El nivel de aplicación

Joan Manuel Marqués Puig
Silvia Llorente Viejo

PID_00147724

UOC
Universitat Oberta de Catalunya
www.uoc.edu

Índice

Introducción

En este módulo didáctico, se describe toda una serie de aplicaciones que utilizan Internet como medio de comunicación y también los protocolos de comunicaciones que tienen asociados. Estas aplicaciones se conocen como aplicaciones distribuidas, ya que están formadas por diferentes partes y cada una se encuentra en máquinas diferentes.

Normalmente, hay una parte denominada *servidor* que se ejecuta en un ordenador, a la cual se conectan los diferentes clientes (que se encuentran en otros ordenadores remotos) con el fin de pedir los servicios (generalmente, solicitan la ejecución de algún tipo de operación).

En este módulo, veremos en primer lugar algunos conceptos básicos referentes a las aplicaciones distribuidas, y después, una serie de aplicaciones distribuidas en Internet, cuyas características siguientes estudiaremos.

- **El modelo**: describimos los diferentes elementos que forman la aplicación y sus características.

- **El protocolo**: exponemos las ideas principales del protocolo de comunicaciones de cada aplicación.

- **El modelo de información**: describimos el formato de información de los protocolos que tienen uno concreto.

- **La funcionalidad**: explicamos la funcionalidad y los comandos que la proporcionan.

Incluimos ejemplos breves del intercambio de comandos para facilitar su comprensión.

Objetivos

En este módulo didáctico, encontraréis las herramientas necesarias para alcanzar los objetivos siguientes:

1. Comprender el modelo cliente-servidor y el modelo *peer-to-peer*, que sirven como base de la implementación de aplicaciones distribuidas.

2. Conocer las aplicaciones distribuidas más utilizadas en Internet y entender los principios básicos de funcionamiento de los protocolos de comunicación que estas usan.

1. Arquitecturas de aplicaciones distribuidas

Hay muchas definiciones de aplicaciones distribuidas. Aquí os ponemos una que creemos que encaja muy bien con el enfoque de este módulo:

> Una aplicación distribuida está formada por una colección de ordenadores autónomos enlazados por una red de ordenadores y soportados por un *software* que hace que la colección actúe como un servicio integrado.

Una arquitectura de *software* determina cómo se identifican y se asignan los componentes del sistema; cómo interactúan los componentes para formar el sistema; la cantidad y la granularidad de la comunicación que se necesita para la interacción, y los protocolos de la interfaz usada por la comunicación.

La arquitectura que hay que utilizar en cada caso depende de muchos factores. Ahora mencionaremos las características más significativas que hay que considerar cuando se diseña una aplicación a escala Internet.

La primera característica es la **gran cantidad tanto de ordenadores como de usuarios** que hay en Internet. Relacionado con esto, está la dispersión geográfica de los mismos. La **dispersión geográfica** afecta a la manera en la que los componentes del grupo perciben las acciones que pasan a la aplicación. Por ejemplo, los usuarios de la aplicación pueden percibir dos acciones que pasan en dos extremos opuestos del mundo en diferente orden, según la proximidad de cada componente al componente generador de la acción. Según la aplicación, necesitaremos o no mecanismos para abordar estas situaciones.

Una tercera característica es **la autonomía** de los diferentes ordenadores que forman la aplicación. Es muy habitual que diferentes partes de una aplicación estén bajo dominios administrados por diferentes administradores. Relacionada con esta característica está la **seguridad**. Cada organización tiene unas políticas de seguridad diferentes, como por ejemplo cortafuegos. Es preciso que las aplicaciones distribuidas que diseñamos se puedan ejecutar teniendo en cuenta estas restricciones impuestas por los diferentes administradores.

Una cuarta característica es la **calidad de servicio**. En una aplicación distribuida a escala Internet, este es un aspecto fundamental que vendrá muy condicionado por la latencia de la red, los cortes y otros fenómenos que le puedan afectar.

Finalmente, están los aspectos relacionados con la **movilidad**: dispositivos que se conectan y desconectan; acceso desde diferentes ubicaciones; trabajo en modo desconectado; calidad de acceso menor (ancho de banda, fiabilidad, etc.). Este último aspecto, sin embargo, mejorará mucho a medida que estos servicios se generalicen.

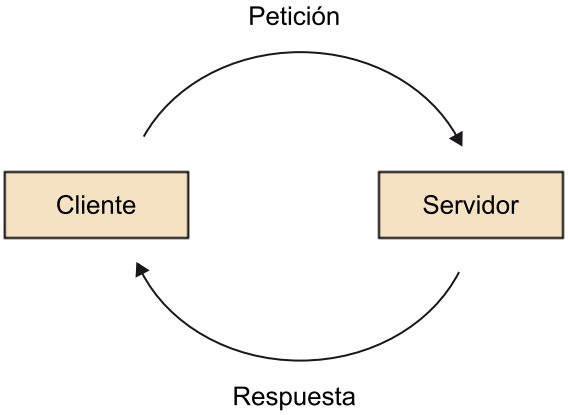
En este módulo, trataremos las dos arquitecturas distribuidas más utilizadas hoy día: cliente-servidor y de igual a igual[1]. Hay otras arquitecturas distribuidas como el *grid*, la publicación-suscripción o el código móvil, entre otras, que no trataremos en este material.

(1)De igual a igual en inglés se denomina ***peer-to-peer*** o ***p2p***.

1.1. Cliente-servidor (*client/server* en inglés)

En el modelo cliente-servidor hay dos tipos de componentes.

- **Clientes**: hacen peticiones de servicio. Normalmente, los clientes inician la comunicación con el servidor.
- **Servidores**: proveen servicios. Normalmente, los servidores esperan recibir peticiones. Una vez han recibido una petición, la resuelven y devuelven el resultado al cliente.

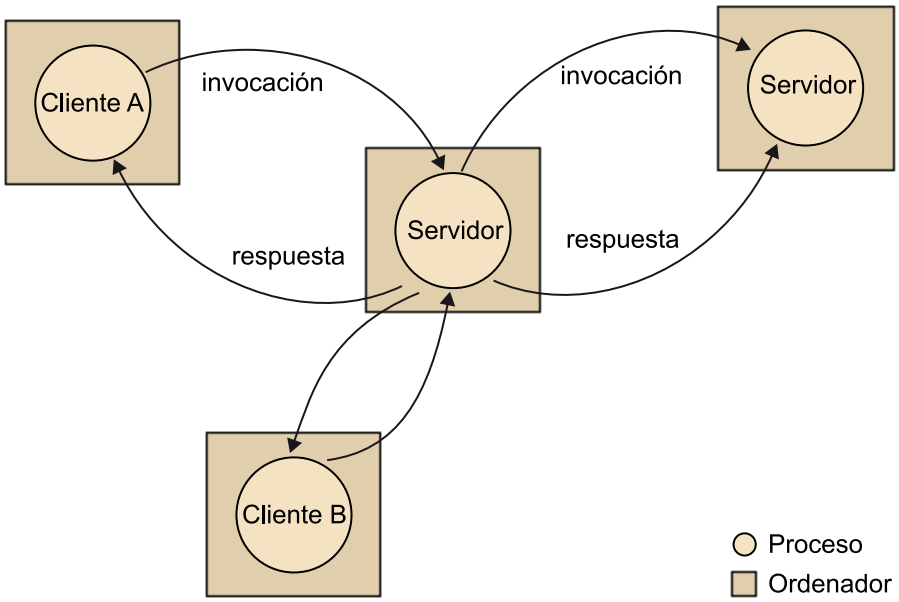


Esta idea se puede aplicar de manera muy variada a muchos tipos diferentes de aplicaciones. Un ejemplo que ayudará a entenderlo es la web. El navegador es el cliente y los ordenadores a los cuales nos conectamos y de los que obtenemos las páginas son los servidores.

Los servidores pueden ser con estado o sin estado. Un servidor sin estado no mantiene ninguna información entre peticiones. Un servidor con estado puede recordar información entre peticiones. El alcance de esta información puede ser global o específico de una sesión. Un servidor web con páginas web

estáticas es un ejemplo de servidor sin estado, mientras que un servidor que genere las páginas de manera dinámica (como la Intrauoc del Campus Virtual de la UOC) es un ejemplo de servidor con estado.

Un servidor también puede ser cliente de otros servidores, tal y como se ve en la figura siguiente. Por ejemplo, una aplicación de correo vía web actúa como servidor para el navegador y como cliente del servidor de correo que gestiona los mensajes del usuario en cuestión. Los servidores web y los otros servicios disponibles en Internet son clientes del servicio de resolución de nombres (DNS). Un tercer ejemplo son los buscadores *(search engines)*, que permiten a los usuarios acceder a sumarios de información de páginas web extendidas por muchos sitios web de toda Internet. Un buscador es al mismo tiempo servidor y cliente: responde a peticiones provenientes de los navegadores clientes y ejecuta programas que, actuando como clientes (robots web; en inglés, *web crawlers*), acceden a servidores de Internet buscando información. De hecho, un buscador incluirá diferentes hilos de ejecución, algunos sirviendo al cliente y otros ejecutando robots web.



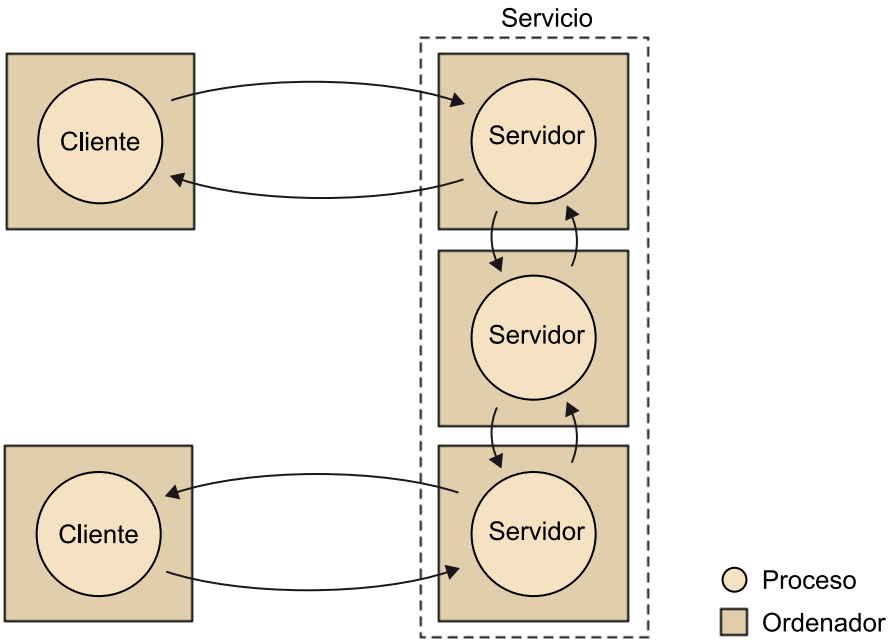
Los servicios también se pueden implementar como diferentes procesos servidores que se ejecutan en diferentes ordenadores y que interactúan para proporcionar un servicio a procesos clientes siguientes. Los servidores se pueden repartir los diferentes objetos que componen el servicio que proporcionan o pueden mantener réplicas de los objetos en distintos ordenadores.

Ejemplo de partición de datos

La web proporciona un ejemplo habitual de partición de los datos, en el que cada servidor gestiona su conjunto de recursos. Un usuario utiliza un navegador para acceder a un recurso situado en cualquiera de los servidores.

La reproducción se usa para incrementar el rendimiento y la disponibilidad y para mejorar la tolerancia a fallos. Proporciona múltiples copias consistentes de los datos en procesos que se ejecutan en diferentes ordenadores. Por ejemplo, la web proporcionada en

www.google.com (o www.uoc.edu) está registrada en diferentes servidores que tienen datos reproducidos.

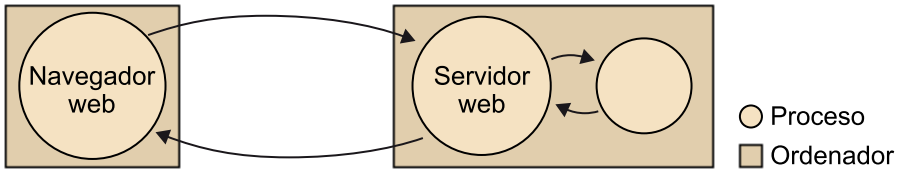


1.1.1. Aplicaciones basadas en la web

Un caso particular de aplicaciones cliente-servidor son las aplicaciones que se ejecutan aprovechando la arquitectura de la web. Estas aplicaciones se basan en el hecho de tener toda la capacidad de procesamiento en un servidor web (o conjunto de servidores) al que se accede desde un navegador web. Cuando el usuario hace clic sobre un enlace de la página web que tiene en su navegador, se genera una petición en el servidor que contiene la información. El servidor, al recibir la petición, retorna la página pedida si la petición era a una página, o retorna el resultado de ejecutar una aplicación si el enlace correspondía a un código para ejecutar (por ejemplo, CGI o Servlet). El navegador web proporciona a la aplicación un entorno donde presentar la información. La comunicación entre cliente y servidor se hace utilizando el protocolo HTTP. El resultado que retorna el servidor al cliente se envía en formato HTML, de manera que para el cliente web es totalmente transparente si accede a una página web o a una aplicación que genera un resultado formateado en HTML.

Las aplicaciones basadas en la web tienen la ventaja de que son accesibles desde cualquier ordenador que disponga de un navegador (la práctica totalidad de los ordenadores conectados hoy día a Internet), sin que sea preciso tener nada más instalado en el ordenador local. El uso de estas arquitecturas también facilita el diseño de las aplicaciones, ya que no hay que implementar la comunicación entre el cliente y el servidor (se utiliza el protocolo HTTP); y la parte de presentación de la aplicación se facilita mucho por el hecho de formatear el documento en HTML y aprovechar las funcionalidades que proporciona el navegador (como los intérpretes de Javascript y Java).

