I/en>, 2006.
- [ITU 2015]** “Measuring the Information Society Report”, 2015, <http://www.itu.int/en/ITU-D/Statistics/Pages/publications/mis2015.aspx>
- [ITU 2012]** The ITU homepage, <http://www.itu.int>
- [ITU-T Q.2931 1995]** International Telecommunication Union, “Recommendation Q.2931 (02/95)—Broadband Integrated Services Digital Network (B-ISDN)—Digital Subscriber Signalling System N° 2 (DSS 2)—User-Network Interface (UNI)—Layer 3 Specification for Basic Call/Connection Control.”
- [IXP List 2016]** List of IXPs, Wikipedia, https://en.wikipedia.org/wiki/List_of_Internet_exchange_points
- [Iyengar 2015]** J. Iyengar, I. Swett, “QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2”, Internet Draft <draft-tsvwg-quic-protocol-00>, junio 2015.
- [Iyer 2008]** S. Iyer, R. R. Kompella, N. McKeown, “Designing Packet Buffers for Router Line Cards”, *IEEE Transactions on Networking*, Vol. 16, N° 3 (junio 2008), págs. 705–717.
- [Jacobson 1988]** V. Jacobson, “Congestion Avoidance and Control”, *Proc. 1988 ACM SIGCOMM* (Stanford, CA, agosto 1988), págs. 314–329.
- [Jain 1986]** R. Jain, “A Timeout-Based Congestion Control Scheme for Window Flow-Controlled Networks”, *IEEE Journal on Selected Areas in Communications SAC-4*, 7 (octubre 1986).

- [Jain 1989] R. Jain, “A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks”, *ACM SIGCOMM Computer Communications Review*, Vol. 19, Nº 5 (1989), págs. 56–71.
- [Jain 1994] R. Jain, *FDDI Handbook: High-Speed Networking Using Fiber and Other Media*, Addison-Wesley, Reading, MA, 1994.
- [Jain 1996] R. Jain, S. Kalyanaraman, S. Fahmy, R. Goyal, S. Kim, “Tutorial Paper on ABR Source Behavior”, *ATM Forum/96-1270*, octubre 1996. <http://www.cse.wustl.edu/~jain/atmf/ftp/atm96-1270.pdf>
- [Jain 2013] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hözlze, S. Stuart, A. Vahdat, “B4: Experience with a Globally Deployed Software Defined Wan”, *ACM SIGCOMM 2013*, págs. 3–14.
- [Jaiswal 2003] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, “Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP backbone”, *Proc. 2003 IEEE INFOCOM*.
- [Ji 2003] P. Ji, Z. Ge, J. Kurose, D. Towsley, “A Comparison of Hard-State and Soft-State Signaling Protocols”, *Proc. 2003 ACM SIGCOMM* (Karlsruhe, Alemania, agosto 2003).
- [Jimenez 1997] D. Jimenez, “Outside Hackers Infiltrate MIT Network, Compromise Security”, *The Tech*, Vol. 117, No 49 (octubre 1997), p. 1, <http://www-tech.mit.edu/V117/N49/hackers.49n.html>
- [Jin 2004] C. Jin, D. X. We, S. Low, “FAST TCP: Motivation, Architecture, Algorithms, Performance”, *Proc. 2004 IEEE INFOCOM* (Hong Kong, marzo 2004).
- [Juniper Contrail 2016] Juniper Networks, “Contrail”, <http://www.juniper.net/us/en/products-services/sdn/contrail/>
- [Juniper MX2020 2015] Juniper Networks, “MX2020 and MX2010 3D Universal Edge Routers”, www.juniper.net/us/en/local/pdf/.../1000417-en.pdf
- [Kaaranen 2001] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen, V. Niemi, *Networks: Architecture, Mobility and Services*, Nueva York: John Wiley & Sons, 2001.
- [Kahn 1967] D. Kahn, *The Codebreakers: The Story of Secret Writing*, The Macmillan Company, 1967.
- [Kahn 1978] R. E. Kahn, S. Gronemeyer, J. Burchfiel, R. Kunzelman, “Advances in Packet Radio Technology”, *Proc. 1978 IEEE INFOCOM*, 66, 11 (noviembre 1978).
- [Kamerman 1997] A. Kamerman, L. Monteban, “WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band”, *Bell Labs Technical Journal* (verano 1997), págs. 118–133.
- [Kar 2000] K. Kar, M. Kodialam, T. V. Lakshman, “Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications”, *IEEE J. Selected Areas in Communications* (diciembre 2000).
- [Karn 1987] P. Karn, C. Partridge, “Improving Round-Trip Time Estimates in Reliable Transport Protocols”, *Proc. 1987 ACM SIGCOMM*.
- [Karol 1987] M. Karol, M. Hluchyj, A. Morgan, “Input Versus Output Queuing on a Space-Division Packet Switch”, *IEEE Transactions on Communications*, Vol. 35, Nº 12 (diciembre 1987), págs. 1347–1356.
- [Kaufman 1995] C. Kaufman, R. Perlman, M. Speciner, *Network Security, Private Communication in a Public World*, Prentice Hall, Englewood Cliffs, NJ, 1995.

- [Kelly 1998]** F. P. Kelly, A. Maulloo, D. Tan, “Rate Control for Communication Networks: Shadow Prices, Proportional Fairness and Stability”, *J. Operations Res. Soc.*, Vol. 49, Nº 3 (marzo 1998), págs. 237–252.
- [Kelly 2003]** T. Kelly, “Scalable TCP: Improving Performance in High Speed Wide Area Networks”, *ACM SIGCOMM Computer Communications Review*, Vol. 33, Nº 2 (abril 2003), pp 83–91.
- [Kilkki 1999]** K. Kilkki, *Differentiated Services for the Internet*, Macmillan Technical Publishing, Indianapolis, IN, 1999.
- [Kim 2005]** H. Kim, S. Rixner, V. Pai, “Network Interface Data Caching”, *IEEE Transactions on Computers*, Vol. 54, Nº 11 (noviembre 2005), págs. 1394–1408.
- [Kim 2008]** C. Kim, M. Caesar, J. Rexford, “Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises”, *Proc. 2008 ACM SIGCOMM* (Seattle, WA, agosto 2008).
- [Kleinrock 1961]** L. Kleinrock, “Information Flow in Large Communication Networks”, RLE Quarterly Progress Report, julio 1961.
- [Kleinrock 1964]** L. Kleinrock, *1964 Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill, Nueva York, NY, 1964.
- [Kleinrock 1975]** L. Kleinrock, *Queuing Systems, Vol. 1*, John Wiley, Nueva York, 1975.
- [Kleinrock 1975b]** L. Kleinrock, F. A. Tobagi, “Packet Switching in Radio Channels: Part I—Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics”, *IEEE Transactions on Communications*, Vol. 23, Nº 12 (diciembre 1975), págs. 1400–1416.
- [Kleinrock 1976]** L. Kleinrock, *Queuing Systems, Vol. 2*, John Wiley, Nueva York, 1976.
- [Kleinrock 2004]** L. Kleinrock, “The Birth of the Internet”, <http://www.lk.cs.ucla.edu/LK/Inet/birth.html>
- [Kohler 2006]** E. Kohler, M. Handley, S. Floyd, “DDCP: Designing DCCP: Congestion Control Without Reliability”, *Proc. 2006 ACM SIGCOMM* (Pisa, Italia, septiembre 2006).
- [Kolding 2003]** T. Kolding, K. Pedersen, J. Wigard, F. Frederiksen, P. Mogensen, “High Speed Downlink Packet Access: WCDMA Evolution”, *IEEE Vehicular Technology Society News* (febrero 2003), págs. 4–10.
- [Koponen 2010]** T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, S. Shenker, “Onix: A Distributed Control Platform for Large-Scale Production Networks”, *9th USENIX conference on Operating systems design and implementation (OSDI'10)*, págs. 1–6.
- [Koponen 2011]** T. Koponen, S. Shenker, H. Balakrishnan, N. Feamster, I. Ganichev, A. Ghodsi, P. B. Godfrey, N. McKeown, G. Parulkar, B. Raghavan, J. Rexford, S. Arianfar, D. Kuptsov, “Architecting for Innovation”, *ACM Computer Communications Review*, 2011.
- [Korhonen 2003]** J. Korhonen, *Introduction to 3G Mobile Communications*, 2nd ed., Artech House, 2003.
- [Koziol 2003]** J. Koziol, *Intrusion Detection with Snort*, Sams Publishing, 2003.
- [Kreutz 2015]** D. Kreutz, F.M.V. Ramos, P. Esteves Verissimo, C. Rothenberg, S. Azodolmolky, S. Uhlig, “Software-Defined Networking: A Comprehensive Survey”, *Proceedings of the IEEE*, Vol. 103, Nº 1 (enero 2015), págs. 14–76. Este artículo se actualiza también en <https://github.com/SDN-Survey/latex/wiki>

[Krishnamurthy 2001] B. Krishnamurthy, J. Rexford, *Web Protocols and Practice: HTTP/1.1, Networking Protocols, and Traffic Measurement*, Addison-Wesley, Boston, MA, 2001.

[Kulkarni 2005] S. Kulkarni, C. Rosenberg, “Opportunistic Scheduling: Generalizations to Include Multiple Constraints, Multiple Interfaces, and Short Term Fairness”, *Wireless Networks*, 11 (2005), 557–569.

[Kumar 2006] R. Kumar, K.W. Ross, “Optimal Peer-Assisted File Distribution: Single and Multi-Class Problems”, *IEEE Workshop on Hot Topics in Web Systems and Technologies* (Boston, MA, 2006).

[Labovitz 1997] C. Labovitz, G. R. Malan, F. Jahanian, “Internet Routing Instability”, *Proc. 1997 ACM SIGCOMM* (Cannes, Francia, septiembre 1997), págs. 115–126.

[Labovitz 2010] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, “Internet Inter-Domain Traffic”, *Proc. 2010 ACM SIGCOMM*.

[Labrador 1999] M. Labrador, S. Banerjee, “Packet Dropping Policies for ATM and IP Networks”, *IEEE Communications Surveys*, Vol. 2, Nº 3 (Third Quarter 1999), págs. 2–14.

[Lacage 2004] M. Lacage, M.H. Manshaei, T. Turletti, “IEEE 802.11 Rate Adaptation: A Practical Approach”, *ACM Int. Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)* (Venecia, Italia, octubre 2004).

[Lakhina 2004] A. Lakhina, M. Crovella, C. Diot, “Diagnosing Network-Wide Traffic Anomalies”, *Proc. 2004 ACM SIGCOMM*.

[Lakhina 2005] A. Lakhina, M. Crovella, C. Diot, “Mining Anomalies Using Traffic Feature Distributions”, *Proc. 2005 ACM SIGCOMM*.

[Lakshman 1997] T. V. Lakshman, U. Madhow, “The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss”, *IEEE/ACM Transactions on Networking*, Vol. 5, Nº 3 (1997), págs. 336–350.

[Lakshman 2004] T. V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, T. Woo, “The SoftRouter Architecture”, *Proc. 3rd ACM Workshop on Hot Topics in Networks (Hotnets-III)*, noviembre 2004.

[Lam 1980] S. Lam, “A Carrier Sense Multiple Access Protocol for Local Networks”, *Computer Networks*, Vol. 4 (1980), págs. 21–32.

[Lamport 1989] L. Lamport, “The Part-Time Parliament”, Technical Report 49, Systems Research Center, Digital Equipment Corp., Palo Alto, septiembre 1989.

[Lampson 1983] Lampson, Butler W. “Hints for computer system design”, *ACM SIGOPS Operating Systems Review*, Vol. 17, Nº 5, 1983.

[Lampson 1996] B. Lampson, “How to Build a Highly Available System Using Consensus”, *Proc. 10th International Workshop on Distributed Algorithms* (WDAG ’96), Özalp Babaoglu and Keith Marzullo (Eds.), Springer-Verlag, págs. 1–17.

[Lawton 2001] G. Lawton, “Is IPv6 Finally Gaining Ground?” *IEEE Computer Magazine* (agosto 2001), págs. 11–15.

[LeBlond 2011] S. Le Blond, C. Zhang, A. Legout, K. Ross, W. Dabbous. 2011, “I know where you are and what you are sharing: exploiting P2P communications to invade users’ privacy.” *2011 ACM Internet Measurement Conference*, ACM, Nueva York, NY, USA, págs. 45–60.

[Leighton 2009] T. Leighton, “Improving Performance on the Internet”, *Communications of the ACM*, Vol. 52, Nº 2 (febrero 2009), págs. 44–51.

- [Leiner 1998]** B. Leiner, V. Cerf, D. Clark, R. Kahn, L. Kleinrock, D. Lynch, J. Postel, L. Roberts, S. Woolf, “A Brief History of the Internet”, <http://www.isoc.org/internet/history/brief.html>
- [Leung 2006]** K. Leung, V. O.K. Li, “TCP in Wireless Networks: Issues, Approaches, and Challenges”, *IEEE Commun. Surveys and Tutorials*, Vol. 8, Nº 4 (2006), págs. 64–79.
- [Levin 2012]** D. Levin, A. Wundsam, B. Heller, N. Handigol, A. Feldmann, “Logically Centralized?: State Distribution Trade-offs in Software Defined Networks”, *Proc. First Workshop on Hot Topics in Software Defined Networks* (agosto 2012), págs. 1–6.
- [Li 2004]** L. Li, D. Alderson, W. Willinger, J. Doyle, “A First-Principles Approach to Understanding the Internet’s Router-Level Topology”, *Proc. 2004 ACM SIGCOMM* (Portland, OR, agosto 2004).
- [Li 2007]** J. Li, M. Guidero, Z. Wu, E. Purpus, T. Ehrenkranz, “BGP Routing Dynamics Revisited.” *ACM Computer Communication Review* (abril 2007).
- [Li 2015]** S.Q. Li, “Building Softcom Ecosystem Foundation”, Open Networking Summit, 2015.
- [Lin 2001]** Y. Lin, I. Chlamtac, *Wireless and Mobile Network Architectures*, John Wiley and Sons, Nueva York, NY, 2001.
- [Liogkas 2006]** N. Liogkas, R. Nelson, E. Kohler, L. Zhang, “Exploiting BitTorrent for Fun (but Not Profit)”, *6th International Workshop on Peer-to-Peer Systems (IPTPS 2006)*.
- [Liu 2003]** J. Liu, I. Matta, M. Crovella, “End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment”, *Proc. WiOpt’03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*.
- [Locher 2006]** T. Locher, P. Moor, S. Schmid, R. Wattenhofer, “Free Riding in BitTorrent is Cheap”, *Proc. ACM HotNets 2006* (Irvine CA, noviembre 2006).
- [Lui 2004]** J. Lui, V. Misra, D. Rubenstein, “On the Robustness of Soft State Protocols”, *Proc. IEEE Int. Conference on Network Protocols (ICNP ’04)*, págs. 50–60.
- [Mahdavi 1997]** J. Mahdavi, S. Floyd, “TCP-Friendly Unicast Rate-Based Flow Control”, unpublished note (enero 1997).
- [MaxMind 2016]** <http://www.maxmind.com/app/ip-location>
- [Maymounkov 2002]** P. Maymounkov, D. Mazières, “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric.” *Proceedings of the 1st International Workshop on Peerto-Peer Systems (IPTPS ’02)* (marzo 2002), págs. 53–65.
- [McKeown 1997a]** N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, M. Horowitz, “The Tiny Tera: A Packet Switch Core”, *IEEE Micro Magazine* (enero–febrero 1997).
- [McKeown 1997b]** N. McKeown, “A Fast Switched Backplane for a Gigabit Switched Router”, *Business Communications Review*, Vol. 27, Nº 12. http://tiny-tera.stanford.edu/~nickm/papers/cisco_fast_wp.pdf
- [McKeown 2008]** N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner. 2008. OpenFlow: Enabling Innovation in Campus Networks. *SIGCOMM Comput. Commun. Rev.* 38, 2 (marzo 2008), págs. 69–74.
- [McQuillan 1980]** J. McQuillan, I. Richer, E. Rosen, “The New Routing Algorithm for the Arpanet”, *IEEE Transactions on Communications*, Vol. 28, Nº 5 (mayo 1980), págs. 711–719.
- [Metcalfe 1976]** R. M. Metcalfe, D. R. Boggs. “Ethernet: Distributed Packet Switching for Local Computer Networks”, *Communications of the Association for Computing Machinery*, Vol. 19, Nº 7 (julio 1976), págs. 395–404.

[Meyers 2004] A. Myers, T. Ng, H. Zhang, “Rethinking the Service Model: Scaling Ethernet to a Million Nodes”, *ACM Hotnets Conference*, 2004.

[MFA Forum 2016] IP/MPLS Forum homepage, <http://www.ipmplsforum.org/>

[Mockapetris 1988] P. V. Mockapetris, K. J. Dunlap, “Development of the Domain Name System”, *Proc. 1988 ACM SIGCOMM* (Stanford, CA, agosto 1988).

[Mockapetris 2005] P. Mockapetris, Sigcomm Award Lecture, video available at <http://www.postel.org/sigcomm>

[Molinero-Fernandez 2002] P. Molinaro-Fernandez, N. McKeown, H. Zhang, “Is IP Going to Take Over the World (of Communications)?” *Proc. 2002 ACM Hotnets*.

[Molle 1987] M. L. Molle, K. Sohraby, A. N. Venetsanopoulos, “Space-Time Models of Asynchronous CSMA Protocols for Local Area Networks”, *IEEE Journal on Selected Areas in Communications*, Vol. 5, Nº 6 (1987), págs. 956–968.

[Moore 2001] D. Moore, G. Voelker, S. Savage, “Inferring Internet Denial of Service Activity”, *Proc. 2001 USENIX Security Symposium* (Washington, DC, agosto 2001).

[Motorola 2007] Motorola, “Long Term Evolution (LTE): A Technical Overview”, http://www.motorola.com/staticfiles/Business/Solutions/Industry%20Solutions/Service%20Providers/Wireless%20Operators/LTE/_Document/Static%20Files/6834_MotDoc_New.pdf

[Mouly 1992] M. Mouly, M. Pautet, *The GSM System for Mobile Communications*, Cell and Sys, Palaiseau, Francia, 1992.

[Moy 1998] J. Moy, *OSPF: Anatomy of An Internet Routing Protocol*, Addison-Wesley, Reading, MA, 1998.

[Mukherjee 1997] B. Mukherjee, *Optical Communication Networks*, McGraw-Hill, 1997.

[Mukherjee 2006] B. Mukherjee, *Optical WDM Networks*, Springer, 2006.

[Mysore 2009] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, A. Vahdat, “PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric”, *Proc. 2009 ACM SIGCOMM*.

[Nahum 2002] E. Nahum, T. Barzilai, D. Kandlur, “Performance Issues in WWW Servers”, *IEEE/ACM Transactions on Networking*, Vol 10, Nº 1 (febrero 2002).

[Netflix Open Connect 2016] Netflix Open Connect CDN, 2016, <https://openconnect.netflix.com/>

[Netflix Video 1] Designing Netflix’s Content Delivery System, D. Fulllager, 2014, <https://www.youtube.com/watch?v=LkLLpYdDINA>

[Netflix Video 2] Scaling the Netflix Global CDN, D. Temkin, 2015, https://www.youtube.com/watch?v=tbqcsHg-Q_o

[Neumann 1997] R. Neumann, “Internet Routing Black Hole”, *The Risks Digest: Forum on Risks to the Public in Computers and Related Systems*, Vol. 19, Nº 12 (mayo 1997). <http://catless.ncl.ac.uk/Risks/19.12.html#subj1.1>

[Neville-Neil 2009] G. Neville-Neil, “Whither Sockets?” *Communications of the ACM*, Vol. 52, Nº 6 (junio 2009), págs. 51–55.

[Nicholson 2006] A Nicholson, Y. Chawathe, M. Chen, B. Noble, D. Wetherall, “Improved Access Point Selection”, *Proc. 2006 ACM Mobicom Conference* (Uppsala Sweden, 2006).

