

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 - Video	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 deseé. 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 commutación 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.

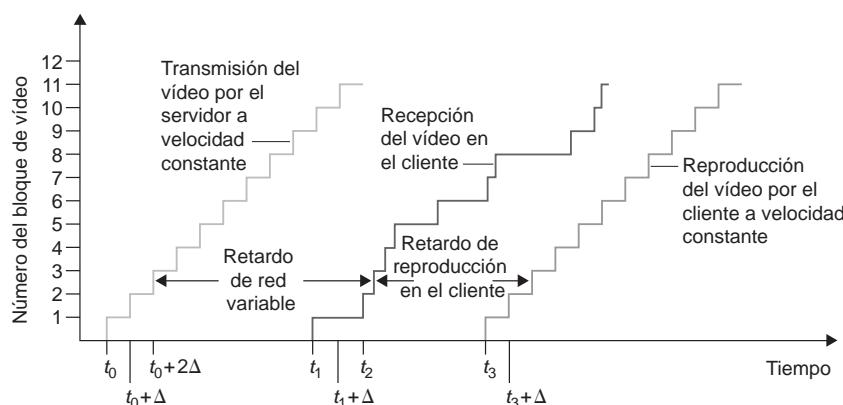


Figura 9.1 • Retardo de reproducción de un flujo de vídeo en el cliente.

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

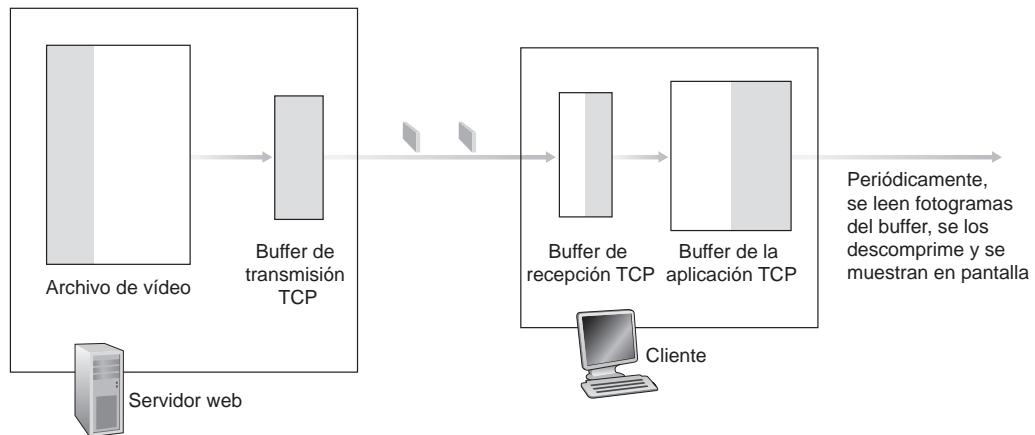


Figura 9.2 ♦ Flujo de vídeo almacenado a través de HTTP/TCP.

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 vídeo

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

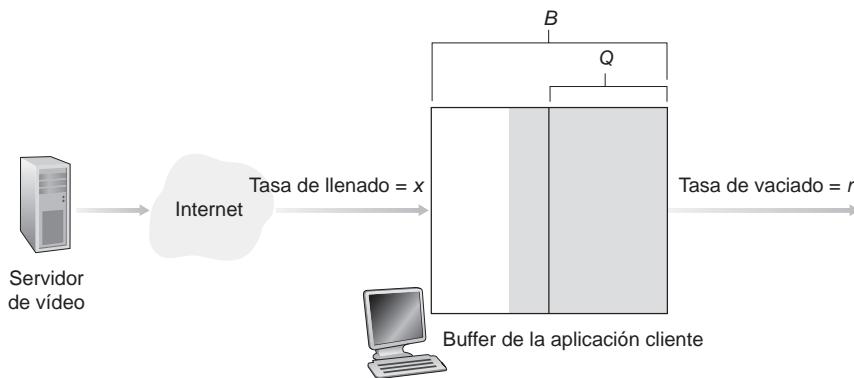


Figura 9.3 ♦ Análisis del almacenamiento en buffer en el lado del cliente durante la transmisión de un flujo de vídeo.