La facilidad y universalidad en el acceso a las aplicaciones que proporciona esta arquitectura es la base de los servicios ofrecidos en Internet. Algunos ejemplos son el Campus Virtual de la UOC; los servidores de correo web tipo Yahoo o Google, y los bancos por Internet.



1.2. De igual a igual (*peer-to-peer*, en inglés)

De una manera muy genérica, podemos decir que un sistema de igual a igual se caracteriza por ser un sistema distribuido, en el que todos los nodos tienen las mismas capacidades y responsabilidades, y en el que toda la comunicación es simétrica.

En Internet, desde sus orígenes, ha habido sistemas y aplicaciones que se han comportado siguiendo la filosofía de igual a igual. Estos sistemas se han caracterizado por el hecho de ser totalmente descentralizados, de gran escala y con capacidad para autoorganizarse. El ejemplo paradigmático son los mensajes Usenet.

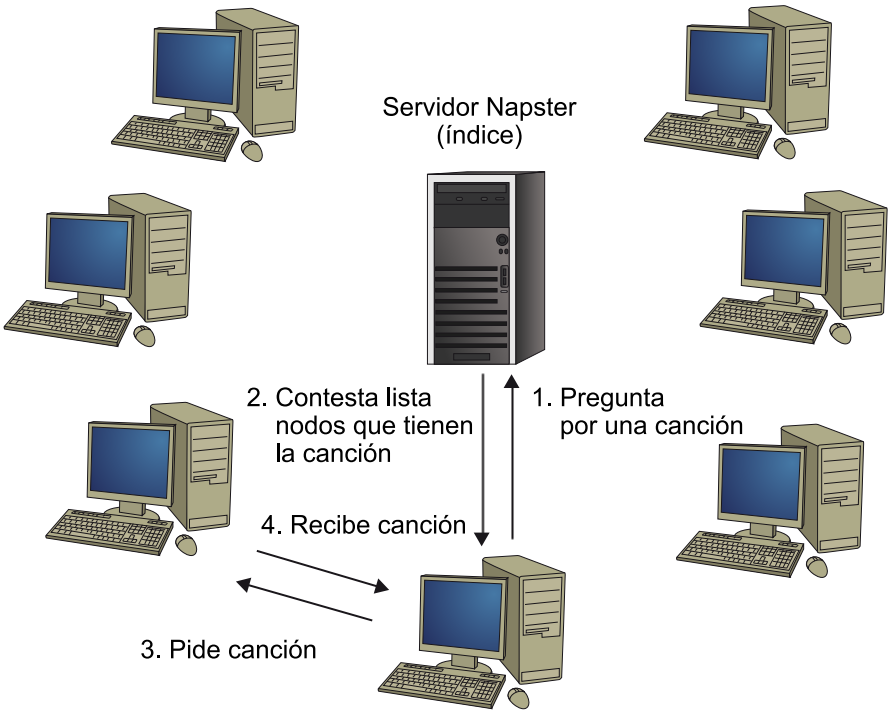
Los mensajes Usenet

Los mensajes Usenet (*Usenet News* en inglés) son un sistema global y descentralizado de grupos de discusión en Internet. Los usuarios leen y ponen mensajes que parecen correos electrónicos (se les denomina *artículos*) en un número determinado de grupos de noticias, que están organizados formando jerarquías lógicas de temas (por ejemplo, informática.linux.distribuciones o informática.lenguajesProgramación.tutoriales). Los mensajes se distribuyen entre un gran número de servidores, que almacenan y se reenvían mensajes unos a otros. Los usuarios se bajan los mensajes y ponen otros nuevos en uno de los servidores. El intercambio de mensajes entre los servidores hace que, a la larga, los mensajes lleguen a todos los servidores.

Sin embargo, el fenómeno de igual a igual empieza como tal a finales de los noventa, de la mano de Napster. En la época de la explosión de Internet –a partir de 1994–, la vertiente de colaboración que había dominado Internet hasta aquel momento empezó a perder protagonismo frente a la vertiente más comercial, que imponía la arquitectura cliente-servidor, liderada por la web como arquitectura estrella.

Dentro de este contexto dominado por la centralización de la arquitectura cliente-servidor, los usuarios de Napster descubrieron que el efecto agregado de que cada individuo pusiera canciones al servicio de una comunidad era que los participantes de la comunidad encontraban con facilidad las canciones que les interesaban.

El funcionamiento de Naspter era muy sencillo. Usaba un servidor (o índice) para proporcionar un servicio de directorio. Cuando un usuario arrancaba un nodo de Naster, este se conectaba al servidor y publicaba la lista de canciones que el nodo local hacía pública. De esta manera, el servicio de directorio sabía, para cada igual, qué objetos tenía disponibles para compartir. Cuando alguien buscaba una canción, hacía una petición de la canción al servidor, y este le contestaba la lista de nodos que tenían un título similar. El usuario elegía uno y se bajaba la canción directamente de este nodo.



Los grandes cambios que aporta Napster, tanto desde el punto de vista de la arquitectura como del funcionamiento del sistema, con respecto a las soluciones centralizadas (cliente-servidor) que predominaban en aquel momento son:

- Los ficheros que hay disponibles son los que los usuarios, de manera individual, deciden aportar al sistema (autonomía de los usuarios).

- La disponibilidad de un fichero depende de si los usuarios que lo tienen están conectados al sistema (conectividad puntual o *ad hoc* en inglés).

- Hay muchos usuarios que proporcionan un mismo fichero (tolerancia a fallos).

- Los recursos necesarios para almacenar los ficheros los aportan los mismos usuarios (coste del sistema).

- El sistema evoluciona y se adapta a medida que los usuarios se conectan o desconectan (autoorganización).

- El sistema soporta a muchos usuarios (escalabilidad). De hecho, llegó a haber millones de usuarios conectados a Napster.

- Al haber muchas reproducciones de una canción, la carga está repartida (mejora de rendimiento).

Aunque hubiera un servidor o índice, se considera que Napster era un sistema de igual a igual, porque los ficheros se encontraban en los ordenadores de los usuarios y la bajada se hacía directamente entre el ordenador que buscaba la canción y el que la ofrecía.

Esta manera de organizar grandes cantidades de información a escala Internet para facilitar la compartición resultó muy eficaz. La prueba de esto la encontramos tanto en el hecho de que el sistema llegó a tener millones de usuarios, como en el hecho de que muchas empresas discográficas lo vieron como una amenaza. Precisamente, el motivo de que Napster dejara de funcionar fue que denunciaron a los propietarios del servicio de directorio por infringir las leyes del *copyright*. El caso Napster acabó con una sentencia judicial que forzó el cierre del servidor índice.

A raíz del éxito de la solución, mucha gente se animó a hacer propuestas más descentralizadas y que superasen las limitaciones de Napster. Gnutella es un ejemplo de esto: se basa en un algoritmo de inundación para localizar el fichero que se busca y, de esta manera, elimina el punto único de fallo que supone tener un servicio centralizado de directorio. Una vez localizado el fichero, la bajada se hace directamente entre quien quiere el fichero y quien lo proporciona. Esta solución tiene el inconveniente de que no es determinista.

Determinismo

Por determinismo entendemos que diferentes ejecuciones de una misma operación den el mismo resultado. Sistemas como el de tipo Gnutella no son deterministas. El algoritmo que utilizan para localizar ficheros dentro del sistema no garantiza que si un fichero está en alguno de los iguales lo encuentre. Puede que, según el camino que haya seguido la consulta, nos diga que no lo ha encontrado, cuando sí que está.

1.2.1. Aplicaciones de igual a igual

Los sistemas y las aplicaciones de igual a igual se han hecho populares de la mano de las aplicaciones de compartición de ficheros, pero hay otros tipos de aplicaciones. Skype es otro sistema tipo de igual a igual que es muy popular. Skype proporciona telefonía en Internet. Utiliza un protocolo propietario de la implementación del que se conocen pocas cosas. Funciona siguiendo una organización con superiguales tal y como lo hace KaZaA. De hecho, Skype fue fundada por los fundadores de KaZaA. Un aspecto que hay que destacar es que consigue superar los problemas que tienen los iguales cuando están detrás de un cortafuego o los problemas derivados del NAT *(network address translation*).

Ved también

Las aplicaciones de igual a igual para la compartición de ficheros como KazaA están explicadas en el apartado "Aplicaciones de igual a igual para la compartición de ficheros" de este módulo didáctico.

También hay otros sistemas de igual a igual para la comunicación síncrona (como la mensajería instantánea), juegos, sistemas de procesamiento distribuido (como SETI@home) o *software* para la colaboración (como Groove).

SETI@home

SETI@home es un proyecto que tiene como objetivo detectar vida inteligente fuera de la Tierra. Distribuye procesamiento entre muchos ordenadores personales que están suscritos al proyecto y analiza datos de radiotelescopios, aprovechando las grandes cantidades de tiempo de procesamiento que los PC desperdician porque no hacen nada.

Groove

Groove es un sistema de igual a igual para facilitar la colaboración y comunicación en grupos pequeños. Proporciona herramientas para la compartición de ficheros, la mensajería instantánea, el calendario, la gestión de proyectos, etc.

Bibliografía complementaria

Encontraréis más información sobre el funcionamiento interno de Skype en:

S. A. Baset; H. Schulzrinne (2006, abril). "An analysis of the Skype peer-to-peer internet telephony protocol". *Proceedings of IEEE INFOCOM* 2006. Barcelona.

1.3. Ventajas y desventajas de cliente-servidor y de igual a igual

Como se ha visto, cliente-servidor y de igual a igual son dos maneras de plantear el diseño de una aplicación. También se ha mencionado que hay otros paradigmas. Este esfuerzo de caracterización, sin embargo, no nos tiene que confundir. A la hora de la verdad, las aplicaciones no implementan nunca una arquitectura pura y muchas veces son híbridos entre diferentes modelos. Cada aplicación tiene sus necesidades y hay que utilizar las características que nos ofrece cada paradigma con el fin de construir una aplicación que satisfaga las necesidades de sus usuarios.

La tabla siguiente pretende resumir las ventajas y desventajas tanto del cliente-servidor como del igual a igual. No pretende ser exhaustiva. Además, hablar de ventajas y desventajas es muy personal o dependerá de cada situación. Lo que para alguien o en una situación es una ventaja, para algún otro u otra situación es una desventaja. Y al revés pasa lo mismo.

	Ventajas	Desventajas
Cliente-servidor	• Datos en los servidores, lo que facilita el control de la seguridad, control de acceso, consistencia de la información, etc. • Hay muchas tecnologías maduras para los sistemas cliente-servidor. Esto hace que haya soluciones muy probadas y que garantizan la seguridad, facilidad de uso y amigabilidad de la interfaz. • Relativa facilidad para actualizar los elementos de un sistema cliente-servidor. • El desarrollo de aplicaciones centralizadas es más sencillo que el de las descentralizadas. • Hay unos responsables de garantizar el servicio, que son los proveedores del servicio (o de cada una de las partes de estos).	• La centralización del modelo: punto único de fallo, dependencia de la autoridad administrativa que provee el servicio, vulnerabilidad a ataques. • Escalabilidad condicionada a la capacidad de hacer crecer el servidor o conjunto de servidores que proporcionan el servicio. Relacionado con esto, es muy costoso construir un sistema que soporte cargas altas de contenidos pesados. • Cuando hay muchos clientes que hacen peticiones, el servidor se puede congestionar. En entornos en los que la demanda puede ser muy variable o incierta, es necesario sobredimensionar el servicio para atender cargas que quizá sólo ocurren de manera esporádica o asumir que en ciertas situaciones puede haber congestión.

	Ventajas	Desventajas
De igual a igual	• Descentralización: los diferentes miembros (iguales) del sistema son los propietarios y tienen el control de los datos y recursos de su ordenador. • El coste de la propiedad (o la mayor parte de esta si es un sistema de igual a igual híbrido) está repartido entre los usuarios. • Escalable: dado que las operaciones se hacen directamente entre los iguales, el sistema puede soportar más operaciones que si hubiera un nodo que centralizara las operaciones. • Autoorganización: no hace falta que un componente del sistema supervise la evolución del sistema cuando cambia su escala (aumentan los usuarios o la carga); hay fallos, o los iguales se conectan o desconectan.	• El comportamiento descentralizado es complejo de implementar. • La responsabilidad del sistema está difundida entre los usuarios: si los usuarios no aportan recursos suficientes, el sistema no puede funcionar. • Tecnología muy nueva. Aunque hay sistemas muy usados y que funcionan bien, las técnicas utilizadas todavía tienen que madurar. • El dinamismo del sistema hace que el sistema sea utilizable sólo si su disponibilidad satisface las necesidades de los usuarios. En la mayor parte de los casos, esto pasa por la reproducción (tanto de información como de capacidad de procesamiento). • Seguridad: a los problemas habituales de los sistemas distribuidos hay que añadir mecanismos para que la información que se aporte sea válida, que no haya usuarios que obtengan información sin aportarla, etc. Estos aspectos todavía no están bien resueltos.

1.4. Requerimientos de las aplicaciones

Las aplicaciones dialogan utilizando un protocolo de transporte. En terminología de niveles, se dice que transporte ofrece unos servicios a aplicación, o dicho al revés, la aplicación requiere al nivel de transporte unas capacidades. En general, estas capacidades giran en torno a tres ejes.

- Transferencia fiable: algunas aplicaciones, como el correo electrónico o la transferencia de archivos, necesitan que la información que viaja llegue y llegue bien. La fiabilidad de la transferencia es entonces un requisito crucial. En cambio, hay aplicaciones que se basan en la transmisión de voz o imagen en tiempo real que no necesita esta fiabilidad, porque si se pierde uno o varios paquetes, la reproducción del sonido o la imagen original todavía es posible, aunque con defectos aceptables.