- [Nielsen 1997]** H. F. Nielsen, J. Gettys, A. Baird-Smith, E. Prud'hommeaux, H. W. Lie, C. Lilley, “Network Performance Effects of HTTP/1.1, CSS1, and PNG”, *W3C Document*, 1997 (también en *Proc. 1997 ACM SIGCOM* (Cannes, Francia, septiembre 1997), págs. 155–166).
- [NIST 2001]** National Institute of Standards and Technology, “Advanced Encryption Standard (AES)”, Federal Information Processing Standards 197, noviembre 2001, <http://csrc.nist.gov/publications/fips/fips-197/fips-197.pdf>
- [NIST IPv6 2015]** US National Institute of Standards and Technology, “Estimating IPv6 & DNSSEC Deployment SnapShots”, <http://fedv6-deployment.antd.nist.gov/snap-all.html>
- [Nmap 2012]** Nmap homepage, <http://www.insecure.com/nmap>
- [Nonnenmacher 1998]** J. Nonnenmacher, E. Biersak, D. Towsley, “Parity-Based Loss Recovery for Reliable Multicast Transmission”, *IEEE/ACM Transactions on Networking*, Vol. 6, Nº 4 (agosto 1998), págs. 349–361.
- [Nygren 2010]** Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun, “The Akamai Network: A Platform for High-performance Internet Applications”, *SIGOPS Oper. Syst. Rev.* 44, 3 (agosto 2010), 2–19.
- [ONF 2016]** Open Networking Foundation, Technical Library, <https://www.opennetworking.org/sdn-resources/technical-library>
- [ONOS 2016]** Open Network Operating System (ONOS), “Architecture Guide”, <https://wiki.onosproject.org/display/ONOS/Architecture+Guide>, 2016.
- [OpenFlow 2009]** Open Network Foundation, “OpenFlow Switch Specification 1.0.0, TS-001”, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf>
- [OpenDaylight Lithium 2016]** OpenDaylight, “Lithium”, <https://www.opendaylight.org/lithium>
- [OSI 2012]** International Organization for Standardization homepage, <http://www.iso.org/iso/en/ISOOnline.frontpage>
- [Osterweil 2012]** E. Osterweil, D. McPherson, S. DiBenedetto, C. Papadopoulos, D. Massey, “Behavior of DNS Top Talkers”, *Passive and Active Measurement Conference*, 2012.
- [Padhye 2000]** J. Padhye, V. Firoiu, D. Towsley, J. Kurose, “Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation”, *IEEE/ACM Transactions on Networking*, Vol. 8 Nº 2 (abril 2000), págs. 133–145.
- [Padhye 2001]** J. Padhye, S. Floyd, “On Inferring TCP Behavior”, *Proc. 2001 ACM SIGCOMM* (San Diego, CA, agosto 2001).
- [Palat 2009]** S. Palat, P. Godin, “The LTE Network Architecture: A Comprehensive Tutorial”, in *LTE—The UMTS Long Term Evolution: From Theory to Practice*. Also available as a standalone Alcatel white paper.
- [Panda 2013]** A. Panda, C. Scott, A. Ghodsi, T. Koponen, S. Shenker, “CAP for Networks”, *Proc. ACM HotSDN '13*, págs. 91–96.
- [Parekh 1993]** A. Parekh, R. Gallagher, “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case”, *IEEE/ACM Transactions on Networking*, Vol. 1, Nº 3 (junio 1993), págs. 344–357.
- [Partridge 1992]** C. Partridge, S. Pink, “An Implementation of the Revised Internet Stream Protocol (ST-2)”, *Journal of Internetworking: Research and Experience*, Vol. 3, Nº 1 (marzo 1992).

[Partridge 1998] C. Partridge, et al. “A Fifty Gigabit per second IP Router”, *IEEE/ACM Transactions on Networking*, Vol. 6, Nº 3 (Jun. 1998), págs. 237–248.

[Pathak 2010] A. Pathak, Y. A. Wang, C. Huang, A. Greenberg, Y. C. Hu, J. Li, K. W. Ross, “Measuring and Evaluating TCP Splitting for Cloud Services”, *Passive and Active Measurement (PAM) Conference* (Zurich, 2010).

[Perkins 1994] A. Perkins, “Networking with Bob Metcalfe”, *The Red Herring Magazine* (noviembre 1994).

[Perkins 1998] C. Perkins, O. Hodson, V. Hardman, “A Survey of Packet Loss Recovery Techniques for Streaming Audio”, *IEEE Network Magazine* (septiembre/octubre 1998), págs. 40–47.

[Perkins 1998b] C. Perkins, *Mobile IP: Design Principles and Practice*, Addison-Wesley, Reading, MA, 1998.

[Perkins 2000] C. Perkins, *Ad Hoc Networking*, Addison-Wesley, Reading, MA, 2000.

[Perlman 1999] R. Perlman, *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols*, 2nd ed., Addison-Wesley Professional Computing Series, Reading, MA, 1999.

[PGPI 2016] The International PGP homepage, <http://www.pgpi.org>

[Phifer 2000] L. Phifer, “The Trouble with NAT”, *The Internet Protocol Journal*, Vol. 3, Nº 4 (diciembre 2000), http://www.cisco.com/warp/public/759/ipj_3-4/ipj_3-4_nat.html

[Piatek 2007] M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, A. Venkataramani, “Do Incentives Build Robustness in BitTorrent?”, *Proc. NSDI* (2007).

[Piatek 2008] M. Piatek, T. Isdal, A. Krishnamurthy, T. Anderson, “One Hop Reputations for Peer-to-peer File Sharing Workloads”, *Proc. NSDI* (2008).

[Pickholtz 1982] R. Pickholtz, D. Schilling, L. Milstein, “Theory of Spread Spectrum Communication—a Tutorial”, *IEEE Transactions on Communications*, Vol. 30, Nº 5 (mayo 1982), págs. 855–884.

[PingPlotter 2016] PingPlotter homepage, <http://www.pingplotter.com>

[Piscatello 1993] D. Piscatello, A. Lyman Chapin, *Open Systems Networking*, Addison-Wesley, Reading, MA, 1993.

[Pomeranz 2010] H. Pomeranz, “Practical, Visual, Three-Dimensional Pedagogy for Internet Protocol Packet Header Control Fields”, <https://righteousit.wordpress.com/2010/06/27/practical-visual-three-dimensional-pedagogy-for-internet-protocol-packet-header-control-fields/>, junio 2010.

[Potaroo 2016] “Growth of the BGP Table—1994 to Present”, <http://bgp.potaroo.net/>

[PPLive 2012] PPLive homepage, <http://www.pplive.com>

[Qazi 2013] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, M. Yu, “SIMPLE-fying Middlebox Policy Enforcement Using SDN”, *ACM SIGCOMM Conference* (agosto 2013), págs. 27–38.

[Quagga 2012] Quagga, “Quagga Routing Suite”, <http://www.quagga.net/>

[Quittner 1998] J. Quittner, M. Slatalla, *Speeding the Net: The Inside Story of Netscape and How It Challenged Microsoft*, Atlantic Monthly Press, 1998.

[Quova 2016] www.quova.com

[Ramakrishnan 1990] K. K. Ramakrishnan, R. Jain, “A Binary Feedback Scheme for Congestion Avoidance in Computer Networks”, *ACM Transactions on Computer Systems*, Vol. 8, Nº 2 (mayo 1990), págs. 158–181.

- [Raman 1999]** S. Raman, S. McCanne, “A Model, Analysis, and Protocol Framework for Soft State-based Communication”, *Proc. 1999 ACM SIGCOMM* (Boston, MA, agosto 1999).
- [Raman 2007]** B. Raman, K. Chebrolu, “Experiences in Using WiFi for Rural Internet in India”, *IEEE Communications Magazine*, Special Issue on New Directions in Networking Technologies in Emerging Economies (enero 2007).
- [Ramaswami 2010]** R. Ramaswami, K. Sivarajan, G. Sasaki, *Optical Networks: A Practical Perspective*, Morgan Kaufman Publishers, 2010.
- [Ramjee 1994]** R. Ramjee, J. Kurose, D. Towsley, H. Schulzrinne, “Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks”, *Proc. 1994 IEEE INFOCOM*.
- [Rao 2011]** A. S. Rao, Y. S. Lim, C. Barakat, A. Legout, D. Towsley, W. Dabbous, “Network Characteristics of Video Streaming Traffic”, *Proc. 2011 ACM CoNEXT* (Tokio).
- [Ren 2006]** S. Ren, L. Guo, X. Zhang, “ASAP: An AS-Aware Peer-Relay Protocol for High Quality VoIP”, *Proc. 2006 IEEE ICDCS* (Lisboa, Portugal, julio 2006).
- [Rescorla 2001]** E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, Addison-Wesley, Boston, 2001.
- [RFC 001]** S. Crocker, “Host Software”, RFC 001 (the *very first* RFC!).
- [RFC 768]** J. Postel, “User Datagram Protocol”, RFC 768, agosto 1980.
- [RFC 791]** J. Postel, “Internet Protocol: DARPA Internet Program Protocol Specification”, RFC 791, septiembre 1981.
- [RFC 792]** J. Postel, “Internet Control Message Protocol”, RFC 792, septiembre 1981.
- [RFC 793]** J. Postel, “Transmission Control Protocol”, RFC 793, septiembre 1981.
- [RFC 801]** J. Postel, “NCP/TCP Transition Plan”, RFC 801, noviembre 1981.
- [RFC 826]** D. C. Plummer, “An Ethernet Address Resolution Protocol—or—Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware”, RFC 826, noviembre 1982.
- [RFC 829]** V. Cerf, “Packet Satellite Technology Reference Sources”, RFC 829, noviembre 1982.
- [RFC 854]** J. Postel, J. Reynolds, “TELNET Protocol Specification”, RFC 854, mayo 1993.
- [RFC 950]** J. Mogul, J. Postel, “Internet Standard Subnetting Procedure”, RFC 950, agosto 1985.
- [RFC 959]** J. Postel and J. Reynolds, “File Transfer Protocol (FTP)”, RFC 959, octubre 1985.
- [RFC 1034]** P. V. Mockapetris, “Domain Names—Concepts and Facilities”, RFC 1034, noviembre 1987.
- [RFC 1035]** P. Mockapetris, “Domain Names—Implementation and Specification”, RFC 1035, noviembre 1987.
- [RFC 1058]** C. L. Hendrick, “Routing Information Protocol”, RFC 1058, junio 1988.
- [RFC 1071]** R. Braden, D. Borman, and C. Partridge, “Computing the Internet Checksum”, RFC 1071, septiembre 1988.
- [RFC 1122]** R. Braden, “Requirements for Internet Hosts—Communication Layers”, RFC 1122, octubre 1989.
- [RFC 1123]** R. Braden, ed., “Requirements for Internet Hosts—Application and Support”, RFC-1123, octubre 1989.

- [RFC 1142] D. Oran, “OSI IS-IS Intra-Domain Routing Protocol”, RFC 1142, febrero 1990.
- [RFC 1190] C. Topolcic, “Experimental Internet Stream Protocol: Version 2 (ST-II)”, RFC 1190, octubre 1990.
- [RFC 1256] S. Deering, “ICMP Router Discovery Messages”, RFC 1256, septiembre 1991.
- [RFC 1320] R. Rivest, “The MD4 Message-Digest Algorithm”, RFC 1320, abril 1992.
- [RFC 1321] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, abril 1992.
- [RFC 1323] V. Jacobson, S. Braden, D. Borman, “TCP Extensions for High Performance”, RFC 1323, mayo 1992.
- [RFC 1422] S. Kent, “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management”, RFC 1422.
- [RFC 1546] C. Partridge, T. Mendez, W. Milliken, “Host Anycasting Service”, RFC 1546, 1993.
- [RFC 1584] J. Moy, “Multicast Extensions to OSPF”, RFC 1584, marzo 1994.
- [RFC 1633] R. Braden, D. Clark, S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, junio 1994.
- [RFC 1636] R. Braden, D. Clark, S. Crocker, C. Huitema, “Report of IAB Workshop on Security in the Internet Architecture”, RFC 1636, noviembre 1994.
- [RFC 1700] J. Reynolds, J. Postel, “Assigned Numbers”, RFC 1700, octubre 1994.
- [RFC 1752] S. Bradner, A. Mankin, “The Recommendations for the IP Next Generation Protocol”, RFC 1752, enero 1995.
- [RFC 1918] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, “Address Allocation for Private Internets”, RFC 1918, febrero 1996.
- [RFC 1930] J. Hawkinson, T. Bates, “Guidelines for Creation, Selection, and Registration of an Autonomous System (AS)”, RFC 1930, marzo 1996.
- [RFC 1939] J. Myers, M. Rose, “Post Office Protocol—Version 3”, RFC 1939, mayo 1996.
- [RFC 1945] T. Berners-Lee, R. Fielding, H. Frystyk, “Hypertext Transfer Protocol—HTTP/1.0”, RFC 1945, mayo 1996.
- [RFC 2003] C. Perkins, “IP Encapsulation Within IP”, RFC 2003, octubre 1996.
- [RFC 2004] C. Perkins, “Minimal Encapsulation Within IP”, RFC 2004, octubre 1996.
- [RFC 2018] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, “TCP Selective Acknowledgment Options”, RFC 2018, octubre 1996.
- [RFC 2131] R. Droms, “Dynamic Host Configuration Protocol”, RFC 2131, marzo 1997.
- [RFC 2136] P. Vixie, S. Thomson, Y. Rekhter, J. Bound, “Dynamic Updates in the Domain Name System”, RFC 2136, abril 1997.
- [RFC 2205] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, S. Jamin, “Resource ReSerVation Protocol (RSVP)—Version 1 Functional Specification”, RFC 2205, septiembre 1997.
- [RFC 2210] J. Wroclawski, “The Use of RSVP with IETF Integrated Services”, RFC 2210, septiembre 1997.
- [RFC 2211] J. Wroclawski, “Specification of the Controlled-Load Network Element Service”, RFC 2211, septiembre 1997.

- [RFC 2215] S. Shenker, J. Wroclawski, “General Characterization Parameters for Integrated Service Network Elements”, RFC 2215, septiembre 1997.
- [RFC 2326] H. Schulzrinne, A. Rao, R. Lanphier, “Real Time Streaming Protocol (RTSP)”, RFC 2326, abril 1998.
- [RFC 2328] J. Moy, “OSPF Version 2”, RFC 2328, abril 1998.
- [RFC 2420] H. Kummert, “The PPP Triple-DES Encryption Protocol (3DESE)”, RFC 2420, septiembre 1998.
- [RFC 2453] G. Malkin, “RIP Version 2”, RFC 2453, noviembre 1998.
- [RFC 2460] S. Deering, R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification”, RFC 2460, diciembre 1998.
- [RFC 2475] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, “An Architecture for Differentiated Services”, RFC 2475, diciembre 1998.
- [RFC 2578] K. McCloghrie, D. Perkins, J. Schoenwaelder, “Structure of Management Information Version 2 (SMIV2)”, RFC 2578, abril 1999.
- [RFC 2579] K. McCloghrie, D. Perkins, J. Schoenwaelder, “Textual Conventions for SMIV2”, RFC 2579, abril 1999.
- [RFC 2580] K. McCloghrie, D. Perkins, J. Schoenwaelder, “Conformance Statements for SMIV2”, RFC 2580, abril 1999.
- [RFC 2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, “Assured Forwarding PHB Group”, RFC 2597, junio 1999.
- [RFC 2616] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, R. Fielding, “Hypertext Transfer Protocol—HTTP/1.1”, RFC 2616, junio 1999.
- [RFC 2663] P. Srisuresh, M. Holdrege, “IP Network Address Translator (NAT) Terminology and Considerations”, RFC 2663.
- [RFC 2702] D. Awduche, J. Malcolm, J. Agogbua, M. O’Dell, J. McManus, “Requirements for Traffic Engineering Over MPLS”, RFC 2702, septiembre 1999.
- [RFC 2827] P. Ferguson, D. Senie, “Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing”, RFC 2827, mayo 2000.
- [RFC 2865] C. Rigney, S. Willens, A. Rubens, W. Simpson, “Remote Authentication Dial In User Service (RADIUS)”, RFC 2865, junio 2000.
- [RFC 3007] B. Wellington, “Secure Domain Name System (DNS) Dynamic Update”, RFC 3007, noviembre 2000.
- [RFC 3022] P. Srisuresh, K. Egevang, “Traditional IP Network Address Translator (Traditional NAT)”, RFC 3022, enero 2001.
- [RFC 3022] P. Srisuresh, K. Egevang, “Traditional IP Network Address Translator (Traditional NAT)”, RFC 3022, enero 2001.
- [RFC 3031] E. Rosen, A. Viswanathan, R. Callon, “Multiprotocol Label Switching Architecture”, RFC 3031, enero 2001.
- [RFC 3032] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, A. Conta, “MPLS Label Stack Encoding”, RFC 3032, enero 2001.
- [RFC 3168] K. Ramakrishnan, S. Floyd, D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP”, RFC 3168, septiembre 2001.

[RFC 3209] D. Awdanche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, “RSVP-TE: Extensions to RSVP for LSP Tunnels”, RFC 3209, diciembre 2001.

[RFC 3221] G. Huston, “Commentary on Inter-Domain Routing in the Internet”, RFC 3221, diciembre 2001.

[RFC 3232] J. Reynolds, “Assigned Numbers: RFC 1700 Is Replaced by an On-line Database”, RFC 3232, enero 2002.

[RFC 3234] B. Carpenter, S. Brim, “Middleboxes: Taxonomy and Issues”, RFC 3234, febrero 2002.

[RFC 3246] B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, “An Expedited Forwarding PHB (Per-Hop Behavior)”, RFC 3246, marzo 2002.

[RFC 3260] D. Grossman, “New Terminology and Clarifications for Diffserv”, RFC 3260, abril 2002.

[RFC 3261] J. Rosenberg, H. Schulzrinne, G. Carmailllo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, “SIP: Session Initiation Protocol”, RFC 3261, julio 2002.

[RFC 3272] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awdanche, B. Christian, W. S. Lai, “Overview and Principles of Internet Traffic Engineering”, RFC 3272, mayo 2002.

[RFC 3286] L. Ong, J. Yoakum, “An Introduction to the Stream Control Transmission Protocol (SCTP)”, RFC 3286, mayo 2002.

[RFC 3346] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awdanche, B. Christian, W. S. Lai, “Applicability Statement for Traffic Engineering with MPLS”, RFC 3346, agosto 2002.

[RFC 3390] M. Allman, S. Floyd, C. Partridge, “Increasing TCP’s Initial Window”, RFC 3390, octubre 2002.

[RFC 3410] J. Case, R. Mundy, D. Partain, “Introduction and Applicability Statements for Internet Standard Management Framework”, RFC 3410, diciembre 2002.

[RFC 3414] U. Blumenthal and B. Wijnen, “User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3)”, RFC 3414, diciembre 2002.

[RFC 3416] R. Presuhn, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, “Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)”, diciembre 2002.

[RFC 3439] R. Bush, D. Meyer, “Some Internet Architectural Guidelines and Philosophy”, RFC 3439, diciembre 2003.

[RFC 3447] J. Jonsson, B. Kaliski, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1”, RFC 3447, febrero 2003.

[RFC 3468] L. Andersson, G. Swallow, “The Multiprotocol Label Switching (MPLS) Working Group Decision on MPLS Signaling Protocols”, RFC 3468, febrero 2003.

[RFC 3469] V. Sharma, Ed., F. Hellstrand, Ed, “Framework for Multi-Protocol Label Switching (MPLS)-based Recovery”, RFC 3469, febrero. 2003. <ftp://ftp.rfc-editor.org/in-notes/rfc3469.txt>

[RFC 3501] M. Crispin, “Internet Message Access Protocol—Version 4rev1”, RFC 3501, marzo 2003.

[RFC 3550] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications”, RFC 3550, julio 2003.

[RFC 3588] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, J. Arkko, “Diameter Base Protocol”, RFC 3588, septiembre 2003.