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_f 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 desperdiando 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 posibles, 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 síncronamente 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.

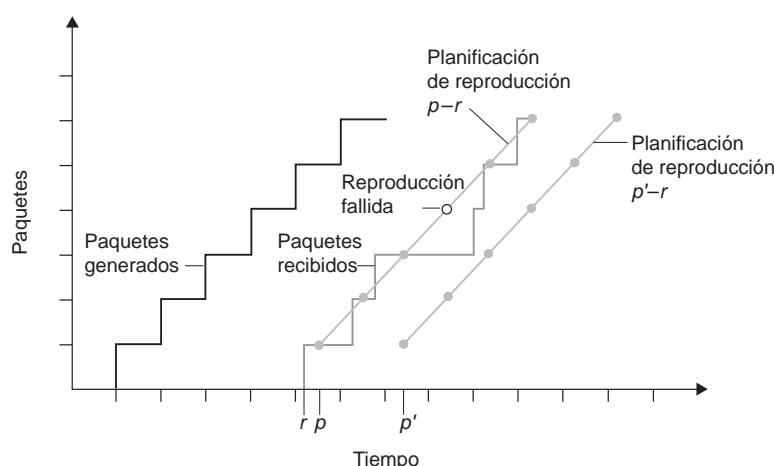


Figura 9.4 ▶ Pérdida de paquetes para diferentes retardos fijos de reproducción.

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

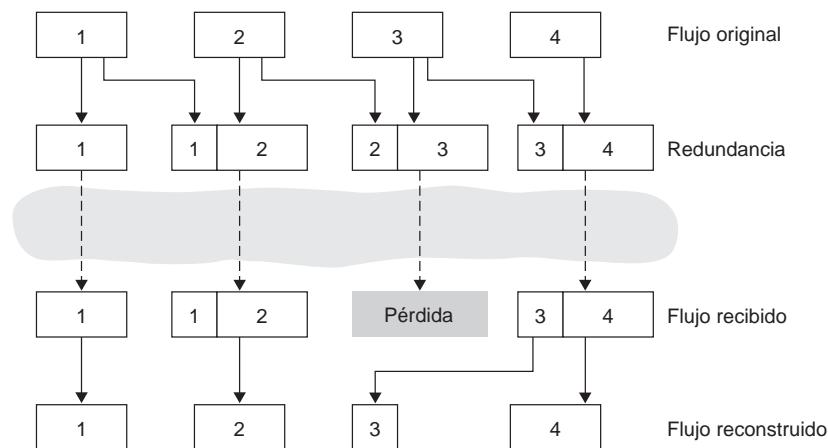


Figura 9.5 ♦ Adición de información redundante de menor calidad.

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.

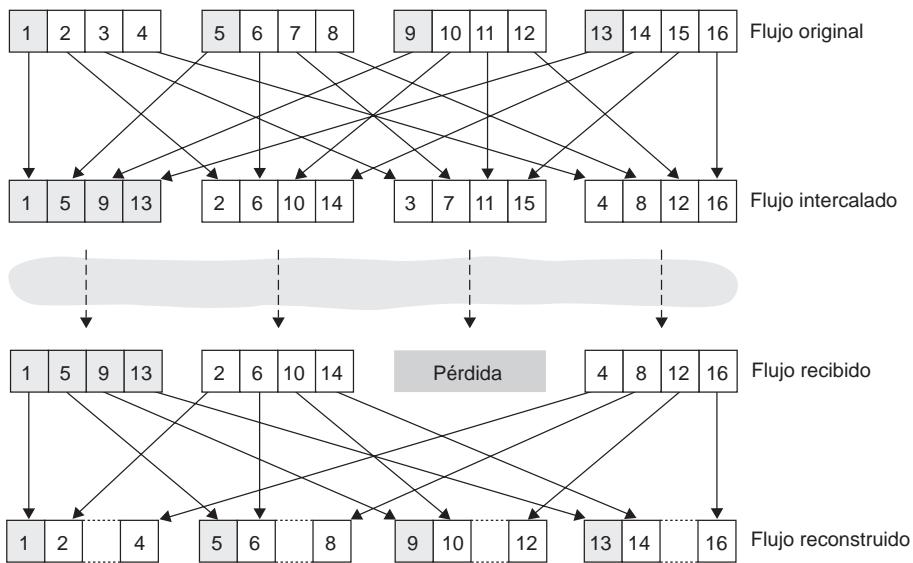


Figura 9.6 ♦ Transmisión de audio intercalado.