- Ancho de banda: una transmisión de imagen en tiempo real necesita un ancho de banda concreto para que se pueda hacer de manera aceptable, pero el envío de un mensaje de correo electrónico se puede hacer con cualquier ancho de banda, por pequeño que sea.

- Temporización: las aplicaciones que permiten comunicaciones en tiempo real, ya sea con sonido, imagen o los dos, necesitan que el ritmo de recepción de los paquetes sea constante, para que en recepción se pueda reproducir el sonido o la imagen original de manera continua, tal y como se captó en origen.

Por lo tanto, cada aplicación tiene unos requerimientos diferentes, y por esto en el nivel de transporte hay diferentes opciones para satisfacer estos requerimientos. Los protocolos originales de Internet (TCP y UDP) se concentraban en el primer requerimiento, porque las primeras aplicaciones que se especifi-

caron no eran de tiempo real. A medida que se han desarrollado este tipo de aplicaciones, se han especificado nuevos protocolos de transporte que ofrecen los servicios adecuados.

2. DNS: Servicio de nombres en Internet

Los ordenadores conectados a Internet están identificados de manera única por su dirección IP. Las direcciones IP son difíciles de recordar y por este motivo se utilizan nombres del estilo de www.uoc.edu o smtp.uoc.edu para identificar los diferentes recursos u ordenadores de Internet. El sistema de nombres de dominio[2] es un servicio desplegado en Internet que permite hacer la traducción entre nombres y direcciones IP.

[2]En inglés, *Domain Name System* (DNS).

Formato de las direcciones IP

Las direcciones IP tienen el aspecto siguiente: 213.73.40.99, donde cada punto separa un *byte* del 0 al 255 expresado en notación decimal.

El DNS[3] es un servicio que recibe millones de peticiones y tiene que ser capaz de resolverlas de manera muy rápida. Para hacernos una idea del volumen de peticiones que debe ser capaz de gestionar el DNS, sólo es necesario que nos fijemos en dos de las aplicaciones más populares de Internet: la web y el correo electrónico. Cada vez que escribimos una dirección web en el navegador, este hace una consulta al DNS para saber a qué dirección IP corresponde. Y cada vez que enviamos un mensaje de correo electrónico, se hace una consulta al DNS para saber a qué servidor de correo hay que enviar el correo correspondiente al dominio de la dirección de correo electrónico.

[3]El DNS está especificado en los RFC 1.034 y PFC 1.035 y actualizado en otros RFC.

A continuación, podemos ver los pasos que se siguen para que el ordenador del usuario pueda enviar una petición de página web al URL www.uoc.edu/eimt/index.html:

1) El ordenador del usuario es el que ejecuta la parte cliente de la aplicación DNS.

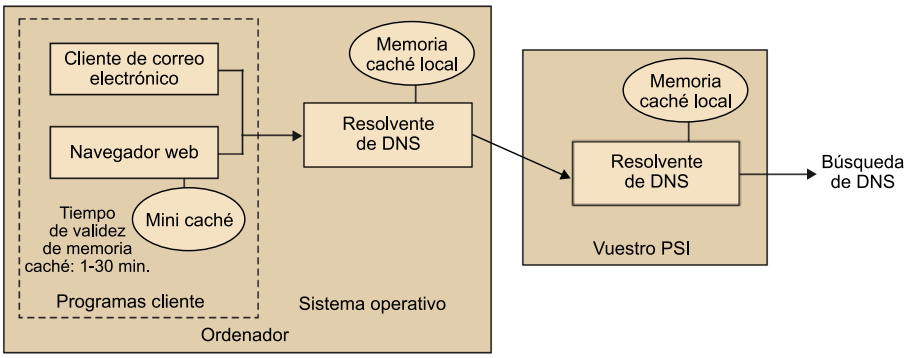
2) El navegador extrae el nombre del sitio web –www.uoc.edu en este caso– del URL y lo pasa a la parte cliente de la aplicación DNS.

3) El DNS cliente envía la petición (que contiene el nombre del sitio web) a un servidor DNS.

4) En algún momento futuro el cliente DNS recibirá la respuesta, que contiene la dirección IP correspondiente al sitio web.

5) Cuando el navegador recibe la dirección IP del DNS, puede iniciar la conexión TCP con el servidor HTTP ubicado en el puerto 80 de esta dirección IP.

La figura siguiente muestra cómo el correo electrónico o el navegador interactúan con el DNS.



Os habréis dado cuenta de que la pregunta al DNS introduce un retraso adicional –a veces sustancial– a la aplicación Internet que lo invoca. Afortunadamente, como veremos más adelante, la dirección IP está cacheada en algún DNS "cercano", lo que ayuda a reducir el tráfico DNS en la red, así como el retraso medio del DNS.

Otros servicios importantes que proporciona el DNS, aparte de los ya vistos de traducción entre el nombre del ordenador y su dirección IP, son los siguientes.

- **Alias**: a veces interesa que un ordenador tenga más de un nombre. En este caso, tiene un nombre que se denomina ***canónico***, que es el nombre del ordenador, y después tiene otros nombres que son sus alias. Estos alias normalmente son más sencillos de recordar. En este caso, se puede invocar el DNS para obtener el nombre canónico del ordenador, así como su dirección IP. En el caso de la UOC, www.uoc.edu es un alias del nombre canónico www-org.uoc.edu.

- **Alias del servidor de correo**: es importante que las direcciones de correo electrónico sean fáciles de recordar. El nombre del servidor que gestiona el correo puede ser más complicado de recordar y puede haber más de un servidor que responda cuando se pide por un dominio determinado. De hecho, el campo MX permite que una institución o empresa tenga el mismo nombre (un alias) para el dominio de correo electrónico y para el servidor web. Por ejemplo, en la UOC tanto el servidor web como el dominio de correo electrónico se denominan *uoc.edu*.

- **Distribución de carga**: el DNS se usa también para hacer distribución de carga entre servidores replicados, como servidores web. Sitios web muy accedidos pueden estar replicados en muchos servidores, cada uno corriendo en un ordenador diferente y con una dirección IP distinta. En este caso, una manera de repartir la carga entre los servidores es asociar un conjunto de direcciones IP a un nombre canónico. El DNS contiene este conjunto de direcciones. Cuando un cliente hace una consulta al DNS por un nombre al que le corresponde un conjunto de direcciones contesta todas las direcciones IP, pero en cada respuesta rota el orden de estas. Puesto que normalmente el servidor HTTP envía la petición a la primera dirección IP de la lista, se consigue una distribución de carga entre los servidores.

Ved también

Para saber más sobre la distribución de contenidos, podéis ver el subapartado 3.8.

Esta rotación también se utiliza en el correo electrónico, y de este modo muchos servidores de correo pueden tener el mismo nombre. Últimamente, compañías de distribución de contenidos como Akamai usan el DNS de maneras más sofisticadas para proporcionar distribución de contenidos web. Estas compañías utilizan el DNS para hacer que los usuarios se descarguen los contenidos de ubicaciones cercanas.

2.1. DNS: base de datos jerárquica y distribuida

El DNS es una base de datos jerárquica y distribuida organizada de manera autónoma, que proporciona un rendimiento a tiempo real a una audiencia global con contribuidores globales.

Un diseño sencillo para el DNS sería tener un servidor que contuviera todas las equivalencias entre nombres y direcciones IP. Los clientes enviarían las peticiones de resolución al servidor DNS y este las contestaría directamente. Esta solución centralizada sería sencilla de implementar, pero tendría numerosos inconvenientes por el hecho de que actualmente en Internet hay millones de servidores. Sería necesario que tuviera una base de datos muy grande capaz de soportar un volumen de consultas muy grande, además de ser capaz de gestionar una frecuencia alta de actualizaciones de las entradas, como nuevos *hosts* o cambios en los *hosts*.

Con el fin de poder atender estos requerimientos de escala (número de entradas y número de peticiones), el DNS usa muchos servidores organizados de manera jerárquica y distribuidos por todo el mundo. Ningún servidor DNS tiene todas las equivalencias entre nombres y direcciones IP. Estas equivalencias están distribuidas en los diferentes servidores DNS.

Hay tres tipos de servidores DNS.

- **Servidores DNS raíz**: hay trece servidores DNS raíz (etiquetados de la A a la M). Cada uno de estos servidores raíz en realidad es un clúster de servidores reproducidos, por seguridad y fiabilidad.

- **Servidores DNS de nivel de dominio superior**[4]: son los responsables de los dominios de primer nivel como org, com, net, edu o cat, así como de todos los dominios de países (us, es, etc.).

- **Servidores DNS autorizados**[5]: cada organización con ordenadores accesibles desde Internet tiene que proporcionar registros DNS accesibles públicamente que permitan hacer la equivalencia entre los nombres de sus ordenadores y la dirección IP de estos. Un servidor DNS autorizado hospeda estos registros. Estos registros pueden estar en servidores DNS autoriza-

Dirección web recomendada

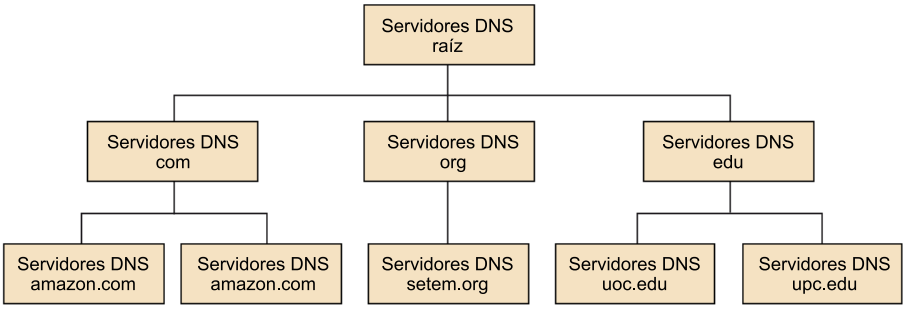
En http://www.root-servers.org/ podéis ver la ubicación de los servidores DNS raíz.

(4)En inglés, *top-level domain* (TLD).

(5)En inglés, *authoritative DNS servers*.

dos de la propia organización o en servidores DNS autorizados de algún proveedor de servicios. En este caso, la organización tiene que pagar por este servicio. La mayoría de las grandes empresas, universidades y centros de investigación mantienen sus servidores DNS autorizados primario y secundario (*backup*).

Porción de la jerarquía de servidores DNS



Hay otro tipo de servidores DNS, denominados ***servidores DNS locales***, que aunque estrictamente no pertenecen a la jerarquía de servidores, son claves para la arquitectura del DNS. Cada empresa proveedora de acceso a Internet (PAI o ISP en inglés) –como una universidad o una empresa de telecomunicaciones que proporciona acceso a Internet a usuarios domésticos– tiene un servidor DNS local. Cuando un nodo se conecta a su PAI, este le proporciona la dirección IP de uno o más servidores DNS locales (normalmente, esto se hace de manera automática vía DHCP). Cuando un ordenador hace una consulta al DNS, la consulta se envía al servidor DNS local, que actúa como *proxy*, y este la envía a la jerarquía de servidores DNS para que resuelva la petición.

Proveedor de acceso a Internet

Proveedor de acceso a Internet (PAI) o proveedor de servicios de Internet (en inglés, *Internet Servide Provider*, ISP) es una empresa que ofrece a sus clientes acceso a Internet. Un PAI conecta a sus usuarios a Internet mediante diferentes tecnologías como DSL (normalmente ADSL), módem de red telefónica conmutada, cable módem, Wi-Fi, etc.

Muchos PAI también ofrecen otros servicios relacionados con Internet como correo electrónico, hospedaje de páginas web, registro de dominios, etc.

DHCP

Dynamic Host Configuration Protocol (DHCP) es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración de manera automática. Se trata de un protocolo de tipo cliente-servidor. El cliente hace *broadcast* de una petición para información de configuración. El servidor DHCP recibe la petición y responde con información de configuración de su base de datos de configuración. Generalmente, un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes a medida que estas van estando libres, sabiendo en todo momento quién ha estado en posesión de aquella IP, cuánto tiempo la ha tenido o a quién se le ha asignado después.

En el caso de no utilizar DHCP, cada ordenador de la red tiene que configurarse manualmente, tarea costosa y propensa a errores.

Ejemplo de obtención de dirección IP

A continuación, hay un ejemplo de los pasos que se seguirían para que el ordenador `estudiante.pai.cat` obtenga la dirección IP del ordenador `sd.eimt.uoc.edu`.

1)`estudiante.pai.cat` envía la petición a su servidor DNS local.

2) El servidor DNS local reenvía la petición a un servidor DNS raíz.

3) El servidor DNS raíz retorna una lista de direcciones IP de servidores DNS responsables del dominio `edu`.

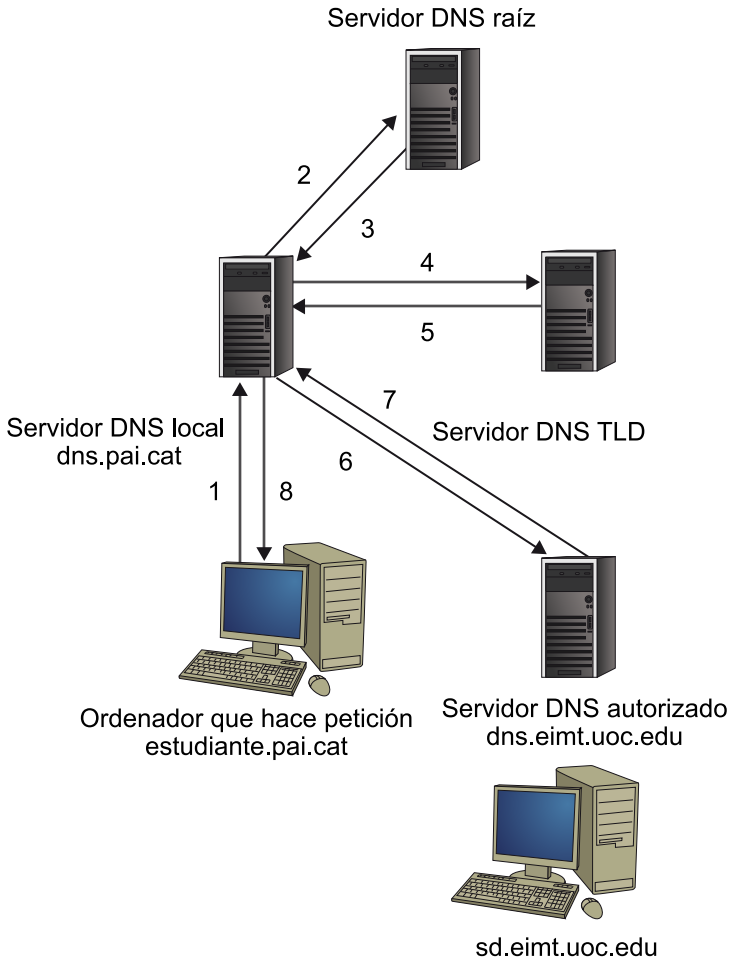
4) El servidor DNS local reenvía la petición a uno de estos servidores DNS TLD.