- [RFC 3649]** S. Floyd, “HighSpeed TCP for Large Congestion Windows”, RFC 3649, diciembre 2003.
- [RFC 3746]** L. Yang, R. Dantu, T. Anderson, R. Gopal, “Forwarding and Control Element Separation (ForCES Framework”, Internet, RFC 3746, abril 2004.
- [RFC 3748]** B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Ed., “Extensible Authentication Protocol (EAP”, RFC 3748, junio 2004.
- [RFC 3782]** S. Floyd, T. Henderson, A. Gurtov, “The NewReno Modification to TCP’s Fast Recovery Algorithm”, RFC 3782, abril 2004.
- [RFC 4213]** E. Nordmark, R. Gilligan, “Basic Transition Mechanisms for IPv6 Hosts and Routers”, RFC 4213, octubre 2005.
- [RFC 4271]** Y. Rekhter, T. Li, S. Hares, Ed., “A Border Gateway Protocol 4 (BGP-4”, RFC 4271, enero 2006.
- [RFC 4272]** S. Murphy, “BGP Security Vulnerabilities Analysis”, RFC 4274, enero 2006.
- [RFC 4291]** R. Hinden, S. Deering, “IP Version 6 Addressing Architecture”, RFC 4291, febrero 2006.
- [RFC 4340]** E. Kohler, M. Handley, S. Floyd, “Datagram Congestion Control Protocol (DCCP”, RFC 4340, marzo 2006.
- [RFC 4443]** A. Conta, S. Deering, M. Gupta, Ed., “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification”, RFC 4443, marzo 2006.
- [RFC 4346]** T. Dierks, E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1”, RFC 4346, abril 2006.
- [RFC 4514]** K. Zeilenga, Ed., “Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names”, RFC 4514, junio 2006.
- [RFC 4601]** B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, “Protocol Independent Multicast—Sparse Mode (PIM-SM): Protocol Specification (Revised”, RFC 4601, agosto 2006.
- [RFC 4632]** V. Fuller, T. Li, “Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan”, RFC 4632, agosto 2006.
- [RFC 4960]** R. Stewart, ed., “Stream Control Transmission Protocol”, RFC 4960, septiembre 2007.
- [RFC 4987]** W. Eddy, “TCP SYN Flooding Attacks and Common Mitigations”, RFC 4987, agosto 2007.
- [RFC 5000]** RFC editor, “Internet Official Protocol Standards”, RFC 5000, mayo 2008.
- [RFC 5109]** A. Li (ed.), “RTP Payload Format for Generic Forward Error Correction”, RFC 5109, diciembre 2007.
- [RFC 5216]** D. Simon, B. Aboba, R. Hurst, “The EAP-TLS Authentication Protocol”, RFC 5216, marzo 2008.
- [RFC 5218]** D. Thaler, B. Aboba, “What Makes for a Successful Protocol?”, RFC 5218, julio 2008.
- [RFC 5321]** J. Klensin, “Simple Mail Transfer Protocol”, RFC 5321, octubre 2008.
- [RFC 5322]** P. Resnick, Ed., “Internet Message Format”, RFC 5322, octubre 2008.
- [RFC 5348]** S. Floyd, M. Handley, J. Padhye, J. Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification”, RFC 5348, septiembre 2008.

[RFC 5389] J. Rosenberg, R. Mahy, P. Matthews, D. Wing, “Session Traversal Utilities for NAT (STUN)”, RFC 5389, octubre 2008.

[RFC 5411] J Rosenberg, “A Hitchhiker’s Guide to the Session Initiation Protocol (SIP)”, RFC 5411, febrero 2009.

[RFC 5681] M. Allman, V. Paxson, E. Blanton, “TCP Congestion Control”, RFC 5681, septiembre 2009.

[RFC 5944] C. Perkins, Ed., “IP Mobility Support for IPv4, Revised”, RFC 5944, noviembre 2010.

[RFC 6265] A Barth, “HTTP State Management Mechanism”, RFC 6265, abril 2011.

[RFC 6298] V. Paxson, M. Allman, J. Chu, M. Sargent, “Computing TCP’s Retransmission Timer”, RFC 6298, junio 2011.

[RFC 7020] R. Housley, J. Curran, G. Huston, D. Conrad, “The Internet Numbers Registry System”, RFC 7020, agosto 2013.

[RFC 7094] D. McPherson, D. Oran, D. Thaler, E. Osterweil, “Architectural Considerations of IP Anycast”, RFC 7094, enero 2014.

[RFC 7323] D. Borman, R. Braden, V. Jacobson, R. Scheffenegger (ed.), “TCP Extensions for High Performance”, RFC 7323, septiembre 2014.

[RFC 7540] M. Belshe, R. Peon, M. Thomson (Eds), “Hypertext Transfer Protocol Version 2 (HTTP/2)”, RFC 7540, mayo 2015.

[Richter 2015] P. Richter, M. Allman, R. Bush, V. Paxson, “A Primer on IPv4 Scarcity”, *ACM SIGCOMM Computer Communication Review*, Vol. 45, Nº 2 (abril 2015), págs. 21–32.

[Roberts 1967] L. Roberts, T. Merril, “Toward a Cooperative Network of Time-Shared Computers”, *AFIPS Fall Conference* (octubre 1966).

[Rodriguez 2010] R. Rodrigues, P. Druschel, “Peer-to-Peer Systems”, *Communications of the ACM*, Vol. 53, Nº 10 (octubre 2010), págs. 72–82.

[Rohde 2008] Rohde, Schwarz, “UMTS Long Term Evolution (LTE) Technology Introduction”, Application Note 1MA111.

[Rom 1990] R. Rom, M. Sidi, *Multiple Access Protocols: Performance and Analysis*, Springer-Verlag, Nueva York, 1990.

[Root Servers 2016] Root Servers home page, <http://www.root-servers.org/>

[RSA 1978] R. Rivest, A. Shamir, L. Adelman, “A Method for Obtaining Digital Signatures and Public-key Cryptosystems”, *Communications of the ACM*, Vol. 21, Nº 2 (febrero 1978), págs. 120–126.

[RSA Fast 2012] RSA Laboratories, “How Fast Is RSA?” <http://www.rsa.com/rsalabs/node.asp?id=2215>

[RSA Key 2012] RSA Laboratories, “How Large a Key Should Be Used in the RSA Crypto System?” <http://www.rsa.com/rsalabs/node.asp?id=2218>

[Rubenstein 1998] D. Rubenstein, J. Kurose, D. Towsley, “Real-Time Reliable Multicast Using Proactive Forward Error Correction”, *Proceedings of NOSSDAV ‘98* (Cambridge, UK, julio 1998).

[Ruiz-Sánchez 2001] M. Ruiz-Sánchez, E. Biersack, W. Dabbous, “Survey and Taxonomy of IP Address Lookup Algorithms”, *IEEE Network Magazine*, Vol. 15, Nº 2 (marzo/abril 2001), págs. 8–23.

- [Saltzer 1984]** J. Saltzer, D. Reed, D. Clark, “End-to-End Arguments in System Design”, *ACM Transactions on Computer Systems (TOCS)*, Vol. 2, Nº 4 (noviembre 1984).
- [Sandvine 2015]** “Global Internet Phenomena Report, Spring 2011”, <http://www.sandvine.com/news/global broadband trends.asp>, 2011.
- [Sardar 2006]** B. Sardar, D. Saha, “A Survey of TCP Enhancements for Last-Hop Wireless Networks”, *IEEE Commun. Surveys and Tutorials*, Vol. 8, Nº 3 (2006), págs. 20–34.
- [Saroiu 2002]** S. Saroiu, P. K. Gummadi, S. D. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems”, *Proc. of Multimedia Computing and Networking (MMCN)* (2002).
- [Sauter 2014]** M. Sauter, *From GSM to LTE-Advanced*, John Wiley and Sons, 2014.
- [Savage 2015]** D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, R. White, “Enhanced Interior Gateway Routing Protocol”, Internet Draft, draft-savage-eigrp-04.txt, agosto 2015.
- [Saydam 1996]** T. Saydam, T. Magedanz, “From Networks and Network Management into Service and Service Management”, *Journal of Networks and System Management*, Vol. 4, Nº 4 (diciembre 1996), págs. 345–348.
- [Schiller 2003]** J. Schiller, *Mobile Communications* 2nd edition, Addison Wesley, 2003.
- [Schneier 1995]** B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, 1995.
- [Schulzrinne-RTP 2012]** Henning Schulzrinne’s RTP site, <http://www.cs.columbia.edu/~hgs/rtp>
- [Schulzrinne-SIP 2016]** Henning Schulzrinne’s SIP site, <http://www.cs.columbia.edu/~hgs/sip>
- [Schwartz 1977]** M. Schwartz, *Computer-Communication Network Design and Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1997.
- [Schwartz 1980]** M. Schwartz, *Information, Transmission, Modulation, and Noise*, McGraw Hill, Nueva York, NY 1980.
- [Schwartz 1982]** M. Schwartz, “Performance Analysis of the SNA Virtual Route Pacing Control”, *IEEE Transactions on Communications*, Vol. 30, Nº 1 (enero 1982), págs. 172–184.
- [Scourias 2012]** J. Scourias, “Overview of the Global System for Mobile Communications: GSM.” <http://www.privateline.com/PCS/GSM0.html>
- [SDNHub 2016]** SDNHub, “App Development Tutorials”, <http://sdnhub.org/tutorials/>
- [Segaller 1998]** S. Segaller, *Nerds 2.0.1, A Brief History of the Internet*, TV Books, Nueva York, 1998.
- [Sekar 2011]** V. Sekar, S. Ratnasamy, M. Reiter, N. Egi, G. Shi, “The Middlebox Manifesto: Enabling Innovation in Middlebox Deployment”, *Proc. 10th ACM Workshop on Hot Topics in Networks (HotNets)*, Article 21, 6 pages.
- [Serpanos 2011]** D. Serpanos, T. Wolf, *Architecture of Network Systems*, Morgan Kaufmann Publishers, 2011.
- [Shacham 1990]** N. Shacham, P. McKenney, “Packet Recovery in High-Speed Networks Using Coding and Buffer Management”, *Proc. 1990 IEEE INFOCOM* (San Francisco, CA, abril 1990), págs. 124–131.
- [Shaikh 2001]** A. Shaikh, R. Tewari, M. Agrawal, “On the Effectiveness of DNS-based Server Selection”, *Proc. 2001 IEEE INFOCOM*.
- [Singh 1999]** S. Singh, *The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*, Doubleday Press, 1999.

[Singh 2015] A. Singh, J. Ong., Agarwal, G. Anderson, A. Armistead, R. Banno, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, A. Vahdat, “Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google’s Datacenter Network”, Sigcomm, 2015.

[SIP Software 2016] H. Schulzrinne Software Package site, <http://www.cs.columbia.edu/IRT/software>

[Skoudis 2004] E. Skoudis, L. Zeltser, *Malware: Fighting Malicious Code*, Prentice Hall, 2004.

[Skoudis 2006] E. Skoudis, T. Liston, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses (2nd Edition)*, Prentice Hall, 2006.

[Smith 2009] J. Smith, “Fighting Physics: A Tough Battle”, *Communications of the ACM*, Vol. 52, N° 7 (julio 2009), págs. 60–65.

[Snort 2012] Sourcefire Inc., Snort homepage, <http://www.snort.org/>

[Solensky 1996] F. Solensky, “IPv4 Address Lifetime Expectations”, in *IPng: Internet Protocol Next Generation* (S. Bradner, A. Mankin, ed.), Addison-Wesley, Reading, MA, 1996.

[Spragins 1991] J. D. Spragins, *Telecommunications Protocols and Design*, Addison-Wesley, Reading, MA, 1991.

[Srikant 2004] R. Srikant, *The Mathematics of Internet Congestion Control*, Birkhauser, 2004

[Steinder 2002] M. Steinder, A. Sethi, “Increasing Robustness of Fault Localization Through Analysis of Lost, Spurious, and Positive Symptoms”, *Proc. 2002 IEEE INFOCOM*.

[Stevens 1990] W. R. Stevens, *Unix Network Programming*, Prentice-Hall, Englewood Cliffs, NJ.

[Stevens 1994] W. R. Stevens, *TCP/IP Illustrated, Vol. 1: The Protocols*, Addison-Wesley, Reading, MA, 1994.

[Stevens 1997] W.R. Stevens, *Unix Network Programming, Volume 1: Networking APIs-Sockets and XTI*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ, 1997.

[Stewart 1999] J. Stewart, *BGP4: Interdomain Routing in the Internet*, Addison-Wesley, 1999.

[Stone 1998] J. Stone, M. Greenwald, C. Partridge, J. Hughes, “Performance of Checksums and CRC’s Over Real Data”, *IEEE/ACM Transactions on Networking*, Vol. 6, N° 5 (octubre 1998), págs. 529–543.

[Stone 2000] J. Stone, C. Partridge, “When Reality and the Checksum Disagree”, *Proc. 2000 ACM SIGCOMM* (Stockholm, Sweden, agosto 2000).

[Strayer 1992] W. T. Strayer, B. Dempsey, A. Weaver, *XTP: The Xpress Transfer Protocol*, Addison-Wesley, Reading, MA, 1992.

[Stubblefield 2002] A. Stubblefield, J. Ioannidis, A. Rubin, “Using the Fluhrer, Mantin, and Shamir Attack to Break WEP”, *Proceedings of 2002 Network and Distributed Systems Security Symposium* (2002), págs. 17–22.

[Subramanian 2000] M. Subramanian, *Network Management: Principles and Practice*, Addison-Wesley, Reading, MA, 2000.

[Subramanian 2002] L. Subramanian, S. Agarwal, J. Rexford, R. Katz, “Characterizing the Internet Hierarchy from Multiple Vantage Points”, *Proc. 2002 IEEE INFOCOM*.

[Sundaresan 2006] K. Sundaresan, K. Papagiannaki, “The Need for Cross-layer Information in Access Point Selection”, *Proc. 2006 ACM Internet Measurement Conference* (Rio De Janeiro, octubre 2006).

- [Suh 2006]** K. Suh, D. R. Figueiredo, J. Kurose y D. Towsley, “Characterizing and Detecting Relayed Traffic: A Case Study Using Skype”, *Proc. 2006 IEEE INFOCOM* (Barcelona, España, abril 2006).
- [Sunshine 1978]** C. Sunshine, Y. Dalal, “Connection Management in Transport Protocols”, *Computer Networks*, North-Holland, Amsterdam, 1978.
- [Tariq 2008]** M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, M. Ammar, “Answering What-If Deployment and Configuration Questions with WISE”, *Proc. 2008 ACM SIGCOMM* (agosto 2008).
- [TechnOnLine 2012]** TechOnLine, “Protected Wireless Networks”, online webcast tutorial, http://www.techonline.com/community/tech_topic/internet/21752
- [Teixeira 2006]** R. Teixeira, J. Rexford, “Managing Routing Disruptions in Internet Service Provider Networks”, *IEEE Communications Magazine* (marzo 2006).
- [Think 2012]** Technical History of Network Protocols, “Cyclades”, <http://www.cs.utexas.edu/users/chris/think/Cyclades/index.shtml>
- [Tian 2012]** Y. Tian, R. Dey, Y. Liu, K. W. Ross, “China’s Internet: Topology Mapping and Geolocating”, *IEEE INFOCOM Mini-Conference 2012* (Orlando, FL, 2012).
- [TLD list 2016]** TLD list maintained by Wikipedia, https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains
- [Tobagi 1990]** F. Tobagi, “Fast Packet Switch Architectures for Broadband Integrated Networks”, *Proc. 1990 IEEE INFOCOM*, Vol. 78, Nº 1 (enero 1990), págs. 133–167.
- [TOR 2016]** Tor: Anonymity Online, <http://www.torproject.org>
- [Torres 2011]** R. Torres, A. Finamore, J. R. Kim, M. M. Munafó, S. Rao, “Dissecting Video Server Selection Strategies in the YouTube CDN”, *Proc. 2011 Int. Conf. on Distributed Computing Systems*.
- [Tourrilhes 2014]** J. Tourrilhes, P. Sharma, S. Banerjee, J. Petit, “SDN and Openflow Evolution: A Standards Perspective”, *IEEE Computer Magazine*, noviembre 2014, págs. 22–29.
- [Turner 1988]** J. S. Turner, “Design of a Broadcast packet switching network”, *IEEE Transactions on Communications*, Vol. 36, Nº 6 (junio 1988), págs. 734–743.
- [Turner 2012]** B. Turner, “2G, 3G, 4G Wireless Tutorial”, <http://blogs.nmscommunications.com/communications/2008/10/2g-3g-4g-wireless-tutorial.html>
- [UPnP Forum 2016]** UPnP Forum homepage, <http://www.upnp.org/>
- [van der Berg 2008]** R. van der Berg, “How the ’Net Works: An Introduction to Peering and Transit”, <http://arstechnica.com/guides/other/peering-and-transit.ars>
- [van der Merwe 1998]** J. van der Merwe, S. Rooney, I. Leslie, S. Crosby, “The Tempest: A Practical Framework for Network Programmability”, *IEEE Network*, Vol. 12, Nº 3 (mayo 1998), págs. 20–28.
- [Varghese 1997]** G. Varghese, A. Lauck, “Hashed and Hierarchical Timing Wheels: Efficient Data Structures for Implementing a Timer Facility”, *IEEE/ACM Transactions on Networking*, Vol. 5, Nº 6 (diciembre 1997), págs. 824–834.
- [Vasudevan 2012]** S. Vasudevan, C. Diot, J. Kurose, D. Towsley, “Facilitating Access Point Selection in IEEE 802.11 Wireless Networks”, *Proc. 2005 ACM Internet Measurement Conference*, (San Francisco CA, octubre 2005).
- [Villamizar 1994]** C. Villamizar, C. Song. “High Performance TCP in ANSNET”, *ACM SIGCOMM Computer Communications Review*, Vol. 24, Nº 5 (1994), págs. 45–60.

- [Viterbi 1995] A. Viterbi, *CDMA: Principles of Spread Spectrum Communication*, Addison-Wesley, Reading, MA, 1995.
- [Vixie 2009] P. Vixie, “What DNS Is Not”, *Communications of the ACM*, Vol. 52, Nº 12 (diciembre 2009), págs. 43–47.
- [Wakeman 1992] I. Wakeman, J. Crowcroft, Z. Wang, D. Sirovica, “Layering Considered Harmful”, *IEEE Network* (enero 1992), págs. 20–24.
- [Waldrop 2007] M. Waldrop, “Data Center in a Box”, *Scientific American* (julio 2007).
- [Wang 2004] B. Wang, J. Kurose, P. Shenoy, D. Towsley, “Multimedia Streaming via TCP: An Analytic Performance Study”, *Proc. 2004 ACM Multimedia Conference* (Nueva York, NY, octubre 2004).
- [Wang 2008] B. Wang, J. Kurose, P. Shenoy, D. Towsley, “Multimedia Streaming via TCP: An Analytic Performance Study”, *ACM Transactions on Multimedia Computing Communications and Applications (TOMCCAP)*, Vol. 4, Nº 2 (abril 2008), p. 16. 1–22.
- [Wang 2010] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, M. Ryan, “c-Through: Part-time Optics in Data Centers”, *Proc. 2010 ACM SIGCOMM*.
- [Wei 2006] W. Wei, C. Zhang, H. Zang, J. Kurose, D. Towsley, “Inference and Evaluation of Split-Connection Approaches in Cellular Data Networks”, *Proc. Active and Passive Measurement Workshop* (Adelaide, Australia, marzo 2006).
- [Wei 2007] D. X. Wei, C. Jin, S. H. Low, S. Hegde, “FAST TCP: Motivation, Architecture, Algorithms, Performance”, *IEEE/ACM Transactions on Networking* (2007).
- [Weiser 1991] M. Weiser, “The Computer for the Twenty-First Century”, *Scientific American* (septiembre 1991): 94–10. <http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html>
- [White 2011] A. White, K. Snow, A. Matthews, F. Monrose, “Hookt on fon-iks: Phonotactic Reconstruction of Encrypted VoIP Conversations”, *IEEE Symposium on Security and Privacy*, Oakland, CA, 2011.
- [Wigle.net 2016] Wireless Geographic Logging Engine, <http://www.wigle.net>
- [Wiki Satellite 2016] Satellite Internet access, https://en.wikipedia.org/wiki/Satellite_Internet_access
- [Wireshark 2016] Wireshark homepage, <http://www.wireshark.org>
- [Wischik 2005] D. Wischik, N. McKeown, “Part I: Buffer Sizes for Core Routers”, *ACM SIGCOMM Computer Communications Review*, Vol. 35, Nº 3 (julio 2005).
- [Woo 1994] T. Woo, R. Bindignavle, S. Su, S. Lam, “SNP: an interface for secure network programming”, *Proc. 1994 Summer USENIX* (Boston, MA, junio 1994), págs. 45–58.
- [Wright 2015] J. Wright, *J. Wireless Security Secrets & Solutions*, 3e, “Hacking Exposed Wireless”, McGraw-Hill Education, 2015.
- [Wu 2005] J. Wu, Z. M. Mao, J. Rexford, J. Wang, “Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network”, *Proc. USENIX NSDI* (2005).
- [Xanadu 2012] Xanadu Project homepage, <http://www.xanadu.com/>
- [Xiao 2000] X. Xiao, A. Hannan, B. Bailey, L. Ni, “Traffic Engineering with MPLS in the Internet”, *IEEE Network* (marzo/abril 2000).
- [Xu 2004] L. Xu, K Harfoush, I. Rhee, “Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks”, *IEEE INFOCOM 2004*, págs. 2514–2524.