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 aproxima 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

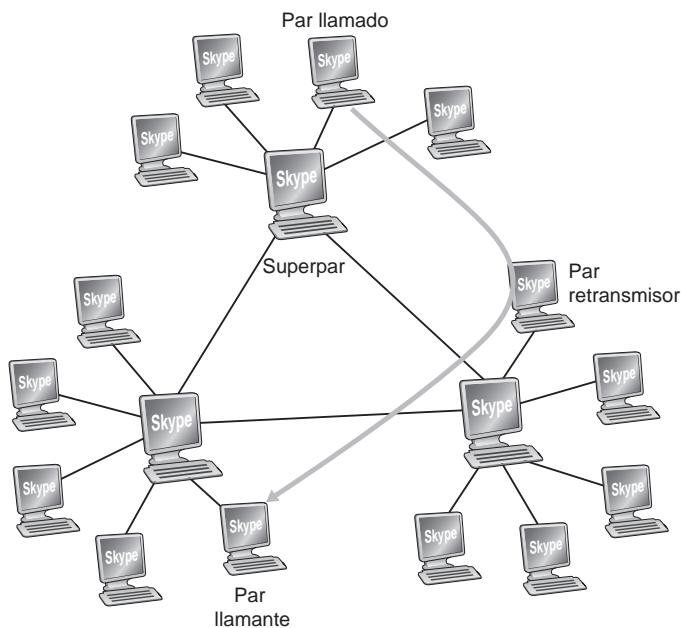


Figura 9.7 ♦ Pares Skype.

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 estudiando!) 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 pasarlo 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

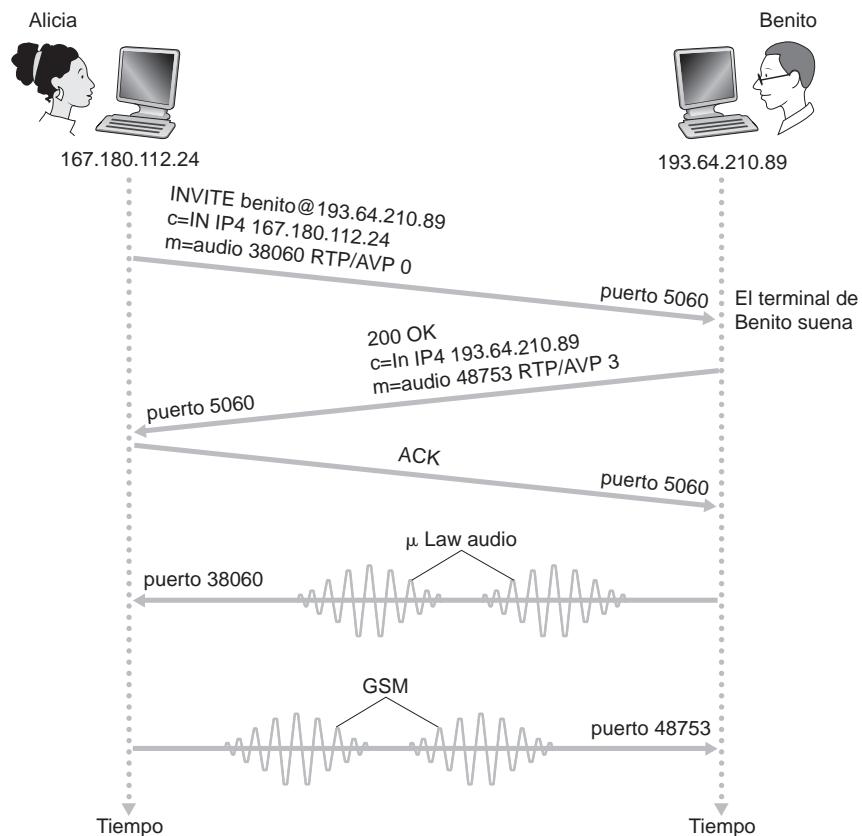


Figura 9.9 ♦ Establecimiento de una llamada SIP cuando Alicia conoce la dirección IP de Benito.