5) El servidor DNS TLD contesta la dirección IP del servidor autorizado por el dominio `uoc.edu`.

6) El servidor DNS local reenvía la petición al servidor DNS de la Universitat Oberta de Catalunya (`dns.uoc.edu`).

7) El servidor DNS de la Universitat Oberta de Catalunya responde la dirección IP del ordenador `sd.eimt.uoc.edu`.

Peticiones DNS iterativas



Frecuentemente, sucede que el servidor TLD no conoce el nombre del servidor autorizado sino sólo de un servidor DNS intermedio, el cual sí conoce el servidor DNS autorizado del ordenador. Supongamos que en el ejemplo que acabamos de ver, la UOC tenga un servidor DNS para toda la universidad –dns.uoc.edu–, y que el servidor DNS de cada departamento sea el autorizado para todos los ordenadores del departamento. En este caso, cuando el servidor DNS intermediario, dns.uoc.edu, recibe una petición por un ordenador con el nombre acabado en `eimt.uoc.edu`, retorna a dns.pai.cat la dirección IP de `dns.eimt.uoc.edu`, que es el servidor DNS autorizado para todos los ordenadores acabados en `eimt.uoc.edu`. El servidor DNS local preguntaría, finalmente, a `dns.eimt.uoc.edu` la dirección IP de `sd.eimt.uoc.edu`.

2.2. *Caching* de DNS

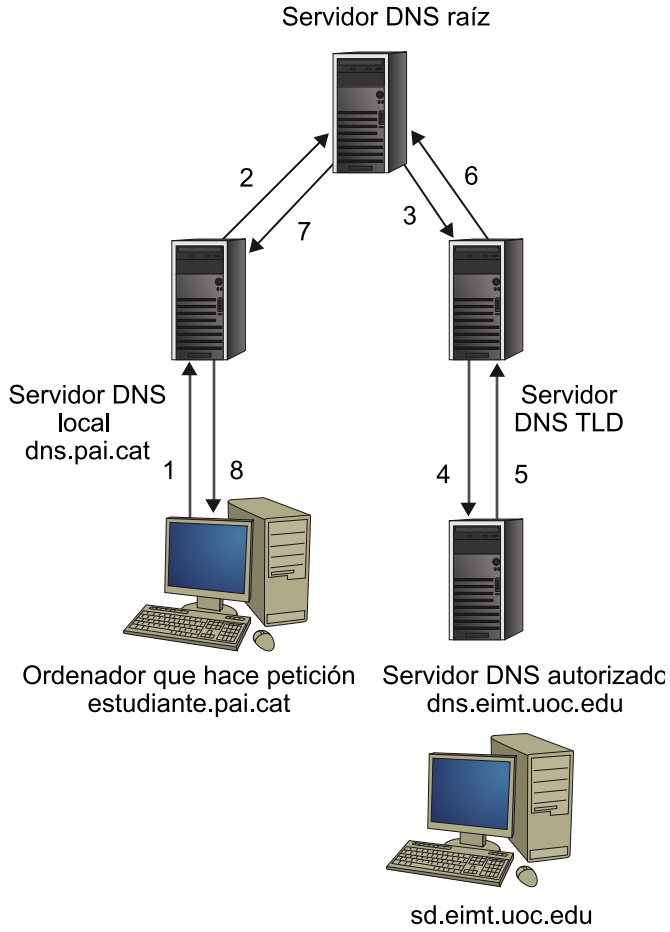
Como se puede ver, cada vez que se pide la resolución de un nombre se envían muchos mensajes, lo que hace incrementar el tiempo para obtener la respuesta y que haya muchos mensajes DNS circulando por Internet. Con el objetivo

de reducir este número de mensajes, el DNS hace *caching*. En el ejemplo de la última figura del subapartado 2.1, el servidor DNS local se guardaría una copia de la dirección IP responsable del dominio `uoc.edu`. De esta manera, si al cabo de un rato recibe una petición por un ordenador del dominio `uoc.edu`, ya preguntaría directamente al servidor DNS autorizado por `uoc.edu`. El servidor DNS local también puede guardarse una copia de la dirección IP de los servidores TLD y ahorrarse preguntar a los servidores DNS raíz. Dado que la equivalencia entre *hosts* y direcciones IP no es permanente, los servidores DNS descartan la información cacheada al cabo de un cierto tiempo (frecuentemente dos días).

2.3. Peticiones recursivas frente a peticiones iterativas

El ejemplo de la última figura del subpartado 2.1 utiliza tanto **peticiones recursivas** como **peticiones iterativas**. La petición de `estudiante.pai.cat` a `dns.pai.cat` es recursiva, ya que pide a `dns.pai.cat` que no resuelva la equivalencia en ningún sitio suyo. Las otras tres consultas son iterativas, ya que las respuestas se retornan directamente a `dns.pai.cat`. En teoría, las peticiones al DNS pueden ser recursivas o iterativas. La figura siguiente muestra la cadena de peticiones para el caso de una resolución recursiva. En la práctica, la mayoría de las peticiones siguen el esquema de la figura que hemos mencionado antes.

Petición DNS



2.4. Registros DNS

Los servidores DNS almacenan **registros de recursos**, que tienen los campos siguientes.

- Nombre.
- Valor.
- Tipo.
- Tiempo de vida[6]: el TTL es el tiempo máximo para que un recurso se borre de la caché.

[6]En inglés, *time to live* (TTL).

Los tipos más usuales son los siguientes:

- A: *Nombre* es el nombre del ordenador y *Valor* es su dirección IP.

- NOS: *Nombre* es un dominio (por ejemplo, `uoc.edu`) y *Valor* es el nombre de un servidor autorizado para este dominio (por ejemplo, `dns.uoc.edu`).

- CNAME: *Valor* es el nombre canónico de un alias (*nombre*). Permite obtener el nombre canónico de un alias (por ejemplo, `sd.uoc.edu`, `einfsun1.uoc.edu`, `CNAME`).

- MX: *Valor* es el nombre canónico de un servidor de correo que tiene Name como alias (por ejemplo, `uoc.edu`, `correu2.uoc.edu`, `MX`). MX permite que los servidores de correo tengan nombres más sencillos. También permite que una organización pueda tener el mismo alias para el dominio de correo que para el servidor web. Si se quiere obtener el nombre canónico del servidor de correo, se preguntará por el campo MX y si se quiere obtener el del otro servidor, se preguntará por el campo CNAME.

2.5. Consideraciones de seguridad

El DNS se diseñó sin considerar la seguridad. Un tipo de vulnerabilidad es la contaminación de la memoria caché del DNS, que hace creer en un servidor DNS que ha recibido información auténtica cuando en realidad no es así.

Contaminación de la memoria caché del DNS

La contaminación de la memoria caché del DNS (*DNS cache poisoning* o *DNS cache pollution* en inglés) es una situación creada maliciosamente o de manera intencionada que proporciona datos a una caché de DNS que no está originada por el servidor DNS autorizado. Este hecho puede ocurrir por un error en el *software*, por mala configuración de los servidores DNS o por acciones malintencionadas que se aprovechan de la arquitectura abierta del DNS.

Domain Name System Security Extensions (DNSSEC) modifica el DNS para añadir respuestas firmadas digitalmente, lo que proporciona a los clientes DNS (resolvedores) autenticación del origen de los datos DNS, integridad de datos –pero no disponibilidad ni confidencialidad– y denegación de existencia autenticada.

Extensión del DNSSEC

A partir de julio del 2010, todos los servidores raíz tendrían que utilizar DNSSEC (http://www.root-servers.org/).

3. La web y el HTTP

La web es un sistema formado por millones de páginas escritas en hipertexto mediante el lenguaje de marcaje HTML y conectadas entre sí mediante vínculos, de manera que formen un solo cuerpo de conocimiento por el cual se puede navegar fácilmente. Con un navegador web, se pueden visualizar páginas que pueden contener texto, imágenes, vídeos y otro multimedia y navegar de unas páginas a otras usando los hiperenlaces.

HTML

El *Hyper Text Markup Language* (HTML) es un lenguaje de marcaje diseñado para estructurar textos y relacionarlos en forma de hipertexto. El World Wide Web Consortium (W3C) controla la evolución del HTML: http://www.w3.org.

La introducción de la web a principios de los noventa supuso una revolución en Internet. Internet dejó de ser una red utilizada mayoritariamente por investigadores, académicos y alumnos de universidad para pasar a ser un medio de comunicación, interacción y publicación de información que ha entrado a formar parte de nuestra cotidianidad.

La web se basa en tres estándares para funcionar:

- El *Uniform Resource Locator* (URL), que se encarga de dar una dirección única con el fin de localizar cada objeto.
- El *Hyper Text Transfer Protocol* (HTTP), que especifica la manera en la que se enviará y se recibirá la información entre el navegador y el servidor.
- El *Hyper-text Markup Language* (HTML), un método para especificar cómo se tiene que ver esta información en el navegador.

En este módulo nos centraremos en el HTTP. Antes, sin embargo, veremos los aspectos básicos de los URL. El URL es una cadena de caracteres que informa al navegador de:

- El ordenador donde está el recurso al que hace referencia.
- El protocolo que tiene que utilizar para obtener este recurso.
- La manera en la que el servidor web encontrará cuál es el recurso.

La sintaxis que se utiliza en el URL es:
`protocolo://[usuario:contraseña]@host[:puerto]/[camino]`,
donde `usuario:contraseña`, el `puerto` y el `camino` son opcionales.

Los protocolos más habituales son: http, ftp (*File Transfer Protocol*) y *file* (para el acceso y la localización de archivos dentro de sistemas de ficheros).

Ved también

El protocolo ftp se trata en el apartado 4 de este módulo.

El *host* identifica la máquina donde está el recurso. Puede ser un nombre (www.wikipedia.org) o una dirección IP (192.12.34.11). El puerto es el número de puerto TCP del servidor, donde se conectará el navegador.

Ejemplo

En la dirección http://www.empresa.com/producto/descipcion.html, http es el nombre del protocolo, www.empresa.com es la dirección del servidor de la empresa y /producto/descripcion.html es el camino hasta llegar al objeto en el servidor. En este caso no hay ningún puerto especificado, lo que quiere decir que se utiliza el puerto 80, que es el puerto por defecto de los servidores web.

3.1. HTTP: *Hypertext Transfer Protocol*

El HTTP[7] es un protocolo a nivel de aplicación para sistemas hipermedia colaborativos y distribuidos, que es la base de la web. El HTTP sigue el paradigma cliente-servidor. Clientes y servidores se intercambian mensajes HTTP. El HTTP define la estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes. Se basa en un transporte fiable –el más habitual es TCP–, aunque podría funcionar en otros transportes fiables. El puerto por defecto para establecer las conexiones es el puerto asignado oficialmente al servicio www, que es el puerto 80.

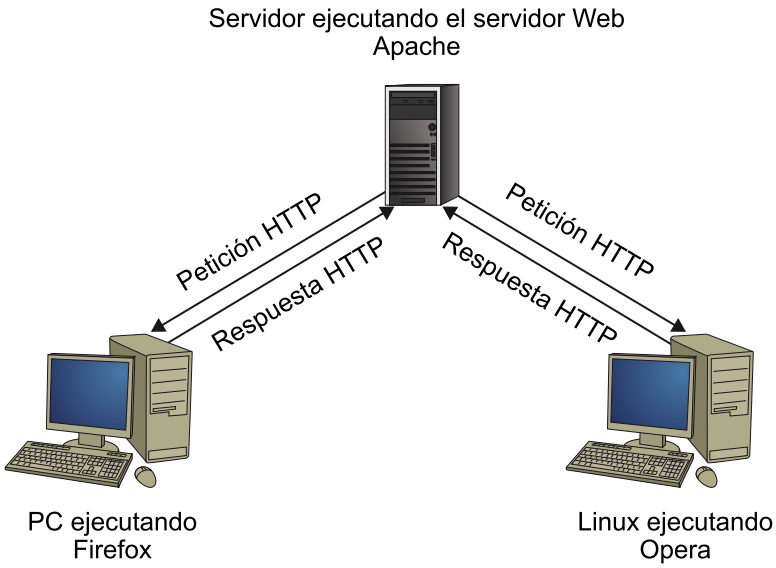
(7)El HTTP está definido en los RFC 1.945 y 2.616.

Una **página web** está formada por diferentes objetos. Un **objeto** es un fichero –que puede ser de diferentes tipos: un fichero HTML, una imagen, un *applet* Java o un vídeo– dirigible por un URL. La mayoría de las páginas web consisten en un fichero base formateado en HTML que referencia a diferentes objetos.

Ejemplo

Una página que contiene texto en HTML y dos imágenes en GIF contiene tres objetos: la página HTML base y las dos imágenes.