- [Yavatkar 1994]** R. Yavatkar, N. Bhagwat, “Improving End-to-End Performance of TCP over Mobile Internetworks”, *Proc. Mobile 94 Workshop on Mobile Computing Systems and Applications* (diciembre 1994).
- [YouTube 2009]** YouTube 2009, Google container data center tour, 2009.
- [YouTube 2016]** YouTube Statistics, 2016, <https://www.youtube.com/yt/press/statistics.html>
- [Yu 2004]** Yu, Fang, H. Katz, Tirunellai V. Lakshman. “Gigabit Rate Packet Pattern-Matching Using TCAM”, *Proc. 2004 Int. Conf. Network Protocols*, págs. 174–183.
- [Yu 2011]** M. Yu, J. Rexford, X. Sun, S. Rao, N. Feamster, “A Survey of VLAN Usage in Campus Networks”, *IEEE Communications Magazine*, julio 2011.
- [Zegura 1997]** E. Zegura, K. Calvert, M. Donahoo, “A Quantitative Comparison of Graph-based Models for Internet Topology”, *IEEE/ACM Transactions on Networking*, Vol. 5, Nº 6, (diciembre 1997). See also <http://www.cc.gatech.edu/projects/gtim> for a software package that generates networks with a transit-stub structure.
- [Zhang 1993]** L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, “RSVP: A New Resource Reservation Protocol”, *IEEE Network Magazine*, Vol. 7, Nº 9 (septiembre 1993), págs. 8–18.
- [Zhang 2007]** L. Zhang, “A Retrospective View of NAT”, *The IETF Journal*, Vol. 3, Issue 2 (octubre 2007).
- [Zhang 2015]** G. Zhang, W. Liu, X. Hei, W. Cheng, “Unreeling Xunlei Kankan: Understanding Hybrid CDN-P2P Video-on-Demand Streaming”, *IEEE Transactions on Multimedia*, Vol. 17, Nº 2, febrero 2015.
- [Zhang X 2102]** X. Zhang, Y. Xu, Y. Liu, Z. Guo, Y. Wang, “Profiling Skype Video Calls: Rate Control and Video Quality”, *IEEE INFOCOM* (marzo 2012).
- [Zink 2009]** M. Zink, K. Suh, Y. Gu, J. Kurose, “Characteristics of YouTube Network Traffic at a Campus Network—Measurements, Models, and Implications”, *Computer Networks*, Vol. 53, Nº 4, págs. 501–514, 2009.

Índice

- 2G, redes celulares, 457–459
3Com, 395
3DES, 498, 531
3G, 15, 430
 arquitectura de red, 459–461
 núcleo de la red, 460
 red de acceso radio, 461
 vídeo a través de, 563
3rd Generation Partnership Project (3GPP), 460, 461
4G, 430, 458
 arquitectura de red, 461–463
 núcleo de la red, 461–463
 plano de datos, 462
 QoS in, 462
 red de acceso vía radio, 463–464
 redes LAN inalámbricas frente a, 458
 tunelización, 462
4G LTE, 457, 458
- A**
A, registros, 112
AAC (*Advanced Audio Coding*), 564, 582
ABR (*Available Bit Rate*), 220
Abramson, Norman, 52, 380, 395
Acceso a Internet por cable, 12, 54
 backoff exponencial binario, 383
DOCSIS, 385–386
Acceso a Internet vía satélite, 14, 363
Acceso al enlace, 365
Acceso celular a Internet, 456–464
Acceso inalámbrico de área extensa, 15
Acceso multodoméstico, ISP de, 333
Acceso múltiple con sondeo de portadora.
 Véase CSMA
Acceso múltiple por división de código. Véase CDMA
Acceso telefónico a Internet, 8, 406
ACK (reconocimiento positivo), 172–176
 corrompido, 174
DHCP, 285, 415
 duplicado, 174, 205
 en tramas 802.11 RTC/CTS, 447–449
 recomendación de generación en TCP, 206
ACK, bit, 194
 TCP, 540–542
ACK, tramas, 447–449
ACK recibido, suceso, 201, 202
Acogida, socket de, 161
Acondicionamiento del tráfico, 598
Acuerdo de nivel de servicio (SLA), 348
Acuerdo en tres fases, proceso de, 84, 136, 192, 210–211, 417
Acuerdos bilaterales, 335
Acumulativo, reconocimiento, 183, 196
Adaptación de la velocidad, 453–454
Adaptadores, 366, 367
 802.11, 441, 444
 consulta ARP y, 391
 detección de errores, 371
 direcciones MAC, 387–389
 en tarjetas separadas, 366
 funcionamiento de CSMA/CD, 382
 independencia de las capas, 390
 LAN sobre placa base, configuración, 366
 tramas Ethernet y, 394–396
 transmisión de datagramas y, 389, 391–393
Adaptativo, backoff, 427
Adleman, Leonard, 502
Admisión de llamadas, 601
AES (*Advanced Encryption Standard*), 498, 527
Agente ajeno, 466
 ancla, 472
 descubrimiento, 472–473
Agente corresponsal, 470–471
Agente de gestión de red, 349–350
Agente de usuario, 96
Agente OpenFlow (OFA), 341
Agente propio, 466
 descubrimiento, 472–473
 enrutamiento indirecto y, 468–470
 registro ante el, 473, 474–476
Agentes software inteligentes, 68
Agregación de direcciones, 282
Agregación de rutas, 282
AH (*Authentication Header*), protocolo, 530
AIMD (*Additive-Increase, Multiplicative-Decrease*), 227–228
 equidad de, 230–232
Aislamiento del tráfico, 403, 594
Akamai, 124, 128
Algoritmo de cifrado, 494

- Algoritmo de control de congestión de TCP, 223–227
Algoritmo de descifrado, 495
Algoritmo de hash seguro (SHA-1), 507, 508
Algoritmo de Prim, 314
Algoritmos de búsqueda, 264
Algoritmos de enrutamiento
 centralizado, 313
 descentralizado, 313
 dinámico, 313
 estático, 313
 no sensibles a la carga, 314
 sensibles a la carga, 313
Algoritmos de enrutamiento de estado de los enlaces (algoritmos LS), 313, 314–317, 319
 centralizados, 314
 complejidad de cálculo, 315
 complejidad del mensaje, 323
 DV comparado con, 322–323
 oscilaciones en, 316–317
 OSPF, 324
 pasos, 314–315
 robustez, 323
 tablas de reenvío, 315–316
 velocidad de convergencia, 323
Algoritmos de enrutamiento por vector de distancias (algoritmos DV), 317–323
 cambios en el coste de los enlaces y fallo de los enlaces, 321–322
 complejidad del mensaje, 323
 descentralizado, 319
 inversa envenenada, 322
 LS comparado con, 322–323
 robustez, 323
 velocidad de convergencia, 323
Algoritmos de planificación, 463
Alias
 de host, 105, 112
 del servidor de correo, 106
Almacenamiento en buffer en el cliente, 566
Almacenamiento en caché, 250
 DNS, 111–112
 push caching, 129
 web, 91–96
ALOHA, protocolos, 377, 395
 backoff en, 427
 puro, 379
ALOHA con particiones, protocolo
 algoritmo de backoff en, 427
colisiones, 377
eficiencia, 378–379
retransmisión, 377
ALOHANet, 52, 380
Alta tasa de bits, 562
Alto, computadoras, 395
Amazon, 54, 413, 561
 fluxos de vídeo, 121, 564
 Netflix y, 127–129
 servicios de la nube, 127
Ámbito con direcciones privadas, 287
Ancho de banda, 24
 adaptación de la calidad del flujo de vídeo
 a, 565
 aislamiento del tráfico y, 594–596
 almacenamiento en caché web, 91–94
 aplicaciones P2P y, 116
 aplicaciones sensibles al, 76
 ATM, 259
 control de congestión y, 223
 cuello de botella, 92
 equidad y, 230–232
 fluxos de vídeo HTTP y, 122–123
 garantías de QoS, 590, 603
 host-a-host, 412, 413
 memoria, 265
 mínimo garantizado, 258
 precarga de vídeo, 569
 propiedades de conmutación de la capa de enlace, 400
 propiedades del vídeo, 562–563
 servicio de mejor esfuerzo y, 258
 Skype y, 581
 tasa de transferencia y, 76
 TCP en rutas con un alto, 229
 terminación anticipada del vídeo, 571
 UDP, fluxos, 566, 567–568
Andreessen, Marc, 53, 151–152
Andreessen Horowitz, 151
Android, dispositivos, 15
Anonimato, 545
Antena direccional, 449
Anuncio de agente, 473, 475
Anycast, dirección, 290
AON (*Active Optical Network*), 13
AP (*Access Point*). Véase Punto de acceso
Apache, servidor web, 130, 163
API (*Application Programming Interface*), 74
Aplicaciones. Véase también Aplicaciones de red; Multimedia, aplicaciones

- distribuidas, 5
- elásticas, 76
- P2P, 116–121
- retardos de las, 37
- SDN de control de red, 338–340
- sensibles a los retardos, 565
- sensibles al ancho de banda, 76
- servicios de transporte disponibles para las, 75–77
- tolerantes a pérdidas, 76, 565
- Aplicaciones de conversaciones en tiempo real.
 - Véase también* Voz sobre IP
 - protocolos para, 582–589
 - RTP, 582–584
 - SIP, 584–589, 614
- Aplicaciones de red, 69
 - arquitecturas, 70–73
 - basadas en estándares, 130
 - comunicación de las, 71
 - principios, 70–81
 - proprietarias, 131
 - requisitos, 77
 - servicios de transporte disponibles, 75–77
- AQM (*Active Queue Management*), 269
- Árbol de recubrimiento, 402
- Archivo de manifiesto, 123
- Área personal, redes de, 454–456
- ARP (*Address Resolution Protocol*), 387, 389–391, 415
 - consulta, 416
 - diseño de redes para centros de datos, 411
 - paquete, 390
 - respuesta, 416
 - tabla, 390
- ARPA (*Advanced Research Projects Agency*), 51, 307, 427
- ARPAnet, 192
 - algoritmos de enrutamiento, 313, 319
 - Cerf, 307
 - conexión de ALOHAnet a, 380
 - desarrollo, 52–53
 - Lam, Simon, 427
 - Metcalfe y, 395
 - Satellite System de, 427
- ARQ (*Automatic Repeat reQuest*), protocolos, 172, 370
- Arquitectura cliente-servidor, 71, 72
 - distribución de archivos, 116–121
 - programación de sockets con TCP, 135
 - programación de sockets con UDP, 132
- Arquitectura de la aplicación, 70
- Arquitectura de las redes celulares, 457–459
 - 4G, 461–464
 - 3G, 459–461
 - 2G, 457–459
- Arquitectura en capas, 40–44
 - encapsulación, 45–46
- Arquitecturas jerárquicas
 - base de datos distribuida DNS, 107–110
 - dentro de sistemas autónomos, 326
 - en redes para centros de datos, 411–412
 - pares Skype, 581
- Arranque lento, 223–224
- AS (*Autonomous Systems*). *Véase* Sistemas autónomos
- ASN (*Autonomous System Number*). *Véase* Número de sistema autónomo
- Asfixia, paquetes de, 221
- Asociación
 - de seguridad (SA), 530–531, 534
 - en arquitectura 802.11, 442
- AS-PATH, 329, 331
- Ataque de denegación de servicio. *Véase* DoS
- Ataque de solo texto cifrado, 496
- Ataque de texto en claro conocido, 496
- Ataque de texto en claro seleccionado, 496
- Ataque de vulnerabilidad, 47
- Ataque por fuerza bruta, 495–496
- Ataque por inundación SYN, 213
- Ataque por reproducción, 517
- Ataque por reproducción de la conexión, 527
- Ataque por reproducción de segmentos, 527
- Atheros AR5006, 366
- ATM, 428
 - cabeceras MPLS, 407
 - control de congestión, 220
 - Ethernet y, 393
 - información QoS, 603
 - interconexión de dispositivo IP con, 407
 - Q2931b y, 603
 - retardo y ancho de banda garantizados, 259
 - SDN y, 343
- AT&T, 4, 311, 360, 613
- Atraer a los ISP, 124
- Atributos BGP, 329
- Audio
 - AAC, 564, 582
 - adaptación de la calidad de Skype, 580
 - cargas útiles soportadas por RTP, 584
 - cuantización, 563

- Audio (*cont.*)
 eliminación de fluctuaciones, 574–577
 fluxos de, 564–565
 MP3, 564, 582
 propiedades, 563–564
- Autenticación
 802.11i, 537–539
 código de autenticación de mensajes, 508–509, 524–526, 532
 del emisor, 519, 521
 del receptor, 519
 en OSPF, 325
 LAN inalámbrica, 443–444
 MD5, 326
 punto terminal, 49–50, 492
 simple, 325
- Auto-aprendizaje, 399, 415
- Auto-escalabilidad, 73
- Autonomía administrativa, 324
- Auto-replicante, malware, 47
- Autoridad de certificación (CA), 512–513, 522
- Auto-sincronización, 317
- Auto-temporizado, 222
- Azure, 55
- B**
- B4, 311, 341, 344
- backoff
 adaptativo, 427
 aleatorio, 445
 en ALOHA con particiones, 427
 exponencial binario, 383
- Baliza, tramas, 443, 454
- Banda ancha, Internet de, 54
- Baran, Paul, 50
- Base de datos de asociaciones de seguridad (SAD), 531
- Base de datos de políticas de seguridad (SPD), 533
- Base de información de gestión (MIB), 349, 350
- Base de información de red (NIB), 341
- BBN, 52
- Bellman-Ford, ecuación, 317–318
- Bellovin, Steven, 558–559
- BER (*Bit Error Rate*), 435–437
 adaptación de la velocidad, 453
- Berners-Lee, Tim, 53
- BGP (*Border Gateway Protocol*), 310, 314, 319, 326–336, 416
- algoritmo de selección de ruta, 331
 anuncio de la información de ruta, 327–329
 atributos, 329
 destino externo al sistema autónomo, 330
 determinación de las mejores rutas, 329–332
 enrutamiento de la patata caliente, 330–331
 externa (eBGP), conexión, 328
 implementación IP-anycast, 332–333
 interna, conexión, 328–329
 papel de, 327
 política de enrutamiento, 333
 ruta, 329
 tablas de enrutamiento, 331
- BIND (*Berkeley Internet Name Domain*), 105
- Bit de paridad, 369
- bind(), 160
- BITNET, 52
- BitTorrent, 119–121, 130, 582
 algoritmo de intercambio, 120
 trackers, 119–121
- Blade, servidor, 409
- Bloque, cifrados de, 497–499
- Bloqueo de cabeza, 268
- Bloqueo, paquetes de, 221
- Bluetooth, 454–455
- Boggs, David, 393, 397
- Botnet (red robot), 47
- Brooks, Fred, 558
- BS (*Base Station*). Véase Estación base
- BSC (*Base Station Controller*), 458
- BSS (*Basic Service Set*), 440, 451
 movilidad a través de, 452–453
- BSS (*Base Station Subsystem*), 458
- BTS (*Base Transceiver Station*), 457
- Bucle de enrutamiento, 322
- Buffer
 almacenamiento en, 566
 de emisión, 192
 de la aplicación cliente, 569–570
 de recepción, 207, 208
 de salida, 20
 en fluxos, 569–570
 tamaño de, en routers, 270
 TCP, 569–570
- Bush, Vannevar, 53
- Buzón de correo, 96
- C**
- CA (*Certification Authority*). Véase Autoridad de certificación

- Cabeceras
 - IPv4, 275–276
 - MIME, 522
 - MPLS, 407–408
 - navegadores web y, 88–89
 - RTP, 584
- Cabeceras de paquetes
 - enrutamiento y, 256, 257
 - MPLS, 407–408
- Cable coaxial, 17
- Cable de cobre de par trenzado, 16–17
- Calidad de servicio (QoS)
 - admisión de llamadas, 601
 - en 4G, 462
 - garantías por conexión, 590, 601–604
 - reserva de recursos, 603
 - RTP y, 583
 - vigilancia del tráfico, 594
- Campo de carga útil, 45
 - en tramas 802.11, 449
- Canales
 - 802.11, 442–444
 - con errores de bit, 172–178
 - con pérdidas, 176–178
 - de acceso múltiple, 373, 374
 - de difusión, 373
 - de radio vía satélite, 18
 - de radio terrestres, 17–18
 - totalmente fiables, 171
- Canónico, nombre de host, 106
- Capa de abstracción de servicios (SAL), 344
- Capa de aplicación, 42
- Capa de aplicación, protocolos de la, 79
 - definición, 79–81
 - DNS, 43
 - FTP, 42
 - HTTP, 42, 80
 - Skype, 79
 - SMTP, 42, 80
- Capa de comunicaciones, SDN, 338
- Capa de conectores seguros. *Véase* SSL
- Capa de enlace, 43–44, 364–368
 - difusión, 373
 - direcciónamiento, 387–389
 - la red como una, 406–409
 - servicios proporcionados por, 364–366
 - tamaño del datagrama IP y, 276
 - ubicaciones de implementación, 366–367
- Capa de gestión del estado de la red,
 - SDN, 339
- Capa de red, 42. *Véase también* Plano de control; Plano de datos
 - datagrama de, 43
 - reenvío y enrutamiento, 254–258
 - relaciones entre la capa de transporte y la, 154–156
 - seguridad, 258, 528–535
 - servicio de mejor esfuerzo, 258
 - servicios, 258–259
- Capa de transporte, 43
 - en Internet, 156–158
 - garantías de temporización, 76
 - reensamblado de fragmentos y, 277
 - relaciones entre la capa de red y la, 154–156
 - seguridad, 77
 - servicios para aplicaciones, 75–77
 - tasa de transferencia, 76
 - transferencia de datos fiable y, 75
- Capa física, 44
- Capacidad del enlace
 - congestión de la red y, 215
 - tamaño de buffer y, 270
- Capas de protocolos, 41–42
- Caracteres comodín en las entradas de la tabla de flujo, 296
- Carga de tráfico
 - buffers y, 270
 - colas y, 267
- Carga ofrecida, 216
- Carga útil de seguridad para encapsulación (ESP), 530, 532–533
- CAST, 522
- CBC (*Cipher Block Chaining*). *Véase* Encadenamiento de bloques cifrados
- CD (*Compact Disk*). *Véase* Disco compacto
- CDMA (*Code Division Multiple Access*), 376, 430, 437–440
- CDN (*Content Distribution Network*). *Véase* Red de distribución de contenido
- Celdas, 457
- Centro de conmutación móvil. *Véase* MSC
- Centro de conmutación pasarela para servicios móviles. *Véase* GMSC
- Centro de Coordinación CERT, 493
- Centro de operaciones de red (NOC), 348
- Centros de datos, 8, 72
 - CDN, 124
 - Google, 125
 - hosts, 409
 - modulares, 412–413