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

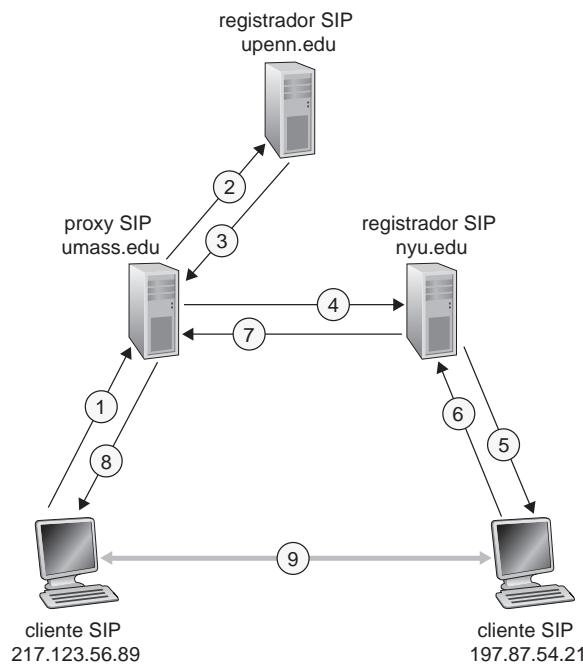


Figura 9.10 ♦ Iniciación de sesión que implica proxies y registradores SIP.

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 comutación 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 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 quizá 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

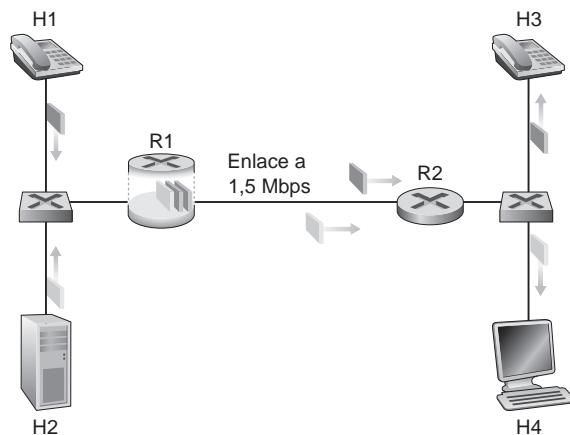


Figura 9.11 ♦ Aplicaciones de audio y HTTP compitiendo.

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

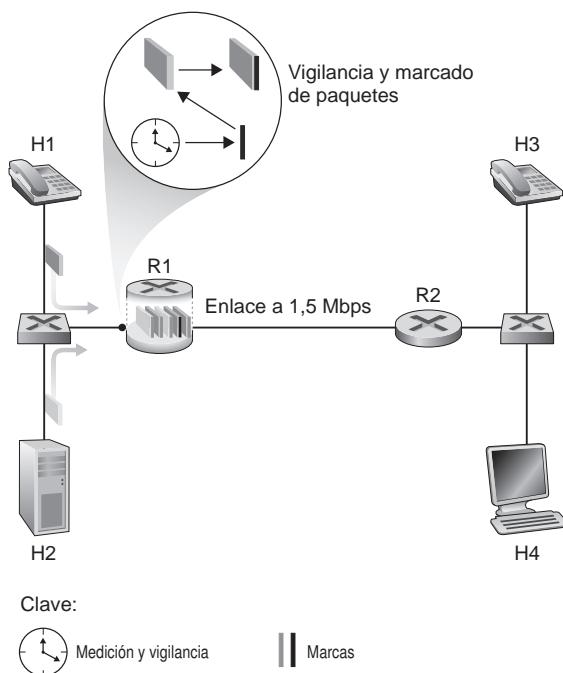


Figura 9.12 ♦ Vigilancia (y marcado) de las clases de tráfico de audio y HTTP.