La figura siguiente ilustra el funcionamiento de la interacción entre un servidor web y unos clientes web: un cliente envía un mensaje HTTP al servidor y este lo procesa y contesta con otro mensaje HTTP. Dado que los navegadores web implementan la parte cliente HTTP, a menudo nos referiremos a clientes HTTP como navegadores. Los navegadores también implementan otras partes, como el visualizador que formatea los documentos HTML y los presenta a los usuarios.



Los servidores HTTP no mantienen ninguna información sobre los clientes, y por este motivo se dice que el HTTP es un **protocolo sin estado**.

3.1.1. Conexiones persistentes y no persistentes

El HTTP tiene dos modos de conexión.

- **Conexiones persistentes**: en el caso de las conexiones persistentes, se obra una conexión TCP diferente para enviar la petición/respuesta para cada objeto que contiene la página que se quiere bajar (es decir, la petición y respuesta de un mismo objeto viajan por la misma conexión).

- **Conexiones no persistentes**: en el caso de las conexiones no persistentes, todas las peticiones y respuestas de los objetos de una página se envían por la misma conexión TCP. Por defecto, el HTTP usa conexiones persistentes, aunque tanto los clientes como los servidores HTTP se pueden configurar para usar conexiones no persistentes.

	Ventajas	Inconvenientes
Conexiones persistentes	Después de enviar una respuesta, el servidor deja abierta la conexión TCP con el cliente. Esto permite que futuras peticiones y respuestas entre el cliente y el servidor se pueden enviar por esta conexión TCP (ya se trate de objetos pertenecientes a la misma página o de diferentes páginas). Además, no hace falta que esperen a que les toque el turno de ser servidas ya que la conexión ya está abierta. El servidor cierra la conexión TCP con el cliente cuando hace un rato que no se usa.	Tarda más en bajar los objetos de una página, ya que no aprovecha el paralelismo que se puede obtener bajando cada objeto de la página por separado.

	Ventajas	Inconvenientes
Conexiones no persistentes	Permite tener más de una conexión en paralelo, lo que reduce el tiempo de respuesta.	Hay que establecer más conexiones: • La bajada de cada objeto sufre un retraso, debido al tiempo de establecimiento de la conexión TCP y al tiempo de hacer la petición del objeto. • Para cada conexión hay que mantener información: *buffers* y otras variables del TCP, lo que puede ser una molestia para el servidor web, ya que puede estar atendiendo muchas peticiones de manera simultánea.

3.2. Formato de los mensajes HTTP

A continuación veremos los mensajes HTTP de petición y respuesta.

3.2.1. Mensaje HTTP de petición

Seguidamente, presentamos el ejemplo de un mensaje de petición HTTP.

```
GET /MiDirectorio/pagina.html HTTP/1.1
Host: www.servidor.edu
```

Como podéis ver, el mensaje está escrito en ASCII. Cada línea acaba con un *carry return* y un *line feed*. La última línea está acabada con una línea en blanco (una línea que sólo contiene un *carry return* y un *line feed*). Los mensajes pueden tener más líneas.

A la primera línea se la denomina ***línea de petición***. Las siguientes reciben el nombre de ***líneas de cabecera***. La línea de petición tiene tres campos.

- **El método**: puede tener diferentes valores `GET`, `POST`, `HEAD`, `PUT`, `DELETE`, etc. El método más usado es el GET. El método GET se usa cuando el navegador pide un objeto.

- **El URL**: identifica el objeto. En el ejemplo, se pide el objeto `/MiDirectorio/pagina.html`.

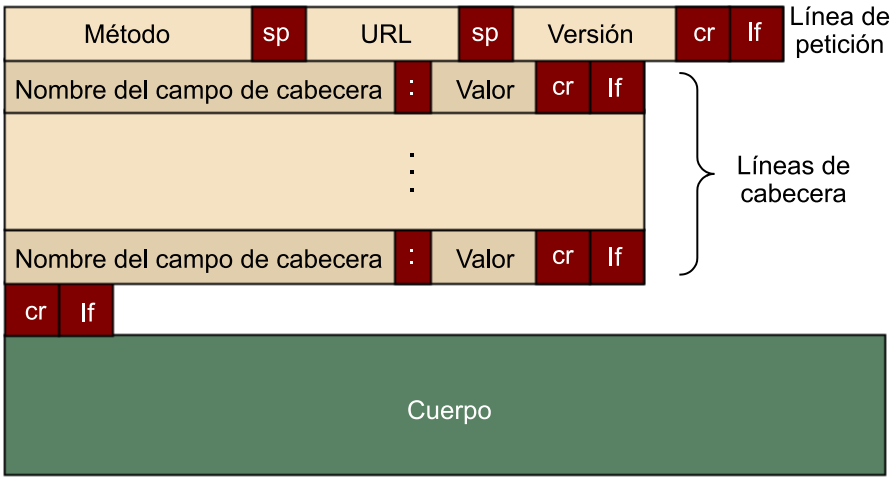
- **Versión de HTTP**: en el ejemplo, el navegador implementa la versión HTTP/1.1.

La línea de cabecera `Host: www.servidor.edu` indica en qué ordenador está el objeto. De entrada esta cabecera podría parecer superflua, pero es útil para los *proxy cache web*.

Ved también

En el subapartado 3.6 se tratan los servidores *proxy cache web*.

De manera más general, en la figura siguiente encontraréis el formato de los mensajes HTTP de petición.



Es fácil ver que el formato de los mensajes de petición sigue el ejemplo que hemos visto anteriormente, con una diferencia: el formato general contiene un "cuerpo". El cuerpo está vacío en el caso del método GET, pero se utiliza con el método POST. Los clientes HTTP acostumbran a utilizar el método POST para enviar datos de un formulario (por ejemplo, cuando un usuario proporciona palabras para hacer una busca en un buscador). Con un mensaje POST, el usuario está pidiendo una página web a un servidor pero el contenido de la página web depende de lo que el usuario ha introducido en los campos del formulario. Si el valor del campo método es POST, el cuerpo contiene lo que el usuario ha introducido en los campos de formulario.

Merece la pena mencionar que no siempre que se utiliza un formulario los datos se envían usando el método POST. Con frecuencia se usa el método GET y los datos se incluyen en el URL. Por ejemplo, si un formulario que utiliza el método GET tiene dos campos (modelo y cantidad) y los valores de los dos campos son AT y 23, entonces el URL tendrá la estructura siguiente:

```
www.negocio.com/compra?modelo=TA&cantidad=23
```

3.2.2. Mensaje HTTP de respuesta

A continuación, se muestra un ejemplo de posible mensaje HTTP de respuesta.

```
HTTP/1.1 200 Ok
Date: Wed, 24 Feb 2010 19:05:40 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Wed, 18 Feb 2009 21:11:55 GMT
Content-Length: 3245
Content-Type: text/html

(datos datos datos...)
```

La respuesta tiene tres partes.

- **Una línea de estado**, que tiene, a su vez, tres partes:
  - La versión del protocolo
  - Un código de estado
  - El mensaje correspondiente al código de estado

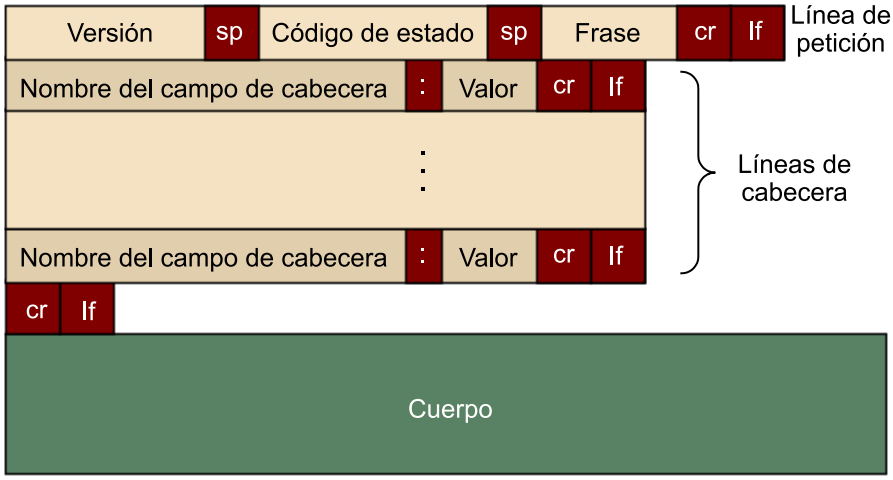
- **Unas cabeceras**. En el ejemplo anterior:
  - `Date:` indica el día y la hora en los que se creó y envió la respuesta. No es el día y la hora en los que se creó o modificó por última vez el objeto, sino el momento en el que el servidor obtiene el objeto del sistema de ficheros, lo incluye en el mensaje de respuesta y lo envía.
  - `Server:` indica que el mensaje lo generó un servidor web Apache.
  - `Last-Modified:` indica el día y la fecha en los que se creó o modificó por última vez el objeto.
  - `Content-Length:` indica el número de *bytes* del objeto que se envía.
  - `Content-Type:` indica que el objeto que contiene el cuerpo es del tipo texto HTML.

- **Un cuerpo**: contiene el objeto pedido.

Ved también

Veréis en el subapartado 3.6 que la cabecera `Last-Modified` es una cabecera crítica para los servidores caché.

En la figura siguiente, encontraréis el formato general de los mensajes HTTP de respuesta.



Algunos códigos de estado son los siguientes.

- `200 OK`: la petición ha tenido éxito y la información se retorna incluida en la respuesta.

- `301 Moved Permanently`: el objeto pedido se ha cambiado de ubicación. El nuevo URL está especificado en la cabecera `Location:` del mensaje de respuesta. El cliente obtendrá de manera automática el nuevo URL.

- `400 Bad Request`: código de error genérico que indica que el servidor no ha podido entender la petición.

- `404 Not Found`: el documento pedido no existe en el servidor.

- `505 HTTP Version Not Supported`: el servidor no soporta la versión del protocolo HTTP pedido.

Mensaje HTTP de respuesta real

Es muy fácil ver un mensaje HTTP de respuesta real. Sólo hay que hacer un Telnet en un servidor web y después teclear en línea un mensaje pidiendo un objeto que esté hospedado en el servidor. Podéis probar con el ejemplo siguiente[8]:

```
telnet www.uoc.edu 80
GET /index.html HTTP/1.1
Host: www.uoc.edu
```

Con Telnet se obra una conexión TCP en el puerto 80 del servidor www.uoc.edu. A continuación, se envía un mensaje HTTP de petición (aunque no veáis lo que estáis escribiendo, se enviará una vez pulséis el retorno. Si os equivocáis no sirve de nada borrar, ya que se enviarán todos los caracteres. Será necesario que volváis a empezar). Recibiréis un mensaje de respuesta que contendrá el fichero HTML de la página que habéis pedido.

(8)Recordad que hay que pulsar dos veces la tecla de retorno después de teclear la última línea.

3.3. HTTPS (HTTP seguro)

El *Hypertext Transfer Protocol Secure* (HTTPS) es una combinación del HTTP con el protocolo SSL/TLS para proporcionar cifrado e identificación segura del servidor. La idea principal tras el HTTPS es crear un canal seguro sobre una red insegura. Esto asegura una protección razonable frente a intrusos que quieran espiar la comunicación y ataques *man-in-the-middle*, siempre y cuando el certificado del servidor esté verificado y se pueda confiar en el mismo.

Ataque *man-in-the-middle*

El ataque *man-in-the-middle* es una forma de espiar en la que el atacante crea conexiones independientes con las víctimas y hace pasar los mensajes entre estas a través de él. Esto hace que las víctimas piensen que están hablando directamente entre sí a través de una conexión privada cuando, en realidad, la conversación está controlada por el atacante.

Un ataque *man-in-the-middle* tiene éxito sólo cuando el atacante puede hacerse pasar por cada una de las víctimas. La mayoría de los protocolos de cifrado incluyen alguna forma de autenticación para evitar este tipo de ataques. Por ejemplo, el SSL autentica el servidor mediante una autoridad de certificación en la que confían las dos partes.

El HTTPS confía en unas autoridades de certificación que se venden preinstaladas en los navegadores web. A partir de aquí, las conexiones HTTPS son fiables si y sólo si se cumple lo siguiente:

- El usuario confía en que la autoridad certificadora genera certificados para sitios web legítimos.

- El sitio web proporciona un certificado válido (un certificado firmado por una autoridad en la que se puede confiar).

- El certificado identifica de manera clara el sitio web.

- O bien los nodos de Internet que intervienen son fiables, o bien el usuario confía en que el protocolo de cifrado usado (TLS o SSL) hace que no se pueda espiar la comunicación.

3.4. Cookies

El HTTP es un protocolo sin estado. Esto simplifica el diseño del servidor y hace que este tenga un rendimiento elevado, ya que es capaz de soportar miles de conexiones TCP simultáneas. A veces, sin embargo, conviene que el servidor web pueda identificar a usuarios. Las cookies[9] se usan para este propósito. Permiten a los servidores web tener información de los usuarios. Las cookies se pueden usar para autenticar, almacenar preferencias del cliente, identificar una sesión, mantener un seguimiento de las compras en una tienda virtual o cualquier otro cosa que se pueda hacer a partir de almacenar datos textuales. La mayoría de los sitios web comerciales usan cookies.

[9]Las cookies están definidas en el RFC 2.965.