- Centros de datos en la nube, 410
 - arquitectura jerárquica, 411–412
- Cerf, Vinton, 52, 192, 307–308, 427
- Certificación de clave pública, 512–514, 522
- Certificado, 513
- César, cifrado de, 495
- Check Point, 540, 546
- China Telecom, 311
- China Unicom, 311
- CIDR (*Classless Interdomain Routing*), 281, 415
- Cifrado, 492
 - asociaciones de seguridad y, 530
 - contraseñas, 516–517
 - de César, 495
 - de clave simétrica, 495–500
 - estándares, 498
 - monoalfabético, 495
 - polialfabético, 496–497
 - tipos de ataques contra el, 496
- Cifrado de clave pública, 495, 500–505
 - en PGP, 522
 - en SSL, 527
- Círculo, 23
- Cisco, 3, 54, 540, 546
- Cisco 12000, serie, 266
- Cisco Catalyst 6500, serie, 264
 - comutación vía bus, 265
- Cisco Catalyst 7600, serie, 264
- Cisco Catalyst 8500, serie, 265
- Cisco CRS, estrategia de comutación, 266
- Clark, Jim, 53, 151
- Clases de tráfico, 594
 - aislamiento, 594–595
- Clasificación de paquetes, 598
- Clave, 495
 - de autenticación, 508
 - de sesión, 503–504, 520
 - gestión de claves IPsec, 534–535
 - maestra (MS), 527
 - maestra de par (PKM), 538
 - privada, 501
 - pública, 501
 - SSL, 524
 - temporal (TK), 539
- Cliente, 8, 27, 71
 - procesos, 73–74, 192
- CMTS (*Cable Modem Termination System*), 12, 385–386
- CNAME, registros, 112
- Codificación de audio avanzada (AAC), 564, 582
- Código de autenticación de mensajes (MAC), 508–509
 - datagramas IPsec, 532
 - en SSL, 524–526
 - firmas digitales y, 510–512
- Códigos polinómicos, 371
- Colas
 - carga de tráfico y, 267
 - con conservación del trabajo, 272
 - con prioridad, 271–272
 - con prioridad sin desalojo, 272
 - de entrada, 267
 - de mensajes, 96
 - de salida, 20, 268–270
 - en routers, 267–270
 - FIFO, 270–271
 - round-robin, 270, 272–273
 - velocidad de línea, 267
 - velocidad de transmisión y, 266–267
 - WFQ, 272–273
- Colisiones, 374
 - ALOHA con particiones, 377
 - CSMA, 380–382
 - CSMA/CD, 446
 - comutación de la capa de enlace que elimina, 400
 - en canales de difusión, 374
 - FDM, 376
 - protocolos de acceso aleatorio y, 377
 - TDM, eliminación de, 375
- Complejidad de cálculo del algoritmo LS, 315
- Comportamiento por salto (PHB), 599, 601
- Compresión de vídeo, 562
- Comprobación de errores, suma de
 - comprobación de UDP y, 167–168
- Comprobación de redundancia cíclica (CRC), 371–373, 435
 - en 802.11, 445, 449
 - en tramas Ethernet, 395
- Comprobaciones de paridad, 369–370
- Computación en la nube, 55
 - Amazon, 127–129
- Comunicación lógica, 154
- Comunicación segura, 492
- Concentrador (*hub*), 393
- Conexión BGP, 328
- Conexión extremo a extremo, 23
- Conexión TCP, 78

- Conexiones no persistentes, 83–85
- Conexiones TCP en paralelo, equidad y, 232
- Confidencialidad, 492, 519
- Congestión
 - algoritmos alternativos, 228
 - causas y costes, 214–219
 - desbordamientos de buffer, 218–219
 - retardos, 216
 - retransmisión y, 216–218
 - routers y, 215–219
 - rutas con múltiples saltos y, 218–219
 - segmentos perdidos, 222
 - tasa de transferencia y, 215–219
- Conjunto de servicio básico. *Véase BSS (Base Service Set)*
- Comutación, 259
 - en reenvío basado en el destino, 263
 - técnicas de, 264–266
 - vía bus, 265
- Comutación de circuitos, 23–27
 - comutación de paquetes y, 25–27
- Comutación de etiquetas multiprotocolo.
 - Véase MPLS*
- Comutación de etiquetas, router de, 407
- Comutación de paquetes, 21–22, 23, 67
 - almacenamiento y reenvío, 20–21
 - comutación de circuitos y, 23–24
 - desarrollo de, 50–51
- Comutador (*switch*), 393
- Comutador de malla, 265–266
- Comutador de paquetes, 3, 19, 259. *Véase también* Switches
 - comutación de malla, 265–266
 - no bloqueante, 266
- Consulta
 - ARP, 390, 416
 - cadena de consultas DNS, 110
 - DNS, mensaje de, 416
- Contenedores, 412–413
- Contraseñas, 516–517
- Control de congestión, 157, 207
 - adaptativo, 166
 - AIMD, 227–228
 - ancho de banda y, 223
 - asistido por la red, 220, 221
 - métodos, 220
 - principios, 214–221
 - TCP, 221–233
 - terminal a terminal, 220
- Control de flujo, TCP, 207–209
- Control de tasa compatible con TCP, protocolo de (TFRC), 235
- Control de trama, campos, 451
- Control del enlace de datos de alto nivel (HDLC), 373
- Control lógicamente centralizado, 310–311
- Control por router, 310
- Controlador de la estación base. *Véase BSC*
- Controlador de red vía radio (RNC), 461, 462
- Controlador OpenFlow (OFC), 341
- Controlador SDN, 338–340
- Controladores de enrutamiento lógicamente centralizados, 257–258
 - SDN y, 257
- Conversaciones de dos, 250
- Conversaciones de voz y vídeo, 565–566
- Cookies, 89–91
 - SYN, 213
- Corrección de errores, 366
 - de nivel de bit, 367
 - hacia adelante. *Véase FEC*
 - técnicas para la, 367–373
- Correo electrónico, 96–104
 - basado en web, 104
 - formatos de los mensajes de, 100
 - IMAP, 101, 103–104
 - protocolos de acceso, 101–104
 - seguro, 518–523
 - servidores, 96–97, 106
 - SMTP, 42, 80, 97–99
- Correspondencia-acción, 264
 - en el reenvío generalizado, 294–295
 - OpenFlow, 297–299
 - tabla de reenvío, 295
 - tablas de flujo, 343
- Corresponsal, 466
 - enrutamiento indirecto, 468–470
- Cortafuegos, 289, 294, 492, 539–546
 - filtros con memoria del estado, 540, 542–543
 - filtros de paquetes tradicionales, 540–542
 - pasarelas de aplicación, 540, 544–546
- CRC (*Cyclic Redundancy Check*). *Véase* Comprobación de redundancia cíclica
- Crecimiento aditivo y decrecimiento multiplicativo. *Véase AIMD*
- Criptografía
 - componentes, 494
 - de clave pública, 494, 500–505, 522, 527
 - principios, 494–505
 - tipos de ataques, 496

- Criptografía de clave simétrica, 495–500
cifrado polialfabético, 496–497
cifrados de bloque, 497–499
correo electrónico seguro con, 519
encadenamiento de bloques cifrados, 499–500
fase de acuerdo SSL, 527
número distintivo y, 517
PGP, 522
- CSMA (*Carrier Sense Multiple Access*), 377, 380–382
- CSMA con detección de colisiones (CSMA/CD), 381, 382–384, 444, 446
- CSMA con evitación de colisiones (CSMA/CA), 440, 444, 446
- CSNET, 52
- CTS (*Clear to Send*), 447–449
- Cuantización, 563
- Cuello de botella, enlace, 38
equidad y, 230–231
- Cuerpo de entidad, 88
- Curiosear, 493
- Cyclades, 52
- D**
- DARPA (*Defense Advanced Research Projects Agency*), 52, 307
- DARTnet, 613
- DASH (*Dynamic Adaptive Streaming over HTTP*), 122–123, 128, 572
- DATA, tramas, 447–449
- Datagramas, 43, 157
capa de red, 45
enrutamiento indirecto de, 473
envío de datagramas fuera de la subred, 391–393
inspección de, 289
IP, 414
IPsec, 531–534
IPv4, formato, 274–276
IPv4, fragmentación, 276–277
IPv6, formato, 290–292
NAT y, 286–288
re-ensamblado, 276–277, 291
- Datos duplicados, paquetes de, 177
- Datos recibidos, suceso, 201, 202
- Davies, Donald, 50
- DCCP (*Datagram Congestion Control Protocol*), 233
- DCTCP (TCP para centro de datos), 233, 235
- DDoS (DoS distribuido), 48
en servidores DNS, 115
- DECnet, 390
- Deering, Steve, 488
- Demultiplexación, 158–164, 415
de la capa de transporte, 157
orientada a la conexión, 161–162
sin conexión, 160–161
- DES (*Data Encryption Standard*), 498, 522
- Desbordamientos de buffer, 218–219
- Descubrimiento de agentes, 473–474
- Descubrimiento de router, 473
- Destinos externos al sistema autónomo, 330
- Desvanecimiento, 437
- Detección de colisiones, 381
CSMA/CD, 381, 382–384, 446
protocolo MAC 802.1, 444–445
- Detección de errores, 172, 366
bits de paridad, 369–370
de nivel de bit, 367
suma de comprobación, 370
técnicas para la, 367–373
- DHCP (*Dynamic Host Configuration Protocol*), 284–286
mensaje ACK, 285, 415
mensaje de descubrimiento, 285
mensaje de oferta, 285
mensaje de solicitud, 414–415
NAT y, 286
nodos móviles y, 285
obtención de direcciones con, 284–286
- Día D, 292
- DIAMETER, 444, 538
- Diferenciación de servicio, 590, 598–601
- Diffie-Hellman, algoritmo de intercambio de claves de, 500
- Diffserv, 598–601
- DIFS (*Distributed Inter-frame Space*), 445
- Difusión
algoritmo de difusión de estado de los enlaces, 314, 323
ARP, mensajes, 389–391, 415
capa de enlace, 373
CSMA y, 381
CSMA/CD y, 382
CTS y RTS, tramas, 447
DHCP, solicitudes, 414–415
en ALOHA, 52, 379, 380
enlaces Ethernet, 397
envenenamiento de switch, 401

- MAC, dirección de, 389, 391
- redes LAN inalámbricas, 373
- software sniffer y, 49
- tramas de sondeo, 443
- Digital Attack Map, 47
- Dijkstra, algoritmo de, 314, 319
 - en OSPF, 324
- Dimensionamiento de la red, 590, 591–592
- Dirección cedida (COA), 467
 - descubrimiento de agentes, 473
 - enrutamiento indirecto y, 468–470
- Direccionamiento, 75
 - con clases, 281
 - de la capa de enlace, 387–393
 - IPv4, 277–286
 - movilidad y, 467–468
 - de procesos, 75
 - SIP, 587
- Direcciones. *Véase también* IP, direcciones;
 - MAC, direcciones
 - ajenas, 467
 - ámbito con direcciones privadas, 287
 - anycast, 290
 - de difusión MAC, 389, 391
 - físicas, 387
 - IP, difusión, 283, 285
 - LAN, 387
 - nodo móvil, 465
 - obtención con DHCP, 284–286
 - permanentes, 467
 - SIP, 587
- Disciplina de planificación de enlace, 596
- Disco compacto (CD), 564
- Diseño centralizado, 107
- Diseño de redes para centro de datos, 410
- Dispositivo gestionado, 348
- Dispositivos inalámbricos, Internet, 429–430
- Dispositivos intermediarios, 288, 294
- Distribución de archivos
 - cliente-servidor, 116–118
 - P2P, 116–121
- Distribución de carga, 106
- Distribuidores con buffer, 397
- División TCP, 226
- DMZ (*Demilitarized Zone*). *Véase* Zona desmilitarizada
- DNS (*Domain Name System*), 43, 104–116, 415
 - almacenamiento en caché, 111–112
 - base de datos jerárquica y distribuida, 107–110
 - cadena de consultas, 110
 - consultas iterativas, 110
 - consultas recursivas, 110
 - enrutamiento dentro del dominio al servidor, 416–417
 - funcionamiento, 106–110
 - inserción de registros en la base de datos, 114–116
 - presencia en Internet, 335–336
 - interferencia con intrusos, 493
 - IP-anycast, 332
 - mensaje de consulta, 416
 - mensaje de respuesta, 417
 - mensajes, 113, 416–417
 - redes CDN y, 124–125
 - registro de recurso, 112, 416
 - registros, 112, 114
 - servicios proporcionados por, 105–106
 - UDP, 164
 - vulnerabilidades, 115
- DNS, servidores, 105
 - autoritativos, 108, 417
 - BIND, 105
 - interacciones de, 110
 - local, 109
 - objetivos de ataques DDoS, 115
 - raíz, 108
 - TLD, 108, 109
- DOCSIS (*Data-Over-Cable Service Interface Specifications*), 385–386
 - backoff exponencial binario, 383
- DOCSIS 2.0, 12
- Dominio, enrutamiento dentro de un, 416–417
- Dos (*Denial-of-Service*), ataques, 47–48
 - distribuidos, 48, 115
 - inundación SYN, 213
- DPI (*Deep Packet Inspection*), 294, 546
- DSL (*Digital Subscriber Line*), 8–10, 11, 54
- DSLAM (*Digital Subscriber Line Access Multiplexer*), 10
- DS-WCDMA (*Direct Sequence Wideband CDMA*), 461
- Duplicado, paquete, 174
- DV, algoritmo. *Véase* Algoritmo por vector de distancias
- DYSWIS, 613
- E**
- EAP (*Extensible Authentication Protocol*), 538
- EAPoL, 538

- Earthlink, 433
e-Bay, 54
eBGP, 328
EC2, 55
ECN (*Explicit Congestion Notification*), 232–233
ECE (*Explicit Congestion Notification Echo*), 233
Educause, 108
Eficiencia
 ALOHA con particiones, 378–379
 CSMA/CD, 384
 protocolos de acceso múltiples con particiones, 378
EIGRP, protocolo, 324
Elásticas, aplicaciones, 76
Eliminación
 OpenFlow, 297
 paquetes, estrategias de, 269
Eliminación del último, 269
Emisión, buffer de, 192
Emisores
 CRC, 371–372
 funcionamiento con un bit de paridad, 369
En vivo, flujos, 566
Encadenamiento de bloques cifrados (CBC), 499–500, 531
 en SSL, 527
Encapsulación, 45–46
 enrutamiento indirecto, 469
Enfoque descendente, 42
Enlace, 364
 de comunicaciones, 2
 de difusión, 373–374
Enlace de comunicaciones inalámbrico, 430, 434–437
 diferencias con los enlaces cableados, 434
 intensidad de señal, 434
 interferencias, 434
 potencia de transmisión, 436
 propagación multicamino, 434
 selección dinámica de técnicas de modulación, 436
 técnicas de modulación, 435–436
 velocidad de transmisión, 436
Enlace punto a punto, 373
 802.11 como, 449
eNodeB, 462
Enrutamiento, 254, 255
 de la patata caliente, 330–331
de llamadas hacia un usuario móvil, 477–478
directo hacia nodos móviles, 470–472
dentro de un sistema autónomo, 323–326, 334, 343
dentro de un dominio, 416–417
entre áreas, 326
entre dominios sin clase. Véase CIDR
entre los ISP, 326–336
hacia nodos móviles, 468–472
indirecto en IP móvil, 473
indirecto hacia nodos móviles, 468–470
intrusos y, 493
lógicamente centralizado, 257
multidifusión, 326, 488
pesos de los enlaces, 325
Enrutamiento, algoritmo de, 254, 255, 311–323
 ARPAnet, 313, 319
 centralizado, 313
 de estado de los enlaces, 313–317
 descentralizado, 313
 dinámico, 313
 en redes de interconexión, 412
 estático, 313
 por vector de distancias (DV), 317–323
 sensible a la carga, 313
 velocidad de convergencia, 323
Entidad de gestión de la movilidad (MME), 463
Entramado, 365
Entramado de conmutación, 260
 bus, 265
 colas y velocidad del, 266–267
 comutación de malla, 265–266
 memoria, 264–265
 red de interconexión, 266
Entrega de paquetes en orden, 258
Entrega fiable, capa de enlace, 366
Entrega garantizada, 258
Entrega garantizada con retardo limitado, 258
Envenenamiento de switch, 401
EPC (*Enhanced Packet Core*), 462
Equidad
 AIMD, 230–232
 conexiones TCP en paralelo y, 232
 TCP y, 230–232
 UDP y, 231–232
Equilibrado de carga, dispositivos de, 294, 410–411
Errores de bit
 no detectados, 368

- transferencia de datos sobre canal con, 171–176
 transferencia de datos sobre canal con pérdidas y, 176–178
ESP (Encapsulation Security Payload), 530, 532–533
 Espaciado corto entre tramas (SIFS), 445
 Espacio distribuido entre tramas (DIFS), 445
 Espacios inteligentes, 68
 Especificación de interfaz de servicio de datos por cable. Véase DOCSIS
 Espectro, derechos de acceso, 433
 Espectro disperso por salto de frecuencia (FHSS), 454–455
 Espectro sin licencia, 433
 Esquema de paridad impar, 369
 Esquema de paridad par, 369
 Esquemas de recuperación frente a pérdidas FEC, 577–578
 intercalado, 578
 ocultación de errores, 579
 Estación base (BS), 431, 440
 transferencia de llamadas, 478–480
 Estación transductora base. Véase BTS
 Estado de la conexión, 165
 Estado de puerto, mensaje, 341
 Estado del usuario, cookies, 89–91
 Estados TCP, 210–212
 Estándar avanzado de cifrado (AES), 498, 527
 Estándar de cifrado de datos (DES), 498, 522
 Estrategias de eliminación de paquetes, 269
 Estrategias de marcado de paquetes, 269
 Estrategias de selección de clústeres, 126
 Estrin, Deborah, 488–489
 Estructura de información de gestión (SMI), 349
 Ethane, proyecto, 343
Ethernet, 5, 14–15, 366, 380, 393
 backoff exponencial binario, 383
 CSMA, 377
 desarrollo de, 52
 difusión, 373
 estructura de la trama, 394–396, 405
 Gigabit, 397
 MTU, 193
 protocolo MAC, 396
 repetidor, 396
 sniffer (husmeador de paquetes), 49
 tecnologías, 396–398
 topologías, 393
 trama VLAN 802.1Q con etiquetado, 405
 tramas, 415
 velocidades, 396–397
Etiqueta VLAN, 405
Evitación de colisiones, 444
 tramas RTS/CTS, 447–449
Evitación de la congestión, 224–225
Evolución a largo plazo. Véase LTE
EWMA (Exponential Weighted Moving Average), 198
Exploración activa, 443
Exploración de puertos, 163
Exploración pasiva, 443
Extremo a extremo, retardo, 35–36, 573
- F**
- Facebook, 562
Fallo de los enlaces
 algoritmo DV y, 321–322
 rutas de reserva precalculadas, 409
FCFS (First-Come-First-Served), 270
FDDI (Fiber Distributed Data Interface), 385, 393
FDM (Frequency-Division Multiplexing), 24–25, 375, 458
 en DOCSIS, 385
 evitación de las colisiones, 376
 ortogonal, 463
 particionamiento del canal, 378
FDMA, 457
FEC (Forward Error Correction), 370, 573
 recuperación frente a pérdidas de paquetes, 577–578
 Skype, 580
Feynman, Richard, 251
FHSS (Frequency-Hopping Spread Spectrum), 454–455
Fibra hasta el hogar (FTTH), 13
Fibra óptica, 54
 en sistemas por cable, 12–13
 medios físicos, 17
FIFO (First-In-First-Out), 270–271
Filtrado, 398
Filtros de paquetes con memoria del estado, 540, 542–543
Filtros de paquetes tradicionales, 540–542
FIN, bit, 194
Fin de temporización, intervalos de duplicación, 203–204
TCP, 200, 203–204