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:

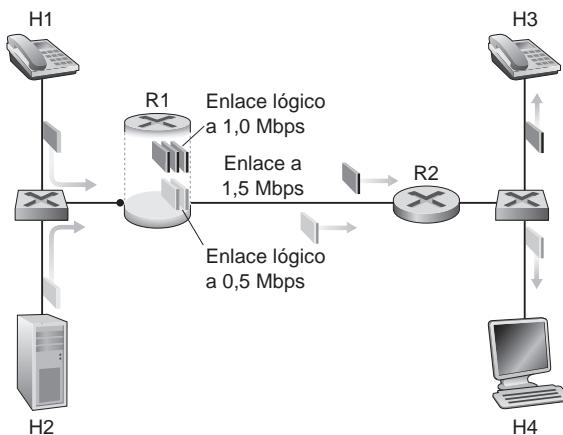


Figura 9.13 ♦ Aislamiento lógico de las clases de tráfico de audio y HTTP.

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.

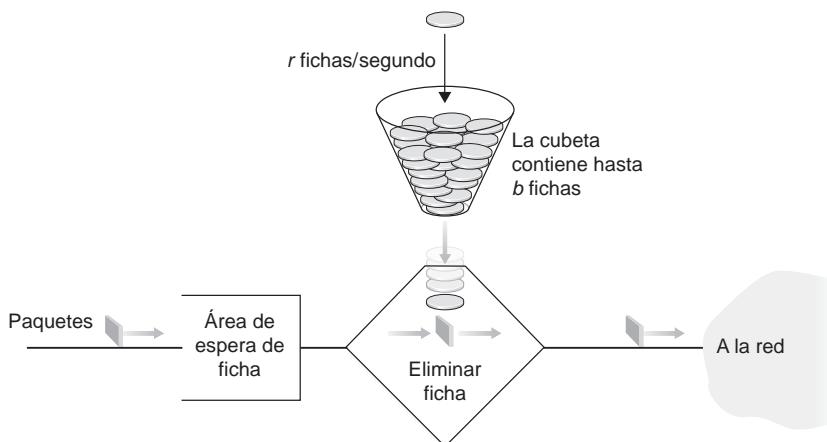


Figura 9.14 ♦ El controlador de la cubeta con pérdidas.

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.) Consideraremos 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.) Consideraremos 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

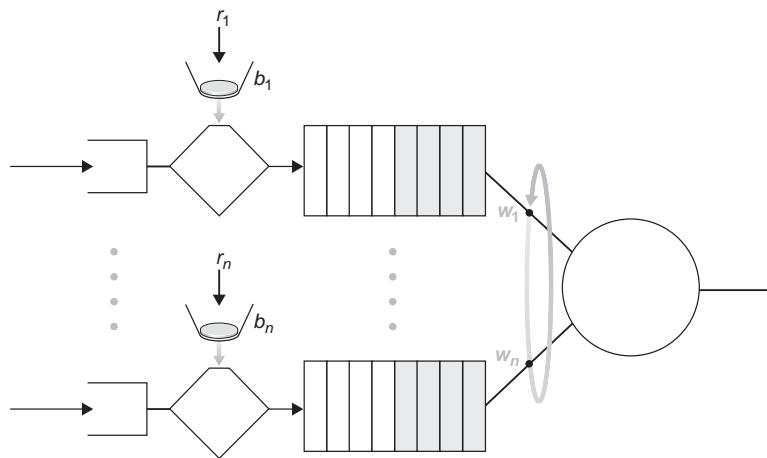


Figura 9.15 \blacklozenge n flujos multiplexados con cubetas con pérdidas y planificación WFQ.