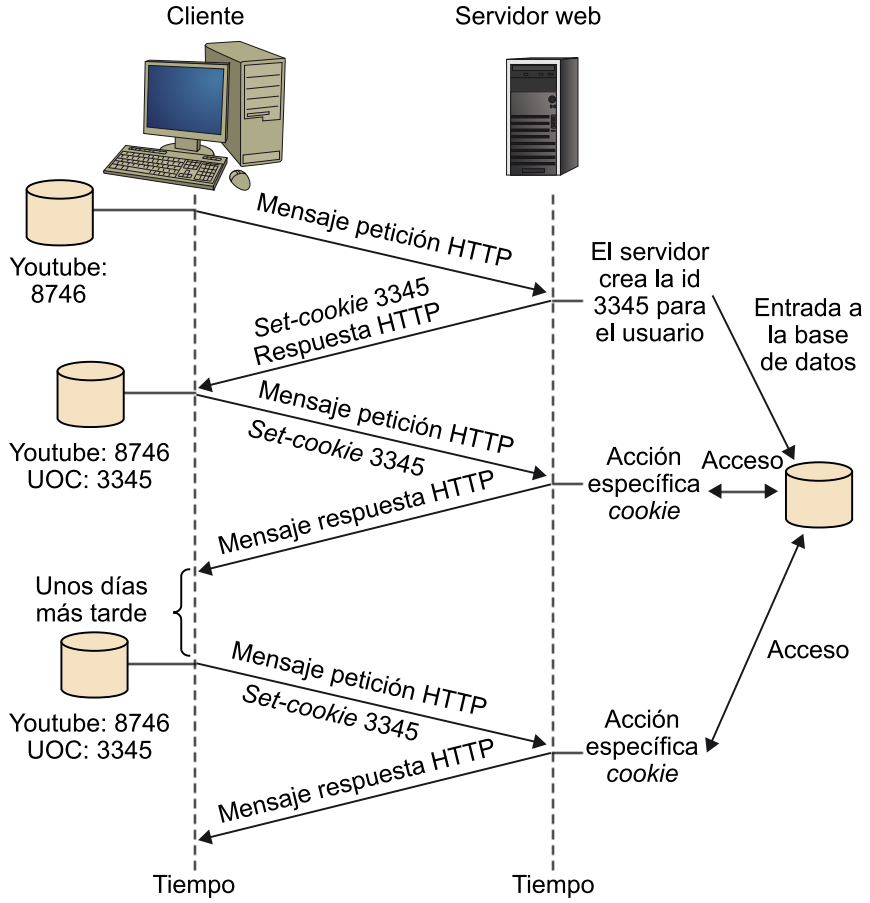
Una cookie es una pequeña porción de texto almacenado en el ordenador del usuario por el navegador web, que consiste en una o más parejas nombre-valor que contienen porciones de información.

El servidor web envía la cookie al navegador web como una cabecera de la respuesta. A partir de este momento, el navegador web envía la cookie cada vez que accede al servidor. El navegador web almacena las cookies en un fichero local gestionado por el navegador. El servidor web puede tener una base de datos con la información relacionada con las cookies.

Ejemplo de uso de una cookie

Supongamos que hay un cliente que se quiere conectar al servidor web de la UOC. Cuando la petición llega al servidor de la UOC, el servidor crea un identificador único y una entrada a una base de datos que este identificador indexa. A continuación contesta al navegador del cliente incluyendo la cabecera `Set-cookie:` en la respuesta HTTP, que contiene el identificador. El navegador recibe la respuesta HTTP y almacena la cookie recibida en un fichero en el que almacena las cookies.

En el ejemplo de la figura siguiente, el fichero de cookies ya contiene una de una conexión anterior a Youtube. El usuario continúa navegando por el sitio web de la UOC. Cada vez que pide una página web, el navegador consulta el fichero de cookies y añade una línea de cabecera a la petición HTTP. Si el usuario vuelve al sitio web de la UOC unos días más tarde, continúa añadiendo la cabecera `Cookie:` en los mensajes de petición.



Las cookies, al ser texto, no son ejecutables. Por el hecho de no ser ejecutables, no se pueden replicar por sí mismas y, por lo tanto, no son virus. Debido al mecanismo para guardar y leer las cookies, estas se pueden usar como *spyware*. Los productos *antispyware* pueden avisar a los usuarios sobre algunas cookies porque estas se pueden usar para hacer un seguimiento de las operaciones del usuario o vulnerar la privacidad.

La mayoría de los navegadores modernos permiten a los usuarios decidir si quieren aceptar o no las cookies de las páginas visitadas, pero la no aceptación de estas puede hacer que algunas páginas web no funcionen correctamente.

3.5. Características de la demanda de páginas web

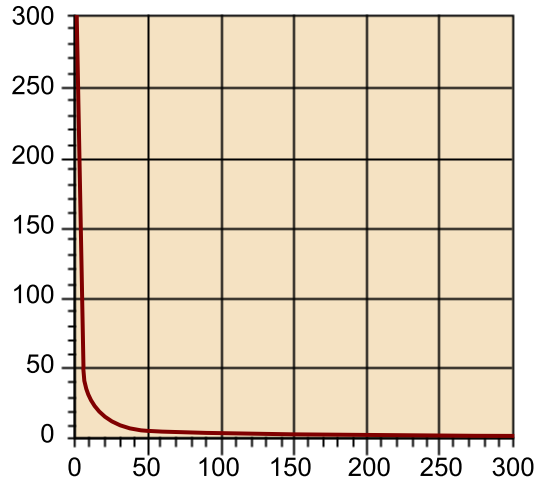
Un servidor web puede tener miles de documentos y, sin embargo, recibir la mayoría de las peticiones para un único documento.

Ley de Zipf

En muchos casos, la popularidad relativa entre diferentes sitios web o entre distintas páginas de un cierto sitio web se rige por la ley de Zipf[10], que formulada de manera sencilla sería: muy pocos casos tienen mucha popularidad y muchos casos tienen muy poca.

[10]La ley de Zipf debe su nombre a George Kingsley Zipf (1902-1950).

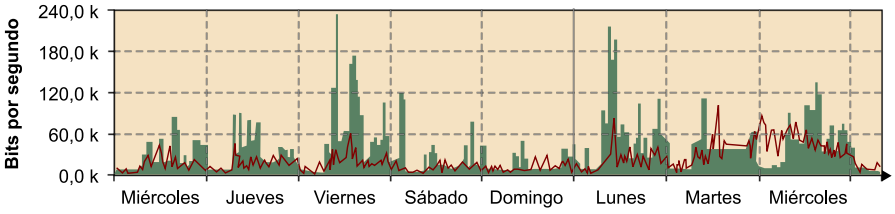
Distribución de popularidad que sigue la ley de Zipf



Casos ordenados por popularidad en el eje *x*, valor de popularidad en el eje *y*

La popularidad de un sitio web puede ser muy variable. Puede recibir muy pocas visitas durante mucho tiempo y, de repente, recibir varias veces más peticiones de las que puede servir: es un tráfico en ráfagas.

Evolución del tráfico entrante y saliente de un sitio web típico durante una semana



Podéis observar la gran variación horaria y la reducción de tráfico durante el fin de semana.

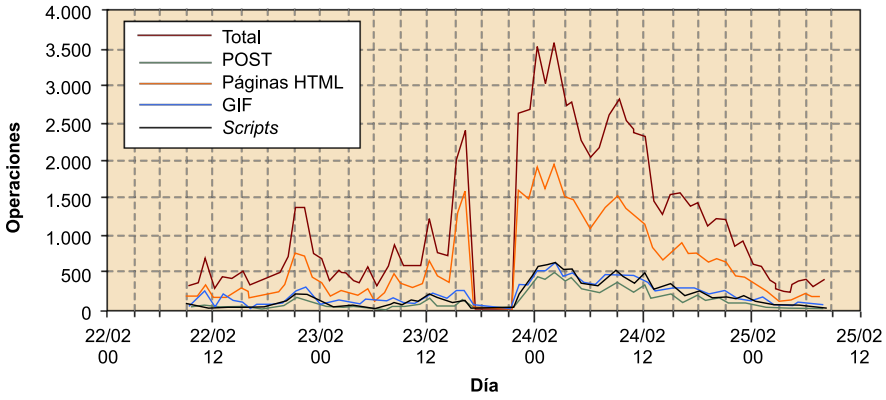
Un servidor puede recibir avalanchas repentinas de tráfico. Por ejemplo, por las estadísticas del servidor web que se ve en la figura siguiente sabemos que, después de ser anunciado en la página de noticias Slashdot, sufrió un exceso de visitas tan alto que el servidor se bloqueó.

Flash crowd

Un cuento de ciencia ficción de Larry Niven (1973) predijo que una consecuencia de un mecanismo de teletransporte barato sería que grandes multitudes se materializarían de manera instantánea en los lugares con noticias interesantes. Treinta años después, el término se usa en Internet para describir los picos de tráfico web cuando un determinado sitio web se convierte repentinamente en popular y es visitado de manera masiva.

También se conoce como efecto *slashdot* o efecto /., que se da cuando un sitio web resulta inaccesible a causa de las numerosas visitas que recibe cuando aparece en un artículo del sitio web de noticias Slashdot (en castellano, Barrapunto).

Peticiones web por hora, servidas por http://counter.li.org durante tres días

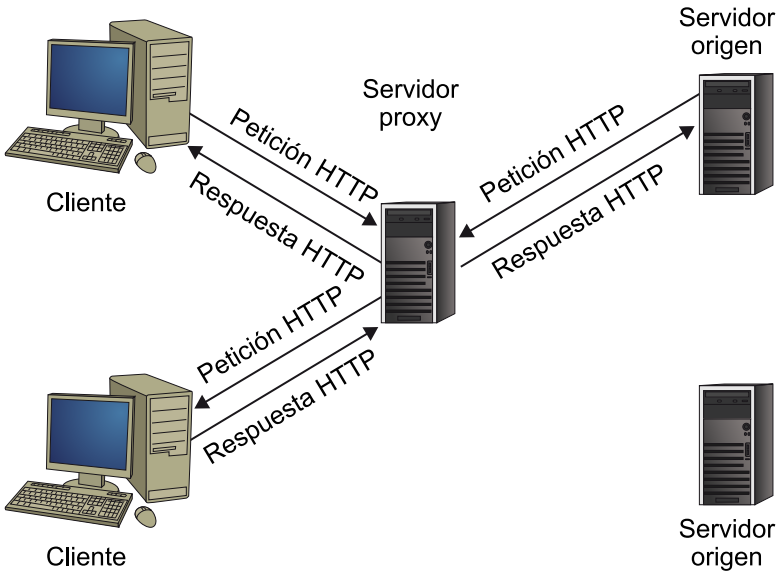


Se puede ver que mientras que el número habitual de operaciones (peticiones web) estaba por debajo de 500, subió rápidamente a unas 2.500, lo que provocó el fallo del sistema. Después de reconfigurarlo, estuvo soportando durante unas doce horas en torno a 3.000 peticiones por hora para bajar posteriormente a valores normales. La historia completa está en la dirección: http://counter.li.org/slashdot.

3.6. Servidores intermediarios

Un servidor intermediario[11] es una entidad de la Red que satisface las peticiones HTTP en lugar del servidor web al que iba dirigida la petición. Los servidores intermediarios mantienen una copia de los objetos accedidos recientemente en un disco del servidor. La figura siguiente muestra el funcionamiento general de un servidor intermediario.

[11]En inglés, *web cache* o *proxy-cache*.



El navegador del usuario se puede configurar para que todas las peticiones HTTP del usuario vayan primero al servidor web intermediario. A partir de este momento:

1) Cada nueva petición del usuario establecerá una conexión TCP con el *proxy-cache* y le enviará la petición HTTP.

2) Si el *proxy-cache* tiene una copia del objeto, lo retorna dentro de un mensaje HTTP de respuesta al navegador cliente.

3) En caso de que el *proxy-cache* no tenga el objeto pedido, este abre una nueva conexión TCP con el servidor origen de la petición para pedirle el objeto.

4) Al recibir la petición, el servidor origen envía el objeto en una respuesta HTTP al *proxy-cache*.

5) Finalmente, el *proxy-cache* almacena el objeto recibido en su disco local y envía una copia, en un mensaje HTTP de respuesta, al navegador cliente (en la conexión TCP establecida entre el cliente y el *proxy-cache*).

El *proxy-cache* actúa como servidor del cliente y como cliente del servidor. Los proveedores de acceso a Internet[12] son los que instalan los *proxy-cache*. Por ejemplo, una universidad puede instalar un *proxy-cache* en la red del campus y configurar todos los navegadores para que apunten al *proxy-cache*.

(12)En inglés, *Internet Service Providers* (ISP)

Hay dos razones principales para desplegar *proxy-cache* en Internet:

- Se puede reducir de manera significativa el tiempo de respuesta de las peticiones.
- Puede reducir de manera significativa el tráfico del enlace de la institución con Internet.

Los servidores intermediarios se usan también en otros protocolos además del HTTP. En general, se sitúan en una discontinuidad para hacer una función. Por ejemplo, para hacer un cambio de red: una máquina conectada a la red interna de una organización, que usa direcciones IPv4 privadas (por ejemplo, 10.*.*.*), y también a Internet, puede hacer de *proxy-cache* para la traducción de direcciones IP entre las dos redes (NAT).

***Network Address Translation* (NAT)**

En los paquetes IP salientes: sustituir la dirección IP de las máquinas internas (no válidas en Internet) por la suya propia; en los paquetes IP entrantes: sustituir su propia dirección IP por la de una máquina interna y reenviar el paquete hacia la red interna. Para saber a quién se tiene que entregar, el servidor intermediario debe asociar cada uno de sus puertos a las máquinas internas, ya que una dirección de transporte es una pareja (dirección IP, puerto).

3.7. El GET condicional

Hemos visto que el *caching* puede reducir el tiempo de respuesta percibido por el usuario, pero la copia que tenga la caché puede ser obsoleta. Es decir, el objeto hospedado en el servidor web puede haberse modificado de manera

posterior a que al cliente lo copiara en su caché. Para solucionarlo, el protocolo HTTP incluye un mecanismo que permite a la caché verificar que su copia del objeto está actualizada.

```
GET /MiDirectorio/pagina.html HTTP/1.1
Host: www.servidor.edu
If-modified-since: Wed, 18 Feb 2009 21:11:55 GMT
```

La cabecera `If-modified-since:` contiene el valor de la cabecera `Last-Modified:` de cuando el servidor envió el objeto. Esta petición GET, que se denomina *GET condicional*, indica al servidor que envíe el objeto sólo si el objeto se ha modificado desde la fecha especificada.