- Firmas digitales, 505, 509–514
- Floyd, Sally, 361
- Fluctuaciones (*jitter*)
 - de los paquetes, 574
 - eliminación, 574–577
- Flujo, 290
- Flujos (*streaming*)
 - buffers TCP, 569–570
 - cifrados de flujo, 497, 536
 - DASH, 122–123, 128, 572
 - de audio/vídeo almacenado, 564–565
 - de vídeo, 121–122, 125–130
 - de vídeo P2P, 565
 - dinámicos adaptativos sobre HTTP. *Véase DASH*
 - en vivo, 566
 - HTTP, 122–123, 566, 568–569
 - HTTP adaptativos, 566
 - Netflix, plataforma, 127–129
 - P2P, 130
 - procesamiento del contenido, 128
 - redes CDN y, 125
 - RTSP, 568
 - UDP, 566, 567–568
- Fragmentación
 - datagrama IPv4, 276–277
 - datagrama IPv6, 291
- Fragmentos, 277
 - BitTorrent, 119–121
 - Netflix, plataforma de flujos de vídeo, 128
- Frame Relay, 407
- FSM (*Finite-State Machine*). *Véase Máquina de estados finitos*
- FSM ampliada, 182
- FTP, protocolo, 42
- FTTH (*Fiber To The Home*), 13
- Full-duplex, servicio, 191
- Función de medida, 600
- Funciones de control de la red en SDN, 337
- Funciones hash criptográficas, 506–507
- G**
 - Garantía estricta, 590
 - Garantía parcial, 590
 - GBN, protocolo, 181–185
 - sucesos, 183
 - TCP, 206
 - Generador, 371
 - Geográficamente más próximo, 126
 - Gestión de la conexión TCP, 209–214
- Gestión de la potencia, 454
- Gestión de red, 348–352
 - comutación de la capa de enlace, 400
 - definición, 348
 - intrusos y, 493
 - marco conceptual, 348–350
- GET, solicitudes, 86, 417
 - condicional, 95–96
 - DASH y, 122–123
 - flujos de vídeo HTTP y, 122
- GGSN (*Gateway GPRS Support Node*), 460–461
- Gigabit Ethernet, 397
- GIST, 613
- GMSC (*Gateway Mobile services Switching Center*), 476
- Google, 8, 54, 235
 - Application Engine de, 55
 - CDN, 129
 - centro de datos, 125
 - correo electrónico web, 104
 - infraestructura de red, 125
 - red privada, 29, 55, 311
 - uso de SDN, 311, 341, 344
- Google Chrome, navegador, 130
 - QUIC, protocolo, 165, 167, 235
- Google Chromium, 235
- Google Talk, 561, 565, 582
- GPRS (*Generalized Packet Radio Service*), 460–461
- Grafo, 312
 - algoritmos de grafos, 314
- Grupo de trabajo de redes móviles ad hoc del IETF, 465
- GSM (*Global System for Mobile Communications*), 457
 - aspectos comunes entre IP móvil y, 481
 - estándares 2G, 457–458
 - transferencia de llamadas, 478–480
- Gusano, 47, 163
- H**
 - H.263, 582
 - Handley, Mark, 488
 - Hash, funciones
 - criptográficas, 506–507
 - firmas digitales y, 510
 - MD5, 507, 508
 - SHA-1, 507
 - sumas de comprobación y, 506–507

- HDLC (*High-Level Data Link Control*), 373
 HELLO, mensaje, 325
 HFC (*Hybrid Fiber Coax*), 12, 363
 Híbrido de fibra y coaxial, cable (HFC), 12, 363
 Hipervínculo, 84
 HLR (*Home Location Register*), 476
 HMAC, 508
 HOL, bloqueo, 268
 Hosts, 2, 8
 alias de host, 105
 centro de datos, 410
 inalámbricos, 430
 Hotmail, 104
 HSPA (*High Speed Packet Access*), 461
 HSS (*Home Subscriber Server*), 463
 HTML, desarrollo de, 53
 HTML, archivo base, 82
 HTTP (*HyperText Transfer Protocol*), 42, 53,
 80, 81–83
 cabecera de rango de bytes de, 571
 conexiones no persistentes, 83–85
 conexiones persistentes, 83, 85
 cookies, 89–91
 formato de los mensajes, 86–89
 GET, mensaje, 417
 ICMP y, 346
 mensajes de respuesta, 87–89, 417
 mensajes de solicitud, 86–87
 puertos, 163–164
 SMTP, comparación con, 100
 solicitud, 415
 HTTP, flujos de vídeo, 122–123, 566
 adaptativos, 566
 buffer de la aplicación cliente, 569–570
 buffers TCP, 569–570
 DASH, 122–123, 128, 572
 precarga de vídeo, 569
 reposición del vídeo, 571–572
 terminación anticipada, 571–572
 YouTube, 129
 HughesNet, 14
 Hulu, 564
 Husmeador de paquetes (*sniffer*), 49, 65
- I**
- IANA, 290
 iBGP (BGP interna), 328–329
 IBM, 52
 ICANN (*Internet Corporation for Assigned Names and Numbers*), 114, 284, 324
 ICMP (*Internet Control Message Protocol*),
 346–348
 filtrado de paquetes, 540
 IPv6 e, 348
 mensajes, 115, 347
 IDEA, 522
 Identificador de conjunto de servicio. *Véase*
 SSID
 Identificador del origen de sincronización
 (SSRC), 584
 IDS (*Intrusion Detection System*), 289, 492,
 546–549
 IEEE 802.1Q, 405, 406
 IEEE 802.3 (Ethernet), 397
 IEEE 802.3z (Gigabit Ethernet), 397
 IEEE 802.5, 385
 IEEE 802.11, 4, 430
 acceso público, 54, 433
 adaptación de la velocidad, 453–454
 adaptadores, 441, 445
 arquitectura, 440–444
 autenticación, 443–444
 canales y asociación, 442–444
 como enlace punto a punto, 449
 conjunto de servicio básico (BSS), 440,
 451, 452–453
 CRC, 445, 449
 detección de colisiones, 444–445
 estándares, 440
 gestión de la potencia, 454
 MAC, direcciones, 387
 MAC, protocolo, 444–449
 movilidad dentro de la misma subred IP,
 452–453
 puntos de acceso, 440–441
 rangos de frecuencia, 440
 redes ad hoc, 441
 RTS/CTS, tramas de control, 447–449
 terminales ocultos 447–449
 transferencia en subredes, 452–453
 transmisión de tramas, 445
 IEEE 802.11, trama, 449
 campos de carga útil y CRC, 449
 campos de dirección, 449–451
 direcciones MAC, 449–451
 número de secuencia, duración y control de
 trama, campos, 451
 IEEE 802.11ac, 440
 IEEE 802.11b, interferencias de otros
 dispositivos, 434

- IEEE 802.11g, 440
IEEE 802.11i, 537–539
IEEE 802.11n, 440
IEEE 802.15.1, 454–455
IEEE 802.15.4, 455–456
IEEE 802.16, 464
IEEE 802 LAN/MAN, comité de estándares, 4
IETF (*Internet Engineering Task Force*), 4, 289, 513, 613
IKE (*Internet Key Exchange*), 534
IKE SA, 534
IMAP (*Internet Mail Access Protocol*), 101, 103–104
Indicador, campo, 194
Índice de parámetro de seguridad (SPI), 531
Información de rutas BGP, anuncio de la, 327–329
Infraestructura de red, redes inalámbricas, 431–434
Ingeniería de tráfico, 325
 MPLS, 409
Ingesta de contenidos, 127
Inspección profunda de paquetes. *Véase* DPI
Instantánea, tasa de transferencia, 37
Integridad de los mensajes, 492, 505, 519, 521
Intel, 395
Intel, adaptador 710, 366
Intensidad de la señal, 434
 desvanecimiento, 437
Intensidad de tráfico, 33
Intercalado, 578–579
Intercambio de claves de Internet (IKE), 534
Interconexión de sistemas abiertos (OSI), 44
Interfaz, 278
 controlador SDN, 338–339
 de datos distribuida para datos. *Véase* FDDI
 de programación de aplicaciones (API), 74
 de sockets, 5, 74
 NIC, 366
Interferencia, 434
Internet. *Véase también* Redes de acceso
 acceso doméstico, 8–14
 acceso empresarial, 14–15
 algoritmos de enrutamiento, 313
 auto-sincronización del router, 316–317
 capa de red, 259
 capa de transporte, 156–158
 Cerf, 307–308
 comercialización de, 53
 componentes de, 2–5
 de banda ancha, 54
 DNS y presencia en, 335–336
 estándares de, 4
 frontera de la red, 8
 historia, 50–55
 infraestructura de los servicios, 5–6
 núcleo de la red, 18
 obtención de presencia en, 335–336
 protocolos de transporte usados por
 aplicaciones de, 166
 registros regionales de, 284
 servicio de mejor esfuerzo, 258
 servicios no proporcionados por, 79
 servicios de transporte proporcionados por,
 77–79
 suma de comprobación de, 371, 506–507
 topología, 393
Internet de las cosas (IoT), 10, 489
Internet Real-Time Lab, 613
Internet Systems Consortium, 286
Interno, router, 327
Interredes, 51–52
Introducción profunda, 124, 125
Intserv, 259
Inundación de conexiones, 48
Inundación del ancho de banda, 48, 115
IoT. *Véase* Internet de las cosas
IP (*Internet Protocol*), 4, 43, 307, 428.
 Véase también IPv4; IPv6
 datagrama, 414
 ICMP e, 346
 modelo de servicio, 157
 pila para, 42
 servicio de mejor esfuerzo, 572–573
 tabla de reenvío, 415
 tráfico total anual que usa, 3
 transición a, 52
IP, direcciones, 53, 75, 105
 clases de, 281
 DHCP, 284–286
 difusión, 283, 285
 direcciones MAC y, 388
 IPv4, 277–286
 IPv6, 290
 NAT y, 286–288
 obtención de bloques de, 283
 presencia en Internet, 335
 programación de sockets, 132
 SIP y, 585–587
 temporales, 284

- IP-anycast, 332–333
 IP, fragmentación
 IPv4, 275–276
 IPv6, 291
 IP móvil, 472–476
 aspectos comunes entre GSM y, 481
 descubrimiento de agentes, 473–474
 registro ante el agente propio, 474–476
 iPhones, 15
 IPS (*Intrusion Prevention System*), 289, 546–549
 IPsec, 512, 528, 529–530
 AH y ESP, 530
 asociaciones de seguridad, 530–531, 534
 datagrama, 531–534
 gestión de claves, 534
 paquete, 531–532
 IPv4
 direcciónamiento, 277–286
 formato del datagrama, 274–276
 fragmentación del datagrama, 276–277
 transición a IPv6 desde, 292–293
 IPv6, 289
 adopción de, 292–293
 formato del datagrama, 290–292
 ICMP, 348
 transición a, 292–293
 tunelización, 292–293
 IPX, 319, 390
 IS-IS, 324, 341, 416
 ISO (*International Organization for Standardization*), 44
 ISO IDRIP, 319
 ISP (*Internet Service Provider*), 4
 acceso, 27
 acuerdos bilaterales entre los, 335
 AS, configuraciones, 324
 CDN e, 125
 de acceso multidomiciliado, 333
 de nivel 1, 28
 enrutamiento entre los, 326–336
 global de tránsito, 27
 infraestructura de Google, 125, 129
 multidomiciliación, 28
 infraestructura de Netflix en, 128
 PoP, 28
 red troncal, 333
 regional, 28
 Iterativas, consultas, 110
 ITU (*International Telecommunication Union*), 513
 IV. Véase Vector de inicialización
 IXP (*Internet Exchange Point*), 28–29
 infraestructura de red de Google, 125, 129
 infraestructura de Netflix en, 128
 redes CDN, 124
J
 Jacobson, Van, 250–251, 488
 Java, programación cliente-servidor con, 131
 Jet Propulsion Laboratory, 308
 Juniper MX2020, 260
 Juniper Networks Contrail, 344
K
 Kahn, Robert, 52, 192, 308, 427
 creación de TCP/IP y, 192
 desarrollo de ARPAnet, 52
 Kankan, 129–130, 561
 Karels, Mike, 250
 Kleinrock, Leonard, 51, 67–68, 307, 427
L
 Laboratorios NPL, 50
 Lam, Simon, 427–428
 Lampson, Butler, 297
 LAN (*Local Area Network*). Véase Redes de área local
 LAN, dirección, 387
 LAN inalámbricas, redes, 14, 363
 4G y, 458
 autenticación, 443–444
 CDMA en, 437
 de infraestructura, 441
 difusión, 373
 seguridad, 535–539
 LAN sobre placa base, configuración, 366
 LEO, satélites, 18
 Licklider, J. C. R., 50
 Limelight, 124
 Límite de saltos, 291
 Línea de abonado digital (DSL), 8–10, 11, 54
 Línea de estado, 88
 Línea de solicitud, 86
 Líneas de cabecera, 86, 88
 Listas de control de acceso, 542, 543
 Local, servidor DNS, 109
 Longitud de cabecera, campo, 194
 LoST, 613

- LTE (*Long-Term Evolution*), 15, 457, 458
 - arquitectura de red, 461–463
 - particiones de tiempo, 463–464
- LTE-Advanced, 463
- M**
- MAC, direcciones
 - adaptadores de red, 387–389
 - ARP y, 389–391
 - autenticación de redes LAN inalámbricas, 443
 - dirección de difusión, 389, 391
 - puntos de acceso en 802.11, 440, 443
 - tramas 802.11, 449–451
 - tramas baliza, 443
- MAC, protocolo. Véase Protocolo de control de acceso al medio
- Malware, 47
 - auto-replicante, 47
- MANET (*Mobile Ad hoc NETworks*), 434, 465
- MAP, mensaje, 385
- Máquina de estados finitos (FSM)
 - ampliada, 182
 - control de congestión de TCP, 224, 225
 - para el protocolo GBN, 182–184
 - para transferencia de datos sobre canal con errores de bit, 171–176
 - para transferencia de datos sobre canal con pérdidas y errores de bit, 176–178
 - para transferencia de datos sobre canal totalmente fiable, 170, 171
- Marca de tiempo, 574, 584
- Marcado de los paquetes, 594
- Marcos temporales, 375
- Máscara de subred, 279
- MD4, 507
- MD5, algoritmo hash, 507, 508
- MD5, autenticación, 326
- MDC (*Modular Data Center*), 412–413
- Mecanismo de goteo, 596–598
- Media móvil exponencialmente ponderada (EWMA), 198
- Medidas en tiempo real, 127
- Medios compartidos, 17
 - retardos, 37
- Medios de difusión, 250
 - flujos en vivo, 566
- Medios físicos, 16–18
 - cable coaxial, 17
 - cable de cobre de par trenzado, 16–17
- canales de radio terrestre, 17–18
- canales de radio vía satélite, 18
- fibra óptica, 17
- Medios guiados, 16
- Mejor esfuerzo, servicio de, 258
 - dimensionamiento, 590, 591–592
 - limitaciones, 572–574
 - provisión de múltiples clases de servicio, 592–598
- Memoria
 - ancho de banda, 263
 - comutación vía, 264–265
 - ternaria direccionable por contenido (TCAM), 264
 - tiempo de acceso, 264
- Mensajes, 18, 43, 73
 - ARP, 389–391, 415
 - complejidad en los algoritmos LS, 322–323
 - de la capa de aplicación, 45
 - de respuesta HTTP, 87–89
 - DHCP, 285, 414–415
 - DNS, 113–114, 415–416
 - formatos de los mensajes de correo electrónico, 100
 - formato para HTTP, 86–89
 - HELLO, 325
 - ICMP, 347
 - intrusos y, 493
 - OpenFlow, 342
 - ping ICMP, 115
 - port-status*, 342
 - regulación del origen, 346–347
 - SIP, 587
- Metcalfe, Bob, 380, 395
- MIB (*Management Information Base*), 349, 350
- Microsoft, 54
 - red privada, 55
- Microsoft Research, 311
- MIME, cabecera, 522
- MIMO (*Multiple Input Multiple-Output*), 440
- Minitel, 53
- MME (*Mobility Management Entity*), 463
- Modelo de servicio, 41
 - de red, 258–259
 - IP, 157
 - transferencia de datos fiable, 169
- Modificación de campos, acción, 297
- Modo de infraestructura, 432
- Modo transporte, 532

- Modo túnel, 531
 Modulación por código de pulsos. *Véase* PCM
 Monoalfabético, cifrado, 495
 Mosaic Communications, 53
 MOSPF, 326
 Movilidad, 429–430, 489
 dentro de la misma subred IP, 452–453
 dirección del nodo y, 465
 direcccionamiento y, 467–468
 en redes VLAN, 453
 gestión de la, 464–472
 gestión de la movilidad en redes celulares, 476–481
 grados de, 464–465
 infraestructura cableada y, 465
 protocolos de las capas superiores y, 481–483
 transferencia (*handoff*) y, 432
 MP3, 564, 582
 MPEG, 582
 MPEG 1 capa 3 (MP3), 564, 582
 MPLS (*Multiprotocol Label Switching*), 406, 407–409
 MSC (*Mobile Switching Center*), 459, 460, 476
 ancla, 480
 propio, 476
 transferencia de llamadas, 478–480
 MSRN (*Mobile Station Roaming Number*), 477
 MSS (*Maximum Segment Size*), 192, 229
 MTU (*Maximum Transmission Unit*), 193, 276
 Multidifusión, enrutamiento, 488
 en OSPF, 326
 Multidomiciliación, 28
 Multimedia, aplicaciones, 562–614
 conversaciones de voz y vídeo sobre IP, 565–566
 fluxos de audio y vídeo almacenado, 564–565
 fluxos de audio y vídeo en vivo, 566
 propiedades del audio, 563–564
 propiedades del vídeo, 562–563
 soporte de red para, 589–604
 TCP, 166
 tipos, 564–566
 UDP, 166
 Múltiples versiones, 563
 Multiplexación, 158–164
 de la capa de transporte, 157
 orientada a la conexión, 161–162
 sin conexión, 160–161
 Multiplexación por división de frecuencia.
 Véase FDM
 Multiplexación por división de frecuencia ortogonal (OFDM), 463
 Multiplexación por división en el tiempo.
 Véase TDM
 MX, registro, 106, 112
- N**
- NAK (reconocimiento negativo), 172–176, 370
 corrompido, 174
 NAT (*Network Address Translation*), 286–288, 294
 Skype, 579–582
 NAT (*Network Address Translator*), 264
 NAT traversal, 288
 NCP (*Network-Control Protocol*), 51, 53
 NCS (*Network Control Server*), 341
 Nelson, Ted, 53
 Netflix, 121, 561, 564
 CDN, 127–129
 Netscape Communications, 53, 151, 523
 NeVoT, 613
 NEXT-HOP, 329–330, 331
 NFV (*Network Functions Virtualization*), 343
 NIB (*Network Information Base*), 341
 NIC (*Network Interface Card*), 366
 NIST, 292, 499
 Nivel de bit, detección y corrección de errores de, 367
 nmap, 163, 214
 No bloqueante, conmutador, 266
 NOC (*Network Operations Center*), 348
 Nodal, retardo, 30
 Nodo de soporte GPRS servidor (SGSN), 460–461
 Nodos, 364
 Nodos móviles
 direcccionamiento, 467–468
 DHCP y, 285–286
 enrutamiento directo hacia, 470–472
 enrutamiento hacia, 468–472
 enrutamiento indirecto hacia, 468–470
 redes ajenas y, 467–468
 Nombre de host, 104
 alias, 105, 112
 en consultas DNS, 109–110
 en registros de recursos DNS, 112
 servicios DNS y, 105–106