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

$$d_{\max} = \frac{b_1}{R \cdot w_1 / \sum w_j}$$

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

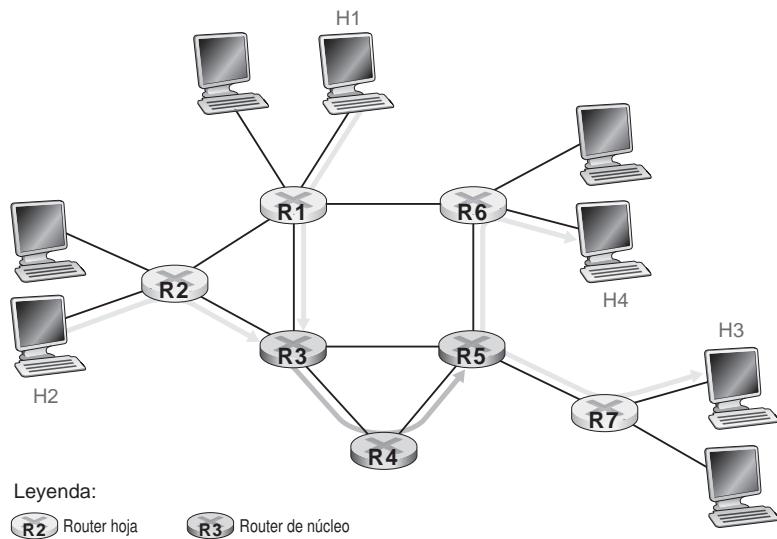


Figura 9.16 ♦ Un ejemplo de red Diffserv simple.

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 crítica, 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).

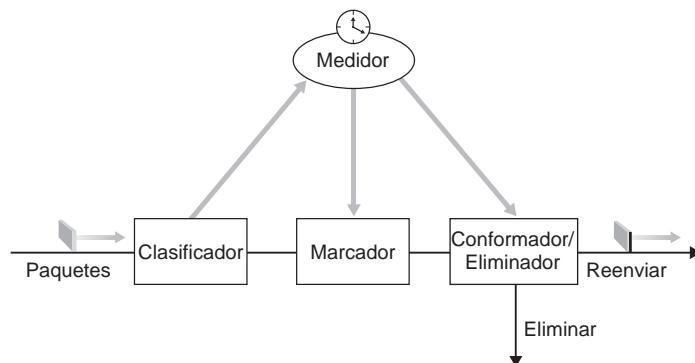


Figura 9.17 Visión lógica de las funciones de clasificación de paquetes y acondicionamiento de tráfico en el router terminal.

- 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/anchos 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 commutación 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 transmitido. 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).

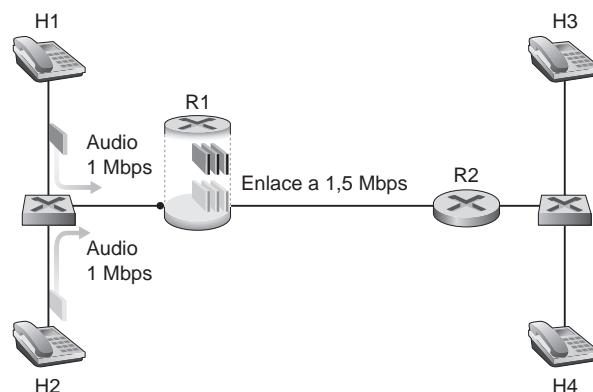


Figura 9.18 ♦ Dos aplicaciones de audio competidoras sobrecargan el enlace de R1 a R2.

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 a 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

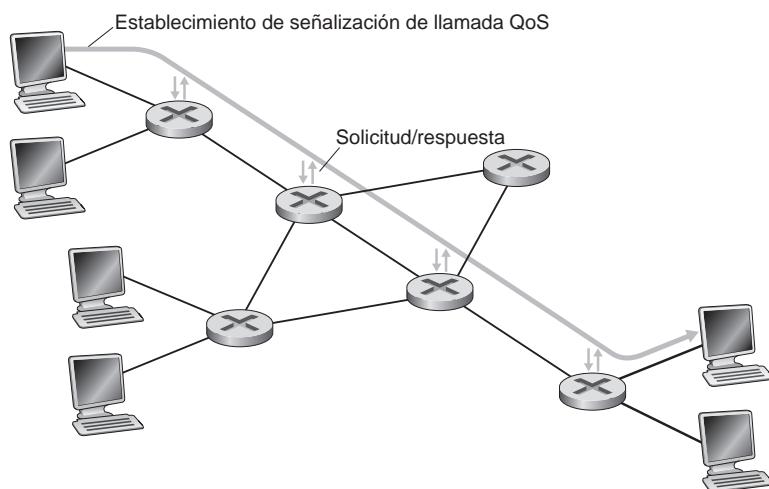


Figura 9.19 ♦ Proceso de establecimiento de llamada.

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. Descríbalos 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?