En el caso de que el objeto no se haya modificado desde la fecha indicada, el servidor web contesta un mensaje a la caché como el siguiente:

```
HTTP/1.1 304 Not Modified

Date: Tue, 24 Mar 2009 18:23:40 GMT

(sin cuerpo)
```

`304 Not Modified` en la línea de estado del mensaje de respuesta indica a la caché que puede pasar su copia del objeto al navegador que ha hecho la solicitud. Hay que destacar que el mensaje de respuesta del servidor no incluye el objeto, ya que esto sólo haría malgastar ancho de banda y la percepción del usuario sobre el tiempo de respuesta.

3.8. Distribución de contenidos

Los mecanismos de *proxy-cache* son útiles para que una comunidad de usuarios que comparten una conexión a Internet puedan ahorrar ancho de banda y reducir transferencias repetitivas de objetos. Sin embargo, esta sólo es una parte del problema. Diferentes agentes participan en el rendimiento de una transferencia web.

- **Proveedor de contenidos (servidor web)**: debe planificar su capacidad para dar servicio en horas punta y aguantar posibles avalanchas de peticiones, y de esta manera tener una cierta garantía de que sus clientes dispondrán de un buen servicio con cierta independencia de la demanda.

- **Cliente**: podría usar una memoria caché para economizar recursos de la red. Cuando un contenido se encuentra en más de un lugar, debería elegir

el mejor servidor en cada momento. El original o una réplica "rápida" (o llevar el objeto a trozos desde distintos lugares al mismo tiempo).

- **Réplica**: si un servidor guarda una réplica de algún contenido, probablemente es porque quiere ofrecerlo y reducir la carga del servidor original. Por lo tanto debe ser "conocido" por sus clientes, pero además el contenido tiene que ser consistente con el contenido original.

- **Proveedor de red**: debería elegir el mejor camino para una petición, vía direccionamiento IP, vía resolución DNS (resolver nombres en la dirección IP más próxima al cliente que consulte, aunque no es lo más habitual), vía servidores *proxy-cache HTTP* y evitar zonas congestionadas (*hot spots*) o *flash crowd* (congestión en el servidor y en la red próxima debida a una avalancha de demandas).

En este subapartado, nos centraremos en los sistemas de distribución de documentos para ayudar a mejorar el rendimiento de la web. Más concretamente, nos centraremos en qué puede hacer el proveedor de los contenidos.

Las aplicaciones que ofrecen contenidos en Internet se enfrentan al reto de la escala: un único servidor frente a millones de personas que de manera eventual pueden pedir sus servicios todos al mismo tiempo: el proveedor de información debe poner tantos recursos como audiencia pueda tener.

Una opción muy utilizada es que el proveedor del contenido disponga de diferentes réplicas de la información. Hay varios trucos para repartir peticiones entre los diferentes servidores que contienen las réplicas:

- *Mirrors* con un programa que redirige la petición HTTP a la mejor réplica.

- Hacer que el servicio de nombres DNS retorne distintas direcciones IP. Normalmente, se envía la petición HTTP a la primera dirección de la lista de direcciones IP recibidas del DNS. Si esta lista está ordenada de manera diferente en cada petición al DNS (método *round roben*), cada vez se hará la petición a una dirección IP diferente.

- Redireccionamiento a nivel de transporte (conmutador de nivel 4, *L4 switch*): un encaminador mira los paquetes IP de conexiones TCP hacia un servidor web (puerto 80) y las redirige a la máquina interna menos cargada.

- Redireccionamiento a un nivel de aplicación (conmutador de nivel 7, *L7 switch*): un encaminador que mira las conexiones HTTP y puede decidir con qué réplica contactar en función del URL solicitado. Muy complejo.

- Enviar todas las peticiones a un servidor intermediario inverso que responda con contenido guardado en la memoria o que pase la petición a uno o varios servidores internos.

Réplica

Una réplica (en inglés, *mirror*) es una copia exacta de otro lugar de Internet. Los *mirrors* proporcionan múltiples fuentes para una misma información y son especialmente útiles por la fiabilidad que esto comporta. También para que el usuario acceda a la ubicación más próxima o para repartir la carga entre diferentes servidores.

3.8.1. Redes de distribución de contenidos

Han surgido empresas que se han dedicado a instalar máquinas en muchos lugares del mundo y algoritmos para decidir qué máquina es la más adecuada para atender las peticiones, según la ubicación del cliente y la carga de la red. Estas empresas venden este "servidor web distribuido" a distintos clientes que pagan por disponer de un sistema de servicio web de gran capacidad y que puede responder a demandas de contenidos extraordinariamente altas. Estos sistemas se denominan redes de distribución de contenidos[13].

(13)En inglés, *Content Delivery Networks* (CDN).

Akamai

Se pueden encontrar diferentes empresas que ofrecen este servicio de redes de distribución de contenidos. Akamai es la más importante. A fecha de febrero del 2010, la UOC utilizaba Akamai para proporcionar el contenido de su portal.

Una CDN es un *software* intermediario (*middleware*) entre proveedores de contenidos y clientes web. La CDN sirve contenidos desde distintos servidores repartidos por todo el planeta. La infraestructura de la CDN está compartida por distintos clientes de la CDN y las peticiones tienen en cuenta la situación de la Red para hacer que cada cliente web obtenga el contenido solicitado desde el servidor más eficiente de la CDN (habitualmente una combinación del más próximo, el menos cargado y el que tiene un camino con más ancho de banda con el cliente).

La CDN dispone de un gran número de puntos de servicio en multitud de proveedores de Internet y puede actuar sobre lo siguiente.

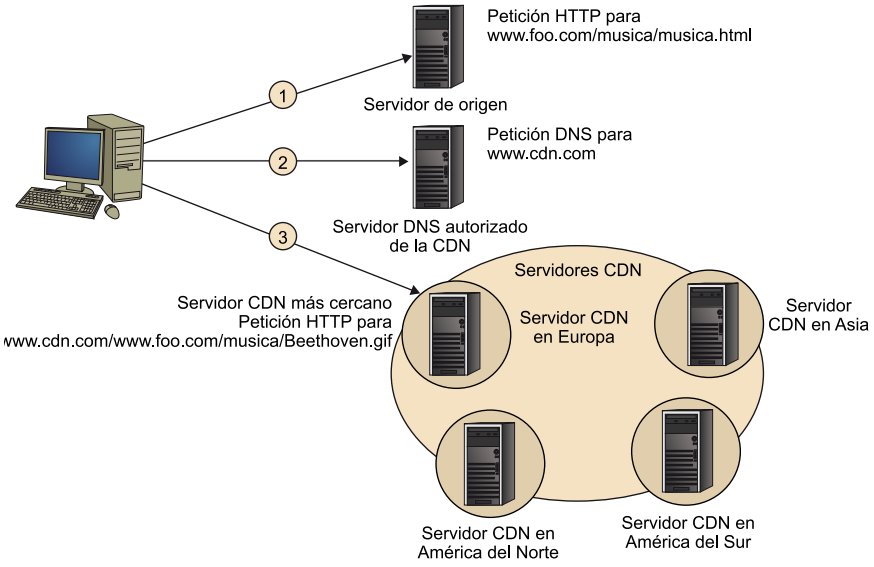
- El DNS: cuando se resuelve un nombre, el servidor DNS responde según la ubicación de la dirección IP del cliente.

- El servidor web: cuando se pide una página web, se reescriben los URL internos según la ubicación de la dirección IP del cliente.

- Cuando se pide un objeto a la dirección IP de un servidor de la CDN, el conmutador (*switch*) de acceso a un conjunto de distintos servidores *proxy-cache*, o réplicas, puede redirigir la conexión al servidor menos cargado del grupo (todo el conjunto responde bajo una misma dirección IP).

Ejemplo de funcionamiento de una CDN

La figura siguiente muestra el ejemplo de los posibles pasos que se pueden dar cuando se hace la petición de una página web con imágenes.



1) El cliente pide al servidor origen (www.foo.com) una página sobre música. El servidor informa al navegador web que la página pedida contiene varios objetos. El URL de estos objetos se modifica para que hagan referencia a la CDN. Por ejemplo, en lugar de contestar http://www.foo.com/musica/Beethoven.gif contesta http://www.cdn.com/www.foo.com/musica/Beethoven.gif.

2) El navegador pide al DNS que resuelva www.cdn.com. La petición llega hasta el DNS autorizado por este dominio, que es un DNS de la empresa propietaria de la CDN. La CDN responde con la dirección IP del nodo de la CDN más próximo a la ubicación del navegador web (que es el que hace la petición para los contenidos) al DNS de la CDN. Esta "proximidad" se calcula a partir de información que va recogiendo sobre las distancias de los nodos de la CDN al ISP.

3) Finalmente, el navegador web pide los objetos al nodo de la CDN que le han asignado.

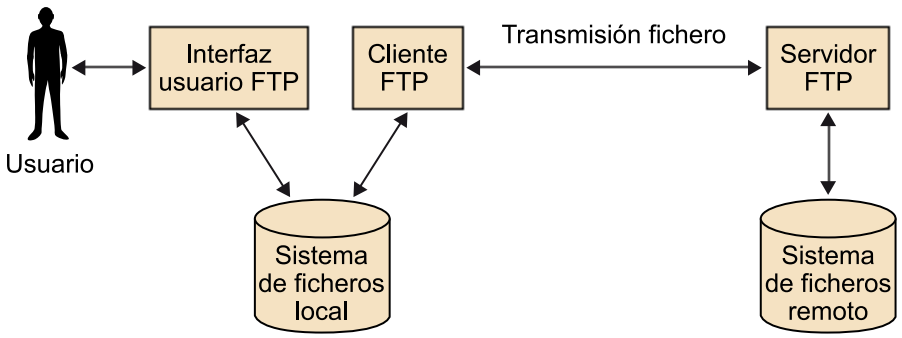
Las CDN no se utilizan sólo para distribuir contenidos web. También se usan para *streaming* de audio y vídeo, tanto almacenado como en directo.

Finalmente hay que decir que, para proporcionar su servicio, las CDN usan el DNS para unas funciones para las que no está diseñado. Las CDN utilizan el DNS para redirigir las peticiones de manera transparente, pero este mecanismo de redirección sobrecarga el DNS ya que hace que las peticiones dirigidas a la CDN no se puedan cachear.

4. Transferencia de ficheros

El protocolo de transmisión de ficheros FTP[14] permite transferir ficheros de un ordenador a otro. Funciona siguiendo el modelo cliente servidor sobre una conexión TCP/IP. En la figura siguiente, se puede ver el funcionamiento del FTP a grandes rasgos.

(14)Siglas en inglés de *File Transfer Protocol*



Para hacer la transferencia, el usuario debe conectarse al ordenador remoto proporcionando un usuario y un *password*. La transferencia se puede hacer tanto del ordenador del usuario al ordenador remoto como al revés.

El FTP utiliza dos conexiones entre el cliente y el servidor, una para los datos y otra para el control. La **conexión de control** usa el puerto TCP número 21 y se utiliza para enviar información de control, como el usuario y el *password*, comandos para cambiar de directorio o comandos para pedir y enviar ficheros. Esta conexión se mantiene abierta durante toda la sesión. La **conexión de datos** se utiliza para enviar los ficheros y se hace por el puerto TCP número 20. Se establece una conexión de datos para cada fichero. Una vez enviado, la conexión se cierra. En el caso de que se quiera enviar otro fichero, se abre una nueva conexión de datos.

El servidor FTP mantiene información de estado durante una sesión. El servidor tiene que asociar la conexión de control con una cuenta de usuario y debe saber en todo momento en qué directorio se encuentra el usuario, ya que este puede ir cambiando de directorio en el servidor remoto.

Los comandos se envían codificados en ASCII y finalizados con un *carriage return* y un *line feed*. Las respuestas también acaban con un *carriage return* y un *line feed*. Los comandos consisten en cuatro caracteres ASCII en mayúsculas,

algunos con argumentos opcionales. Algunos de los comandos más usuales (estos comandos son del protocolo, no los comandos que el usuario pone en el intérprete de comandos de la aplicación FTP cliente) son los siguientes.

- `USER nombreUsuario`: para enviar el nombre de usuario al servidor.
- `PASS clave`: para enviar la clave de acceso al servidor.
- `LIST`: para pedir al servidor que envíe la lista de ficheros del directorio remoto actual. La lista de ficheros se envía por una conexión de datos.
- `RETR fichero`: para obtener un fichero del directorio actual del ordenador remoto.
- `STOR fichero:` para enviar un fichero al directorio actual del ordenador remoto.

Cada comando enviado al ordenador remoto genera una respuesta de este. Las respuestas son números de tres dígitos seguidos de un mensaje opcional. Algunas respuestas habituales son:

- `331 Username Ok, password required`
- `125 Data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

Bibliografía

Acerca de las peticiones y las respuestas, encontraréis más información sobre los comandos y respuestas en el RFC 959.

4.1. Seguridad en la transferencia de ficheros

La especificación original del FTP es inherentemente insegura porque no tiene ningún método para transferir datos de manera cifrada. Esto quiere decir que en la mayoría de las configuraciones de red, el nombre de usuario, la clave de acceso, los comandos FTP y los ficheros transferidos se pueden capturar desde la misma red utilizando un *sniffer*. La solución habitual a este problema es usar SFTP[15] o FTPS[16].

[15]SFTP es la abreviatura de *SSH File Transfer Protocol* o *Secure File Transfer Protocol*.

[16]FTPS es la abreviatura de FTP sobre SSL.