- Notación decimal con puntos, 278, 389–390
 Notificación explícita de congestión (ECN), 232–233
 Novell IPX, 319
 NOX, controlador, 340, 344
 NS, registros, 112
 NSFNET, 52
 nslookup, programa, 114
 NTT, 4
 Núcleo de la red, 18
 - conmutación de circuitos, 23–27
 - conmutación de paquetes, 18–22, 25–26
 - redes 3G, 460
 - redes 4G, 461–463
 - una red de redes, 27–29
 Número de itinerancia de la estación móvil (MSRN), 477
 Número de puerto de destino, campo, 159, 194
 Número de puerto de origen, campo, 159, 194
 Número de reconocimiento, campo, 194
 Número de secuencia, campo, 194
 Número de sistema autónomo (ASN), 324
 Número distintivo, 517, 535, 536
 Números de puerto, 75, 132
 - bien conocidos, 159
 - NAT y, 286–288
 - socket, 160
 Números de reconocimiento, 195–196
 - superpuesto, 197
 - Telnet y, 196–197
 Números de secuencia, 174
 - cálculo del código MAC en SSL, 526
 - control de fluctuaciones con, 574
 - protocolos con procesamiento en cadena, 180
 - protocolo GBN, 181–182
 - retransmisiones y, 174
 - RTP, 584
 - SR, protocolo, 186, 187
 - TCP, 195–197
 - Telnet y, 196–197
 - tramas 802.11, 451

O

 - Objeto, 82
 - Objetos gestionados, 349
 - OC (*Optical Carrier*), 17
 - OFA (*OpenFlow Agent*), 341
 - OFC (*Open Flow Controller*), 341

OFDM (*Orthogonal Frequency Division Multiplexing*), 463
 OLT (*Optical Line Terminator*), 13
 ONIX, controlador SDN, 341
 ONOS, 340, 344–346
 ONT (*Optical Network Terminator*), 13
 Opciones, campo, 194
 OpenDaylight, 340, 344
 OpenDaylight Lithium, 340, 344
 OpenFlow, 339, 340–341

 - acción, 297
 - correspondencia, 296–297
 - correspondencia-acción, 297–299
 - tabla de flujo, 295
 Organización de Soporte de Direcciones de ICANN, 284
 Orientada a la conexión, demultiplexación, 161–162
 Orientada a la conexión, multiplexación, 161–162
 OSI (*Open Systems Interconnection*), modelo de referencia, 42, 44
 OSPF (*Open Shortest Path First*), 310, 314, 324–326, 416

 - autenticación, 325
 - difusión, 325–326
 - Dijkstra, algoritmo de, 324
 - multidifusión (MOSPF), 326
 - pesos de los enlaces, 325
 - seguridad, 325
 - subredes, 324
 OVSDB, 344

P

 - P2P, arquitectura, 72–73, 428
 - distribución de archivos, 116–121
 - escalabilidad, 116–118
 - Skype, 579–582
 - P2P, flujos de vídeo, 130, 565
 - Packet Radio, 427
 - Packet Satellite, 427
 - Paging, 458
 - Paquetes, 2, 19
 - ARP, 390
 - de asfixia o bloqueo, 221
 - de datos duplicados, 177
 - desordenados, 184
 - duplicados, 174
 - entrega en orden, 258
 - fluctuación de, 574

- inspección profunda de, 289, 294, 546
- IPsec, 531–532
- procesamiento de, 402
- reenvío, 254
- RTP, 583
- Paquetes de control, 260
 - de Skype, 580
- Par trenzado no apantallado. *Véase* UTP
- Pares (*peers*), 28, 72–73
 - BitTorrent, 119–121
 - no filtrado de forma optimista, 120
 - no filtrados, 120
 - P2P, flujos de vídeo, 130
 - retransmisores, 580
 - Skype, 581
 - vecinos, 119
- Paridad bidimensional, 370
- Paridad de un bit, 369–370
- Particiones con éxito, 378
- Particiones de tiempo, 375
 - en LTE, 463–464
- Pasarela de aplicación, 540, 544–545
- Pasarela de la red de datos empaquetados (P-GW), 462, 463
- Pasarela de servicio (S-GW), 463
- Pasarela, router de, 327
- Patata caliente, enrutamiento de la, 330–331
- Paxos, 341
- PCM (*Pulse Code Modulation*), 564, 582
- PDU (*Protocol Data Unit*), 350, 351
- Pérdida de paquetes, 21, 33, 267
 - FEC, 577–578
 - intercalado, 578
 - ocultación de errores, 579
 - recuperarse de, 577–579
 - VoIP y, 573
- Pérdida de propagación, 434
- Pérdidas, canales con, 176–178
- Perfil de tráfico, 600–601
- Periodos de inactividad, 24
- Permanente, dirección, 467
- Persistente, conexión, 83, 85
- Pesos de los enlaces en OSPF, 325
- PGP (*Pretty Good Privacy*), 519, 522–523
- P-GW (*Packet Data Network Gateway*), 462
- PHB (*Per-Hop Behavior*), 599, 601
 - de reenvío expedito, 601
 - de reenvío garantizado, 601
- Photobell, 67
- Picored, 455
- Pila de protocolos, 42
- ping, programa, 346
- ping, mensajes, 115
- Pipelining*, 180
 - TCP, 199
- Planificación de paquetes
 - colas con prioridad, 271–272
 - FIFO, 270–271
 - round robin, 272–273
 - WFQ, 272–273
- Planificación oportunista, 463
- Planificador de paquetes, 270
- Plano de control, 253, 261, 309
 - 4G, 462
 - SDN, 336–346
- Plano de datos, 253
 - 4G, 462
 - IP, 274–293
 - reenvío generalizado y SDN, 294–299
 - routers, 260–273
 - SDN y, 337, 342–343
- Plano de reenvío, 260–261
- Plataforma de control de enrutamiento (RCP), 360
- PLMN (*Public Land Mobile Network*), 476
- Plug-and-play, 284, 400
- PMK (*Pairwise Master Key*), 538
- PMS (*Pre-Master Secret*), 527
- Polialfabético, cifrado, 496–497
- Política de enrutamiento BGP, 333–335
- PON (*Passive Optical Network*), 13–14
- PoP (*Point of Presence*), 28
- POP3 (*Post Office Protocol*—Versión 3), 102
- Portadora óptica (OC), 17
- Potencia de transmisión, 436
- Pouzin, Louis, 52
- PPP (*Point-to-Point Protocol*), 363, 373
 - MTU, 193
- Precarga de vídeo, 569
- Preferencia local, 331
- Prefijo, 263, 281
- Preparado para enviar (CTS), 447–449
- Presión inversa, 569
- Primero el menos común, 120
- Primero en entrar, primero en salir. *Véase* FIFO
- Principio terminal a terminal, 168
- Privacidad, 545
 - equivalente a la del cable (WEP), 535–537
 - VoIP y, 582

- Problema de acceso múltiple, 373
- Problema de enrutamiento triangular, 470
- Problema del terminal oculto, 436, 447–449
- Procesador de enrutamiento, 260
- Procesamiento, retardo de, 31
- Procesamiento del contenido, 128
- Procesamiento en cadena, 180
- Procesamiento en el puerto de entrada, 262–264
 - tablas de reenvío en, 263
- Procesamiento en el puerto de salida, 266
- Procesamiento nodal, retardo de, 30
- Proceso de acuerdo, 78
 - SSL, 524, 527–528
 - TCP en tres fases, 84, 136, 192, 210–211, 417
- Procesos
 - cliente, 73–74
 - comunicación entre, 73–75
 - direcciónamiento de, 75
 - interfaz entre el proceso y la red, 74
 - protocolos de la capa de transporte y, 154
 - servidor, 73–74, 192
- Programación basada en sucesos, 185
- Programación de sockets, 130–131
 - arquitectura cliente-servidor, 132
 - direcciones IP, 132
 - números de puerto, 132, 160
 - TCP, 135–139
 - UDP, 131–135
- Propagación de información, 250
- Propagación multicamino, 434
- Propagación, retardo de, 30, 31–33
- Protocolo, 5. *Véase también los protocolos específicos*
 - de acceso al medio, 363
 - de acceso aleatorio, 375, 377–384, 395, 444
 - de acceso múltiple, 374–375, 444
 - de aplicación bien conocidos, 159
 - de autenticación, 515–518
 - de enrutamiento, 22
 - de establecimiento de llamada, 603
 - de la capa de transporte, 154
 - de localización de usuarios móviles, 471
 - de parada y espera, 173, 180, 181
 - de paso de testigo, 385
 - de paso de testigo en anillo, 385
 - de red, 7–8
 - de sondeo, 384
 - de toma de turnos, 375, 384–385, 444
 - de ventana deslizante, 182
- definición, 6–8
- sin memoria del estado, 83
- Protocolo ampliable de autenticación (EAP), 538
- Protocolo de control de acceso al medio (protocolo MAC), 365
 - 802.11, 444–449
 - Ethernet, 397
- Protocolo de control de congestión de datagramas (DCCP), 233
- Protocolo de control de red (NCP), 51, 53
- Protocolo de control de transmisión. *Véase TCP*
- Protocolo de datagramas de usuario. *Véase UDP*
- Protocolo de enrutamiento entre sistemas autónomos, 326, 334
- Protocolo de flujos en tiempo real (RTSP), 568
- Protocolo de gestión de red, 350
- Protocolo de iniciación de sesión. *Véase SIP*
- Protocolo de Internet. *Véase IP*
- Protocolo de oficina de correos versión 3. *Véase POP3*
- Protocolo de resolución de direcciones. *Véase ARP*
- Protocolo de transferencia de datos fiable, 169
 - construcción, 170–178
 - con procesamiento en cadena, 178–180
- Protocolo de transferencia de hipertexto. *Véase HTTP*
- Protocolo de transmisión para control de flujos (SCTP), 235
- Protocolo de transporte en tiempo real. *Véase RTP*
- Protocolo dinámico de configuración de host. *Véase DHCP*
- Protocolo punto a punto. *Véase PPP*
- Protocolo simple de gestión de red. *Véase SNMP*
- Protocolo simple de transferencia de correo. *Véase SMTP*
- Protocolos de particionamiento del canal, 375–376, 437, 444
 - CDMA, 376
 - FDM, 376
 - TDM, 375–376
- Proveedor, 27
- Proveedor de servicios de Internet. *Véase ISP*
- Provisión de ancho de banda, 591
- Proxy, servidor, 91, 545

SIP, 588
 PSH, bit, 194
 Puerto de entrada, 260
 Puerto de salida, 260, 264
Pull, protocolo, 100
 Puntero de datos urgentes, campo, 194
 Punto a punto, conexiones, 191
 Punto de acceso (AP), 431, 433
 direcciones MAC, 441, 443
 en redes LAN inalámbricas de
 infraestructura, 441
 exploración, 443
 gestión de la potencia, 454
 movilidad entre, 464–465
 Punto de intercambio de Internet. *Véase* IXP
 Punto de presencia (PoP), 28
Push caching, 129
Push, protocolo, 100
 Python, 131
 conexiones TCP, 137–139
 conexiones UDP, 133–135, 160
 números de puerto, 160

Q

Q2931b, protocolo, 603
 QoS. *Véase* Calidad de servicio
 QQ, 565
 QUIC, protocolo, 165, 167, 235

R

RADIUS, 444, 538–539
 Raíz, servidores DNS, 108, 111
 Rand Institute, 50
 RC4, cifrado de flujo, 536
 RCP (*Routing Control Platform*), 360
 Realimentación del receptor, 172
 Receptor
 CRC, 371
 operación del receptor con un bit de
 paridad, 369–370
 Reconocimientos
 acumulativos, 183, 196
 de la capa de enlace, 444, 445, 446
 negativos, 172–176, 370
 positivos, 172–176, 205, 285, 447–449
 superpuesto, 197
 TCP, 195–197, 207
 Recuperación rápida, 226–227
 Recursivas, consultas, 110
 Red, adaptadores de. *Véase* Adaptadores

Red de acceso vía radio
 4G, 463–464
 3G, 461
 Red de confianza, 523
 Red de distribución de contenido (CDN), 94,
 123–126
 casos de estudio, 127–130
 centros de datos, 124
 comercial, 124
 DNS y, 124–125
 estrategias de selección de clústeres, 126
 flujos de vídeo y, 125
 funcionamiento, 124–126
 Google, 129
 IP-anycast, 332
 ISP y, 124
 IXP y, 124
 Kankan, 129–130
 Netflix, 127–129
 privada, 124, 127–129
 YouTube, 129
 Red de interconexión, 410
 algoritmos de enrutamiento, 413
 comutación vía, 266
 Red de redes, 27–29, 53
 Red definida por software. *Véase* SDN
 Red móvil terrestre pública propia (*home*
 PMLN), 476
 Red óptica activa (AON), 13
 Red óptica pasiva (PON), 13–14
 RED (*Random Early Detection*), 269
 Redes. *Véase también* Internet; Redes
 de acceso; Redes de área local;
 Redes celulares; Redes inalámbricas
 ad hoc, 441
 ajenas, 466–468
 ataques a las, 46–50
 CDN, 94, 123–130
 celulares, 15, 433, 437, 456–464, 476–481
 de acceso vía radio, 461, 463–464
 de área personal, 454–456
 de circuitos virtuales, 407
 de comutación de paquetes vía radio, 52
 de comutación de paquetes vía satélite, 52
 de malla inalámbrica, 434
 de proveedores de contenidos, 29
 frontera de la red, 8, 10, 467
 móviles ad hoc (MANET), 434, 465
 para centro de datos, 409–413
 PLMN, 476

- Redes (cont.)
 - privadas, 29, 55, 287, 311, 529
 - privadas virtuales (VPN), 409, 529–530
 - programables, 337
 - proliferación de, 52–53
 - propias, 466, 476
 - red móvil terrestre pública propia, 476
 - seguridad, 492–495
 - sociales, 54
 - soporte de red para aplicaciones multimedia, 589–604
 - tasa de transferencia en, 37–39
 - topología de red conmutada, 402
 - visitadas, 466, 476
 - VLAN, 403–406, 411, 453
 - VPN, 409, 529–530
 - WPAN, 454–455
- Redes celulares
 - 2G, 457–459
 - 3G, 15, 430, 433, 459–461, 563
 - 4G, 430, 457, 458, 461–464
 - gestión de la movilidad en, 476–481
 - GSM, 457–459, 478–481
 - LTE, 15, 457, 458, 461–464
 - uso de CDMA, 376
- Redes de acceso, 8–15, 333
 - 3G, 15
 - cable, 12, 54, 383, 385–386
 - DSL, 8–9, 54
 - empresariales, 14–15
 - Ethernet, 14–15
 - FTTH, 13
 - HFC, 12
 - LTE, 15, 457, 458, 461–464
 - radio, 461–464
 - satélite, 14, 363
 - telefónico, 8, 14, 406
 - WiFi, 14–15
- Redes de área local (LAN), 14–15. *Véase también* LAN inalámbricas, redes conmutadas, 386–406
 - husmear, 401
- Redes de área local virtuales (VLAN), 403–406
 - en redes para centros de datos, 411
 - movilidad, 453
- Redes de computadoras, 2
 - historia, 50–55
 - interfaz entre el proceso y las, 74
 - modelo de grafo, 312
- tasa de transferencia en, 37–39
- Redes inalámbricas, 429–489
 - ad hoc, 432
 - basadas en infraestructura y múltiples saltos, 433–434
 - basadas en infraestructura y un único salto, 433
 - de malla, 434
 - elementos de, 430–432
 - husmeador de paquetes (sniffer), 49
 - infraestructura de red, 431–434
 - multisalto, enlaces punto a punto en, 449
 - protocolos de las capas superiores y, 481–483
 - sin infraestructura y múltiples saltos, 434
 - sin infraestructura y un único salto, 433
 - tipos, 433–434
 - transferencia (*handoff*), 432
 - vehicular ad hoc, 434
- Redes para centro de datos, 409–413
 - arquitectura jerárquica, 411–412
 - equilibrado de carga, 410–411
 - tendencias, 412–413
- Redundancia espacial, 562
- Redundancia temporal, 562
- Re-ensamblado
 - datagrama IPv4, 276–277
 - datagrama IPv6, 291
- Reenvío (*forwarding*), 254, 256, 259
 - basado en el destino, 261, 262–264
 - basado en el flujo, 336
 - OpenFlow, 297
 - paquetes, 254
 - regla de coincidencia con el prefijo más largo, 263, 281
 - SDN, 336
 - switches de la capa de enlace, 398–399
 - trama ampliada MPLS, 407
- Reenvío generalizado, 261, 294–299
 - acción, 297
 - correspondencia, 296–297
 - correspondencia-acción, 297–299
- Registrador (entidad de registro), 114, 335
 - SIP, 588
- Registro
 - ante el agente propio, 474–476
 - en IP móvil, 475
 - SSL, 526
- Registro de ubicación de visitantes (VLR), 476
 - enrutamiento de llamadas, 477–478

- Registro de ubicaciones propias (HLR), 476
 - enrutamiento de llamadas, 477–478
- Registros de recursos (RR), 112, 416
- Regla de coincidencia con el prefijo más largo, 263, 281
- Relación señal-ruido (SNR), 435–437
 - adaptación de la velocidad, 453
- Reloj ack, 250
- Repetición selectiva (SR), 180, 186–191
 - funcionamiento, 188
 - sucesos y acciones, 187
 - tamaño de ventana, 188, 189
 - TCP como, 206
- Repetidor, 396
- Reserva de recursos, 603
- Resumen de rutas, 282
- Retardos
 - de cola, 20–21, 30, 31, 33–35
 - de cola y congestión de la red y, 216
 - de procesamiento, 31
 - de procesamiento nodal, 30
 - de propagación, 30, 31–33
 - de propagación de canal, 382
 - de reproducción adaptativo, 575–577
 - de reproducción fijo, 575
 - de transmisión, 30, 31–33
 - en aplicaciones, 37
 - en medios compartidos, 37
 - en redes de conmutación de paquetes, 30–37
 - en sistemas terminales, 37
 - extremo a extremo, 35–36, 573
 - limitado, 258
 - nodal, 30
 - tipos de, 30–33
 - total nodal, 30
- Retransmisión, 172
 - ALOHA con particiones, 377
 - basada en el tiempo, 177
 - congestión y, 216–218
 - CSMA/CA y, 445
 - CSMA/CD y, 446
 - gestión del temporizador en TCP, 200
 - intervalo de fin de temporización TCP, 200
 - números de secuencia para tratar la, 174
 - paquetes duplicados, 174
 - protocolos de acceso aleatorio, 377
 - rápida, 204–205
- Retransmisores Skype, 580–581
- Retroceder N (GBN), protocolo, 181–185
 - sucesos, 183
 - TCP como, 206
- Rexford, Jennifer, 360–361
- RFC (*Requests For Comments*), 4
 - como estándares de protocolos, 130
- RFC 2616, 130
- RIP, 314, 319, 416
- Rivest, Ron, 502, 507
- RNC (*Radio Network Controller*), 461, 462
- Roberts, Lawrence, 51, 427
- Robustez, algoritmos LS y DV, 323
- Round-robin, cola, 270, 272–273
- Routers, 3, 19, 294
 - arquitectura, 261
 - auto-sincronización, 316–317
 - colas, 266–270
 - componentes de los, 259–262
 - con funcionalidad NAT, 286–288
 - congestión y, 215–219
 - control por router, 310
 - de conmutación de etiquetas, 407
 - de frontera, 260, 326, 410
 - de pasarela, 327
 - entramado de conmutación, 264–266
 - interno, 327
 - plano de datos, 260–273
 - plano de reenvío, 260–261
 - tablas de reenvío, 256, 257
 - procesamiento en el puerto de entrada, 262–265
 - procesamiento en el puerto de salida, 266
 - reenvío basado en el destino, 261, 262–265
 - tamaño de buffer, 270
 - switches y, 401–402
- RSA, algoritmo, 502–505, 522
- RST, bit, 194
- RSVP protocol, 603
- RTP (*Real-Time Transport Protocol*), 568, 582–584
 - tipos de carga útil de audio y vídeo, 584
- cabecera, 583
 - campos de cabecera de paquetes, 583–584
 - paquete, 583
 - sesión, 583
- RTS (*Request to Send*), 447–449
- RTSP (*Real-Time Streaming Protocol*), 568
- RTT (*Round-Trip Time*), 84
 - estimación, 198–200
 - tamaño de buffer y, 270
- TCP, 221–222

- RTTEstimado, 198
- RTTMuestra, 198
- Ruta, 3
 - algoritmo de selección de ruta BGP, 331
 - BGP, 329
 - con alto ancho de banda, 229
 - con múltiples saltos, 218–219
 - de reserva, 409
 - más corta, 313
 - varias rutas de igual coste, 326
- Ruta de coste mínimo, 312
 - ecuación de Bellman-Ford para, 317–318
 - LS, algoritmo, 314–316
- S**
- SA (*Security Association*). Véase Asociación de seguridad
- SAD (*Security Association Database*), 531
- SAL (*Service Abstraction Layer*), 344
- Saltos inalámbricos, 432–434
- Satélites de la órbita baja terrestre (LEO), 18
- Satélites geoestacionarios, 18
- Scantlebury, Roger, 50
- Schulzrinne, Henning, 582, 613–614
- SCTP (*Stream Control Transmission Protocol*), 235
- SDN (*Software-Defined Networking*), 257, 258, 360, 360, 489
 - aplicaciones SDN de control de red, 338–340
 - arquitectura, 337
 - cambio en el estado de un enlace, 342–343
 - características fundamentales, 336–338
 - control lógicamente centralizado, 310–311
 - plano de control, 261, 336–346
 - plano de datos, 337, 342–343
 - reenvío de paquetes y, 259
 - reenvío generalizado y, 294–299
 - responsabilidades del procesador de enrutamiento, 260
 - tablas de reenvío, 260, 262
- Segmentos, 43, 154, 156
 - capa de transporte, 43
 - perdidos, 222
 - reconocidos, 223
- SYN TCP, 417, 540
- tamaño máximo, 192, 193
- TCP, 193
- TCP, estructura, 193–197
- UDP, 414
 - UDP, estructura, 167
- Seguridad, 491–559
 - ataque por inundación SYN, 213
 - capa de red, 258, 528–535
 - correo electrónico, 518–523
 - cortafuegos, 289, 294, 492, 539–546
 - envenenamiento de switch, 401
 - IDS, 289, 492, 546–549
 - inspección de datagramas, 289
 - OSPF y, 325
 - protocolo de transporte, 77
 - redes LAN inalámbricas, 535–539
 - vulnerabilidades DNS, 115
- Seguridad de la capa de transporte (TLS), 523
- Seguridad operacional, 492, 539–549
 - cortafuegos, 539–546
 - IDS, 289, 492, 546–549
- Selectivo, reconocimiento, 207
- Semáforos, puntos de acceso inalámbrico, 433
- Señales baliza, 479
- Señalización del establecimiento de llamada, 603
- Servicio de control de flujo, 207
- Servicio de entrega de mejor esfuerzo, 157
- Servicio de red, aplicaciones de, 344
- Servicio de transferencia de datos fiable, 200
- Servicio diferenciado, 590
 - Diffserv, 598–601
- Servicios
 - capa de enlace, 364–366
 - capa de red, 258–259
 - capa de transporte, 75–77
 - de control de flujo, 207
 - de Información de GE, 52
 - DNS, 105–106
 - en capas, 41
 - en la nube, 226
 - full-duplex, 191
 - no fiables, 157
 - TCP, 78, 156
 - UDP, 79
- Servicios de transporte
 - disponibilidad, 75–77
 - proporcionados por Internet, 77–79
 - requisitos de aplicaciones de red, 77
- Servidor de abonado doméstico (HSS), 463
- Servidores, 8, 71, 73
 - de control de red (NCS), 341
 - de correo, 96–97, 106
 - de gestión, 348

- DNS, 105, 108–110, 115
- DNS autoritativos, 108, 417
- DNS local, 109
- DNS raíz, 108, 111
- interacción del usuario mediante cookies, 89–91
- introducción profunda, 124
- procesos servidor, 73–74, 192
- proxy, 91, 545, 588
- TCP, programación de sockets con, 139
- UDP, programación de sockets con, 134–135
- web, 53, 82, 163–164
- S-GW (*Service Gateway*), 463
- SGSN (*Serving GPRS Support Node*), 460–461
- SHA-1 (algoritmo de hash seguro), 507, 508
- Shamir, Adi, 502
- SIFS (*Short Inter-frame Spacing*), 445
- Sin conexión, demultiplexación, 160–161
- Sin conexión, multiplexación, 160–161
- SIP (*Session Initiation Protocol*), 585–589, 614
 - direcciones, 587
 - llamada a una dirección IP conocida, 585–586
 - mensajes, 587
 - proxy, 588
 - traducción de nombres y localización de usuarios, 587–589
- Sistema basado en anomalías, 547
- Sistema basado en firma, 547, 548
- Sistema de detección de intrusiones. Véase IDS
- Sistema de nombres de dominio. Véase DNS
- Sistema de prevención de intrusiones. Véase IPS
- Sistema de terminación de módem por cable. Véase CMTS
- Sistema global de comunicaciones móviles. Véase GSM
- Sistemas autónomos (AS), 324
 - anuncio de rutas BGP, 327–329
 - conexiones iBGP, 328
 - enrutamiento dentro de, 323–326, 334
 - enrutamiento entre, 334, 344
 - jerarquía, 326
- Sistemas terminales, 2, 4, 8, 9
 - retardo en, 37
- Skype, 561, 565, 579–582
 - calidad de audio y vídeo, 580
 - jerarquía de pares, 581
- paquetes de control, 580
- pares retransmisores, 580–582
- TCP y, 580
- técnicas P2P en, 580–582
- UDP y, 580
- SLA (*Service Level Agreement*), 348
- Slammer, gusano, 163
- SMI (*Structure of Management Information*), 349
- SMTP (*Simple Mail Transfer Protocol*), 42, 80, 96, 97–99
 - comparación con HTTP, 100
 - protocolos de acceso para correo electrónico, 101–102
- SNA, 52
- Sniffer (husmeador de paquetes), 49, 65
- SNMP (*Simple Network Management Protocol*), 344, 350–352
- Snort, 548
- SNR (*Signal-to-Noise Ratio*). Véase Relación señal-ruido
- Sobrecarga debida a la cabecera de los paquetes, 165
- Sockets, 158
 - de acogida, 161
 - interfaz de, 5, 74
 - números de puerto, 160
 - simultáneos, 162
 - TCP, 415, 417
- SOHO (*Small Office, Home Office*), subredes, 286
- Solicitud de agente, 474
- Solicitud de comentarios. Véase RFC
- Solicitud de conexión, 161
- Solicitud de eco, 347
- Solicitud de transmisión (RTS), 447–449
- Solicitud HTTP, mensaje de, 86–87
- Sondear, 384
- Sondeo de portadora, 381
- Sondeo, tramas de, 443
- SPD (*Security Policy Database*), 533
- SPI (*Security Parameter Index*), 531
- Spotify, 562
- Sprint, 4
- SRI (Stanford Research Institute), 51, 67
- SSID (*Service Set Identifier*), 442
 - en tramas baliza, 443
- SSL (*Secure Sockets Layer*), 78, 151, 427, 523–528, 545
 - cierra de la conexión, 528

- SSL (*Secure Sockets Layer*) (cont.)
 - deducción de claves, 524–525
 - fase de acuerdo, 524, 527–528
 - transferencia de datos, 525–526
- SSRC (*Synchronization Source Identifier*), 584
- Stanford Research Institute (SRI), 51, 67
- StarBand, 14
- Subredes, 278–282
 - movilidad, 452–453
 - obtención de un bloque de direcciones IP, 283
 - en OSPF, 324
 - SOHO, 286
 - transmisión de datagramas, 391–393
- Subsistema de estaciones base (BSS), 458
- Sucesos de fin de temporización
 - GBN, protocolo, 183
 - SR, protocolo, 187
 - TCP, 200, 201, 202
- Suma de comprobación, campo, 194
- Sumas de comprobación (*checksums*)
 - cabeceras IPv4, 275–276
 - de Internet, 371, 506–507
 - detección de errores con, 370
 - detección de paquetes ACK y NAK
 - corruptos, 174
 - funciones hash y, 506–507
 - UDP, 167–168
- Superpares, 581
- Suplantación IP, 49–50, 516
- SWAN, 311
- Switch, 393
- Switches de la capa de enlace, 3, 19, 259, 398–403
 - auto-aprendizaje, 399
 - búsqueda de la dirección de destino, 264
 - propiedades, 400
 - redes VLAN y, 403
 - reenvío y filtrado, 398–399
 - TOR, 409
- SYN, bit, 194
- SYN, cookies, 213
- SYN TCP, segmento, 417, 540
- SYNACK segmento, 209, 213
- T**
- Tabla de conexiones, 543
- Tabla de commutación, 398
 - envenenamiento, 401
- Tabla de flujo, 295
- caracteres comodín, 296
- correspondencia-acción, 343
- SDN, 339
- Tabla de traducciones NAT, 287
- Tablas de enrutamiento, 319
 - BGP, 331
- Tablas de reenvío, 22, 256, 257
 - correspondencia-acción, 295
 - en algoritmo LS, 315–316
 - en el procesamiento de entrada, 263–264
 - en SDN, 260, 262
 - IP, 415
 - prefijos, 263
 - routers, 256, 257
 - tarjetas de línea, 262
- Tamaño de ventana, 182
 - en SR, 188, 189
- Tamaño máximo de segmento (MSS), 192, 229
- Tanteo del ancho de banda, 223, 227
- Tarjeta de interfaz de red (NIC), 366
- Tarjetas de línea, 366
 - tablas de reenvío, 263
 - puertos de entrada y de salida, 260
- Tasa de errores de bit. Véase BER
- Tasa de transferencia, 37–39, 76
 - congestión y, 215–219
 - de la capa de transporte, 76
 - instantánea, 37
 - media, 37
 - por conexión, 215–216
 - TCP, 228
- TCAM (*Ternary Content Addressable Memories*), 264
- TCP (*Transmission Control Protocol*), 4, 156.
 - Véase también SSL
 - ACK bit, 540–542
 - algoritmo de control de congestión de, 223–227
 - aplicaciones multimedia que usan, 166
 - arranque lento, 223–224
 - auto-temporizado, 222
 - buffers, 569–570
 - cierre de la conexión, 211
 - conexión, 191–193
 - conexiones en paralelo con navegadores, 84–85
 - conexiones en paralelo, equidad, 232
 - control de congestión, 221–233
 - control de flujo, 207–209
 - demultiplexación, 161–162

- desarrollo, 53
- equidad y, 230–232
- establecimiento de la conexión, 209–210
- estructura del segmento, 193–197
- evitación de la congestión, 224–225
- full-duplex, servicio, 191
- gestión de la conexión, 209–214
- gestión del temporizador, 200–201
- intervalos de fin de temporización, 200, 203–204
- intervalo de fin de temporización para la retransmisión, 200
- números de reconocimiento, 195–197
- números de secuencia, 195–197
- procesamiento en cadena, 199
- proceso de acuerdo en tres fases, 84, 136, 192, 210, 417
- programación de sockets, 130, 135–139
- punto a punto, conexiones, 191
- recomendación para la generación de mensajes ACK en, 206
- reconocimiento acumulativo, 196
- reconocimiento selectivo, 207
- recuperación rápida, 226–227
- redes inalámbricas y, 481
- retransmisión rápida, 204–205
- RTT, estimación, 198–200
- rutas con un alto ancho de banda, 229
- segmentos 193
- seguro, 78
- servicios proporcionados por, 78, 157
- servidores web y, 163–164
- Skype, 580
- SMTP, 98
- socket, 415, 417
- socket cliente, 137–138
- socket servidor, 139
- sockets de conexión simultáneos, 162
- solicitud de conexión, 161
- sucesos fin de temporización, 198–200, 201, 202
- suma de comprobación de Internet, 371
- tasa de transferencia, 228
- transferencia de datos fiable, 200–207
- transición a, 53
- variables, 221–222, 224, 227
- ventana de congestión, 221, 227
- ventana de recepción, 207, 208
- TCP/IP, 4, 192
- TCP Reno, 227, 228
- TCP Tahoe, 227
- TCP Vegas, 228
- TDM (*Time-Division Multiplexing*). Véase Multiplexación por división en el tiempo
- Técnicas de conexión dividida, 482
- Técnicas de modulación
- PCM, 564, 582
- selección dinámica de, 436
- SNR y BER, 435–436
- Telefonía móvil, 15, 54, 376, 429
- Telefonía por Internet, 79, 565, 573
- Teléfonos tradicionales, 489
- Telenet, 52
- Telnet, 99, 196–197, 516, 544–545
- Temporizador de cuenta atrás, 177
- Terminación de línea óptica (OLT), 13
- Terminación de red óptica (ONT), 13
- Terminal a terminal, control de congestión, 220
- Testigo, 385
- Texto cifrado, 494
- Texto en claro, 494, 496
- Texto plano, 494, 496
- TFRC (*TCP-Friendly Rate Control*), 235
- Third Generation Partnership Program, 293
- Tiempo de arrendamiento de la dirección, 285
- Tiempo de distribución, 116–118
- Tiempo de ida y vuelta. Véase RTT
- Tiempo de respuesta, rendimiento de los servicios de la nube, 226
- Tiempo de vida (TTL), 275
- Tipo de servicio (TOS), 275, 593, 594
- TK (*Temporal Key*), 539
- TLD (*Top-Level Domain*), servidores DNS, 108, 109
- TLS (*Transport Layer Security*), 523
- Tolerantes a pérdidas, aplicaciones, 76, 565
- Tomlinson, Ray, 51
- Topología completamente conectada, 412
- TOR, 545
- TOR, switch, 409
- Tormentas de difusión, 402
- Torrente, 119–121
- Torres de telefonía, 431
- TPID (*Tag Protocol Identifier*), 405
- Traceroute, 35–36, 347
- Trackers, 119–121
- Traducción de direcciones de red. Véase NAT (*Network Address Translation*)
- Traductor de direcciones de red (NAT), 264

- Tramas, 44
 802.11, 449–451
 802.11, transmisión de, 445
 ACK, 447–449
 baliza, 443, 454
 colisiones entre, 374
 CTS, 447–449
 DATA, 447–449
 de la capa de enlace, 45, 364
 de reconocimiento, 445
 de sondeo, 443
 Ethernet, 415
 Ethernet, estructura de, 394–396
 MPLS, 407–408
 RTS, 447–449
 VLAN, 405
- Transferencia (*handoff*)
 en GSM, 478–481
 en redes inalámbricas, 432
 en subredes 802.11, 452–453
- Transferencia de datos bidireccional, 170
- Transferencia de datos fiable, 75, 157, 190–191
 implementación del servicio, 169, 170
 modelo de servicio para, 169
 principios, 169–191
 sobre canal con errores de bit, 171–176
 sobre canal con pérdidas y errores de bit, 176–178
 sobre canal totalmente fiable, 170–171
 TCP, 200–207
- Transferencia de datos unidireccional, 170
- Transmisión de almacenamiento y reenvío, 19–20
- Transmisión, retardo de, 30, 31, 32–33
- Transparente, 398
- Transporte, fiabilidad en el nivel de aplicación, 166–167
- Transporte orientado a la conexión, 191–214
- Transporte sin conexión, 164–168
- Triple-DES, 522
- Troncalización VLAN, 404, 405
- TTL (*Time-To-Live*), 275
- Túnel, 292
- Tunelización, 292
 en redes 4G, 462
- Tymnet, 52
- U**
- UCLA, 67, 307
- UDP (*User Datagram Protocol*), 156, 157, 164–168
 aplicaciones multimedia que usan, 165–166
 DNS, 164
 equidad y, 231–232
 estructura de un segmento, 167
 fiabilidad, 166–167
 fluxos, 566, 567–568
 multiplexación y demultiplexación, 160
 naturaleza sin conexión de, 164
 RTP y, 582–583
 segmento, 414
 servicios, 79
 Skype, 580
 suma de comprobación, 167–168
 suma de comprobación de Internet, 371
 ventajas, 165
 VoIP y, 573
- UDP, programación de sockets con, 131–135
- cliente, 133–134
- números de puerto, 160
- servidor, 134–135
- UmbralAL, 224–227
- UMTS (*Universal Mobile Telecommunications Service*), 459, 461
- Unidad de datos de protocolo. Véase PDU
- Unidad máxima de transmisión (MTU), 193, 276
- Unión Internacional de Telecomunicaciones (ITU), 513
- URG, bit, 194
- URL, SIP, 587
- UTP (*Unshielded Twisted Pair*), 16
- Utilización del canal, 179
- V**
- Valor de espera aleatorio, 445
- VANET (*Vehicular ad hoc network*), 434
- Vecino, 312
- Vector de inicialización (IV), 500, 527, 536–537
- Velocidad de chip, 437
- Velocidad de convergencia, algoritmo de enrutamiento, 323
- Velocidad de transmisión, 2, 216
 BER y, 436
 colas y, 266–267
- Velocidades de línea y colas, 266–267
- Ventana de congestión, 221, 227
- Ventana de recepción, 194, 207, 208

- VentCongestion, 221, 223–227
 VentRecepcion, 221–222
Verisign Global Registry Services, 108
Vídeo
 calidad con Skype, 580
 cargas útiles soportadas por RTP, 584
 por Internet, 121–122
 propiedades, 562–563
Vídeo, flujos de, 121–122
 almacenados, 564–565
 en vivo, 566
 Netflix, plataforma, 127–129
 P2P, 565
 precarga, 569
 procesamiento, 128
 redes de distribución de contenido y, 125
 repositionamiento, 571–572
Videoconferencia, 582
Vigilancia del tráfico, 594
 mecanismo de goteo, 596–598
Virtualización de enlaces, 406–409
Virtualización de funciones de red (NFV), 343
Virus, 47
VLAN (Virtual Local Area Network). Véase
 Redes de área local virtuales
VLR (Visitor Location Register). Véase
 Registro de ubicación de visitantes
VoIP. Véase Voz sobre IP
Volumen de tráfico IP, 3
Voz sobre IP (VoIP), 37, 430, 565, 572–582,
 613
 eliminación de fluctuaciones, 574–577
 fluctuación de los paquetes, 574
 limitaciones del servicio IP de entrega de
 mejor esfuerzo, 572–574
 pérdida de paquetes, 573
 problemas de privacidad, 582
 retardo extremo a extremo, 573
 RTP, 582–584
 SIP, 585–589
VPN (Virtual Private Network). Véase Redes
 privadas virtuales
- W**
- Web, navegadores, 53–54, 73–74, 82
 almacenamiento en caché web, 91–94
 conexiones en paralelo, 84–85, 232
 cookies, 89–91
 correo electrónico, 104
 GET condicional y, 95–96
 líneas de cabecera, 88–89
 solicitudes GET, 86
 SSL, 523
Web, página, 82
 solicitud de, 413–418
Web, servidores, 53, 82
 TCP y, 163–164
Wechat, 561, 582
WEP (Wired Equivalent Privacy), 535–537
WFQ (colas equitativas ponderadas), 272–273,
 597–598
WiFi, 4, 5, 366, 430, 440
 adaptación de la velocidad, 453–454
 arquitectura, 440–444
 campos de dirección, 449–451
 carga útil y CRC, campos, 449
 gestión de la potencia, 454
 husmeador de paquetes (*sniffer*), 49
 implementación de la capa de enlace, 366
 jungla, 442
 MAC, direcciones, 449–451
 MAC, protocolo, 444–449
 movilidad dentro de la misma subred IP,
 452–453
 número de secuencia, duración y control de
 trama, campos, 451
 pública, 54, 433
 redes inalámbricas de área extensa y, 15
 tramas, 449–451
 ubicuo, 433
 uso empresarial de, 14–15
**WiMAX (World Interoperability for
 Microwave Access)**, 464, 613
Wireless Philadelphia, 433
Wireshark, 49, 65–66, 403
World Wide Web, 69, 81
WPAN (Wireless Personal Area Network),
 454–456
- X**
- X.25, serie de protocolos, 53, 428
 X.509, 513
Xerox Palo Alto (Xerox PARC), centro de
 investigación, 395
XTP, 371
- Y**
- Yahoo, 54
 correo electrónico web, 104
Youku, 121, 561

YouTube, 121, 564
centros de datos, 125
redes CDN, 129

Zimmerman, Phil, 522
Zona desmilitarizada (DMZ), 547
Zonas de disponibilidad, 413

Z

zeroconf, 284
Zigbee, 455–456