El SFTP es un protocolo que proporciona acceso, transferencia y gestión de ficheros sobre un canal fiable. Normalmente se utiliza con el SSH, y el SSH es el que proporciona el canal fiable, aunque se podría utilizar con otros protocolos de seguridad. Por lo tanto, la seguridad no la proporciona directamente el protocolo SFTP, sino el SSH o el protocolo que se utilice para este propósito.

IETF

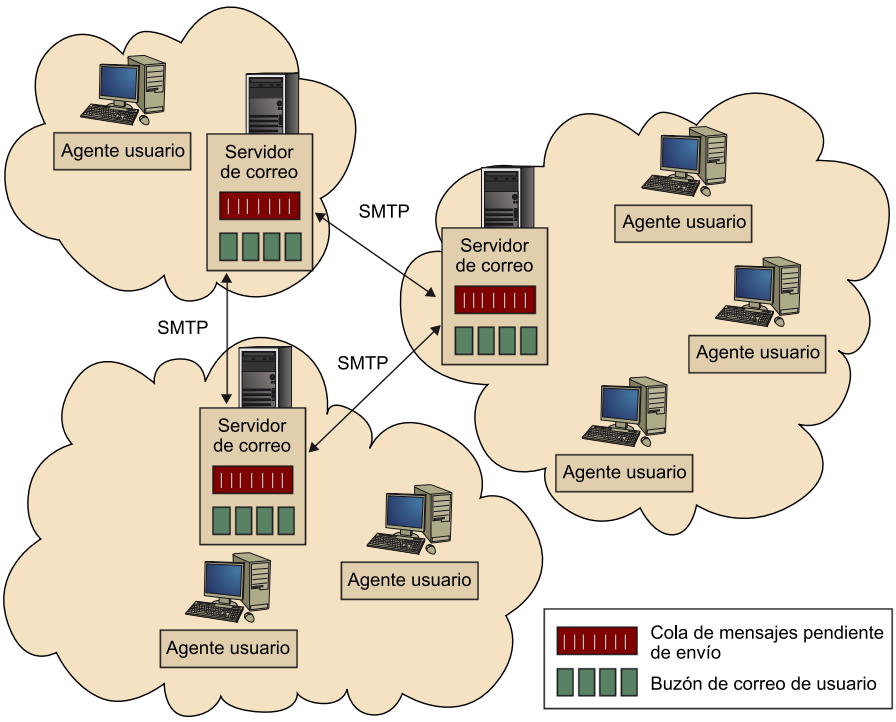
El SFTP fue diseñado por la Internet Engineering Task Force (IETF) como una ampliación del *Secure Shell Protocol* (SSH) versión 2.0.

5. Correo electrónico en Internet

El correo electrónico existe desde los principios de Internet. En los años sesenta los sistemas *mainframe* ya tenían formas de comunicación uno a uno tipo mensajería, y a lo largo de los años estos sistemas han ido evolucionando hasta llegar al nivel de sofisticación y potencia actuales. Hoy día, es una de las aplicaciones más importantes y usadas de Internet.

En la figura siguiente, se presenta una visión del sistema de correo en Internet. Cuando un usuario quiere enviar un mensaje, el agente usuario de mensajería que está utilizando envía el mensaje a un servidor de correo. Este –el servidor de correo origen– pregunta al DNS la dirección correspondiente al campo MX del dominio destino del mensaje y envía el mensaje al puerto 25 de este servidor de correo vía SMTP. La transmisión del mensaje se hace utilizando el protocolo TCP porque proporciona una transmisión fiable de los datos. Una vez el servidor destino ha aceptado el mensaje, lo almacena en el buzón del usuario destinatario para que más adelante el usuario destinatario –previa autenticación de este– lo lea vía un agente usuario de mensajería. Debemos destacar el hecho de que el protocolo SMTP normalmente no utiliza ningún servidor intermediario entre el servidor origen ni el destino. La comunicación se hace directamente aunque cada uno esté en una punta del mundo.

Visión de alto nivel del sistema de correo electrónico en Internet



En el caso de que el servidor origen no pueda enviar el mensaje al servidor destino por algún fallo, el servidor origen encola el mensaje y reintenta enviarlo más tarde. Los reintentos se hacen normalmente cada treinta minutos. Si después de algunos días no se puede entregar el mensaje, el servidor origen elimina el mensaje y notifica al remitente que no lo ha podido entregar.

5.1. SMTP

El protocolo ***Simple Mail Transfer Protocol*** (SMTP) está definido en el RFC 2.821 y es el estándar actual de transmisión de correo electrónico en Internet.

RFC del SMTP

El primer RFC que describe el SMTP data de 1982 (RFC 821), pero antes de esta fecha ya había versiones de lo que ahora es el SMTP.

El protocolo SMTP tiene muy buenas cualidades, como se puede deducir por el éxito que ha conseguido, pero arrastra algunos inconvenientes heredados del pasado. Por ejemplo, tanto el cuerpo de los mensajes como las cabeceras van codificados en caracteres ASCII de 7 bits. Esto hace que cualquier dato que no esté codificado en este formato deba codificarse en ASCII de 7 bits antes de enviarlo y volverse a descodificar una vez recibido.

Ejemplo de envío de un mensaje

A continuación, ponemos el ejemplo del envío de un mensaje `uoc.edu` a `upc.cat` utilizando SMTP. El ejemplo muestra los comandos que se intercambian una vez la conexión fiable (la conexión TCP) ha sido establecida. Con el objetivo de clarificar la comprensión del ejemplo, los comandos que envía el cliente (originador) se han prefijado con `C:` y los del servidor (receptor) con `S:`. También hemos numerado las líneas para poder referirnos a estos más fácilmente.

```
1 S: 220 upc.cat
2 C: HELO uoc.edu
3 S: 250 Hello uoc.edu please to meet you
4 C: MAIL FROM: <marques@uoc.edu>
5 S: 250 marques@uoc.edu ... Sender OK
6 C: RCPT TO: <puig@upc.cat>
7 S: 250 puig@upc.cat ... Recipient OK
8 C: DATA
9 S: 354 Enter mail, end with "." on a line by itself
10 C: Este es un mensaje de correo de ejemplo.
11 C: ¿Verdad que es sencillo?
12 C: .
13 S: 250 Message accepted for delivery
14 C: QUIT
15 S: 221 upc.cat closing connection
```

En este ejemplo, el cliente envía el mensaje `Este es un mensaje de correo de ejemplo. ¿Verdad que es sencillo?` desde el servidor de correo `uoc.edu` a `upc.cat`.

Como parte del diálogo, el cliente envía los comandos: `HELO`, `MAIL FROM`, `RCPT TO`, `DATA` y `QUIT`. El cliente envía el mensaje (líneas 10 a 11) después del comando DATA y lo acaba con una línea que sólo contiene un punto (línea 12: `CRLF.CFLR`), y que indica el final del mensaje enviado al servidor. El servidor contesta cada comando con un código de respuesta y un texto opcional explicativo (en inglés).

El protocolo SMTP usa conexiones persistentes. En el caso de que el servidor origen tenga más de un mensaje para enviar al mismo servidor destino, los mensajes se envían aprovechando la misma conexión TCP. El cliente empieza cada mensaje con un nuevo `MAIL FROM: uoc.edu`, acaba cada mensaje con un punto en una línea que sólo contiene un punto y envía el comando `QUIT` una vez ha acabado de enviar todos los mensajes.

Diálogo con el servidor SMTP

Si tenéis acceso a un servidor SMTP que no requiera autenticación, vosotros mismos podéis intentar dialogar directamente con el servidor SMTP. Para hacerlo, sólo tenéis que hacer: `telnet nombreServidor 25`, siendo `nombreServidor` el del servidor SMTP al cual os conectáis. Al hacer esto, estáis estableciendo una conexión TCP con el servidor SMTP. Podéis repetir los comandos del ejemplo prefijados con `C:` y veréis que se envía un mensaje de correo electrónico. (Intentad poner como recipiente una dirección de correo electrónico vuestra y recibiréis el mensaje.)

5.2. Formato de los mensajes

Tal y como sucede en las cartas postal que tienen un sobre y una carta, los mensajes de correo electrónico tienen dos partes.

- **Cabecera**: incluye la información general del mensaje. Sería equivalente al sobre de la carta postal y está formada por varios campos cabecera.

- **Cuerpo del mensaje**: contiene el mensaje en sí. Corresponde al contenido de la carta postal. Esta parte es opcional.

Formato de las cabeceras

El RFC 822 especifica el formato de las cabeceras, así como su semántica. Cada campo de la cabecera consta de un **nombre del campo** seguido del carácter `":/`, opcionalmente acompañado por un **cuerpo del campo**, y acaba con un `CRLF`.

Ejemplo de cabecera de un mensaje

El ejemplo del subapartado 5.1 no contiene ninguna cabecera. En la continuación repetimos el mismo mensaje, incluyendo la cabecera del mensaje. De este modo, primero encontramos la **cabecera** (líneas 10 a 13), seguida del **cuerpo** del mensaje (líneas 14 y 15). Tal y como se puede ver en la línea 13, el separador entre la cabecera y el cuerpo del mensaje es una línea en blanco (o sea, CRLF). Estos campos de la cabecera son diferentes de los comandos del SMTP ((`HELO`, `MAIL FROM`, `RCPT TO`, `DATA` y `QUIT`)), aunque puedan parecer similares.

```
1 S: 220 upc.cat
2 C: HELO uoc.edu
3 S: 250 Hello uoc.edu please to meet you
4 C: MAIL FROM: <marques@uoc.edu>
5 S: 250 marques@uoc.edu ... Sender OK
6 C: RCPT TO: <puig@upc.cat>
7 S: 250 puig@upc.cat ... Recipient OK
8 C: DATA
9 S: 354 Enter mail, end with "." on a line by itself
10 C: From: marques@uoc.edu
11 C: To: puig@upc.cat
12 C: Subject: Mensaje de ejemplo
13 C:
14 C: Este es un mensaje de correo de ejemplo.
15 C: ¿Verdad que es sencillo?
16 C: .
17 S: 250 Message accepted for delivery
18 C: QUIT
19 S: 221 upc.cat closing connection
```

5.2.1. Formato de mensajes MIME

Tal y como hemos dicho anteriormente, una limitación de la norma RFC 822 es que define un formato de mensaje y un contenido con una única parte de texto en ASCII de 7 bits. Se vio que este formato era muy pobre y que hacía falta algún método para superar sus limitaciones. El formato MIME[17] redefine el formato del mensaje para permitir, sin perder la compatibilidad con el formato definido por el RFC 822, dar apoyo a:

- Texto codificado con un juego de caracteres diferente del ASCII de 7 bits.
- Adjunciones que no sean texto.
- Cuerpos del mensaje con múltiples partes.
- Cabeceras codificadas con un juego de caracteres diferente del ASCII de 7 bits.

[17]MIME es la abreviatura de *Multipurpose Internet Mail Extensions*.

RFC del formato MIME

El formato de mensajes MIME está especificado en los RFC 2.045, 2.046, 2.047, 4.288, 4.289 y 2.049.

Para hacerlo, MIME define un conjunto de nuevos campos de cabecera `MIME-version`, `Content-ID`, `Content-Type`, `Content-Disposition` y `Content-Transfer-Encoding`.

En este módulo, nos centraremos en `Content-Type:` y `Content-Transfer-Encoding::`.

Content-Type

`Content-Type:` indica al agente de usuario de mensajería receptor del mensaje qué tipo de contenido contiene el cuerpo del mensaje, lo que permite que este tome la acción apropiada para cada contenido de mensaje. El tipo de contenido se especifica con un tipo y un subtipo.

Ejemplo

`Content-Type: image/JPEG`
indica que el cuerpo del mensaje contiene una imagen formateada en JPEG. De esta manera, el agente usuario de mensajería que reciba el mensaje puede hacer las acciones necesarias, como enviar el cuerpo del mensaje a un descompresor JPEG.

Formato: `Content-Type: type/subtype; parameters`

Algunos tipos	Algunos subtipos
text	plain, html
image	jpeg, gif
audio	basic, mp4, mpeg
video	mpeg, quicktime
application	msword, octet-stream, zip

La Internet Assigned Numbers Authority (IANA) mantiene la lista de tipos y subtipos. Podéis encontrarla en http://www.iana.org/assignments/media-types.

Content-Transfer-Encoding

Recordad que hemos dicho que los mensajes SMTP tienen que ir codificados en ASCII de 7 bits. La cabecera `Content-Transfer-Encoding:` indica que el cuerpo del mensaje ha sido codificado en ASCII y el tipo de codificación utilizado.

Ejemplo

`Content-Transfer-Encoding: base64`
indica que se ha utiliza la codificación base64 para codificar el cuerpo del mensaje. La codificación base64 codifica una secuencia arbitraria de *bytes* en una forma que satisface las reglas del ASCII de 7 bits, y por lo tanto, no confundirá el SMTP.

Otra técnica popular para codificar que también satisface el ASCII de 7 bits es la *quoted-printable*, que se acostumbra a utilizar para convertir texto codificado en ASCII de 8 bits (que puede contener caracteres no ingleses) en ASCII de 7 bits.

Ejemplo de *quoted-printable*

El siguiente es un ejemplo de *quoted-printable*:

```
Subject:  =?iso-8859-1?Q?Qu=E9?=  tal  =?iso-8859-1?Q?est=E1s=3F?=
```

que corresponde a "Subject: ¿Qué tal estás?", donde:

- `=` delimita la secuencia codificada con *quoted-printable*. En este caso, hay delimitadas dos secuencias: `¿Qué` y `estás?`
- `?` separa las partes.
- Codificado en ISO-8859-1.
- `Q`: codificación *quoted-printable*.
- `E9` es el código correspondiente a `é` en la codificación ISO-8859-1

Cuando volváis a recibir un mensaje (frecuentemente reenviado) con una secuencia como esta ya sabréis de dónde sale :-)

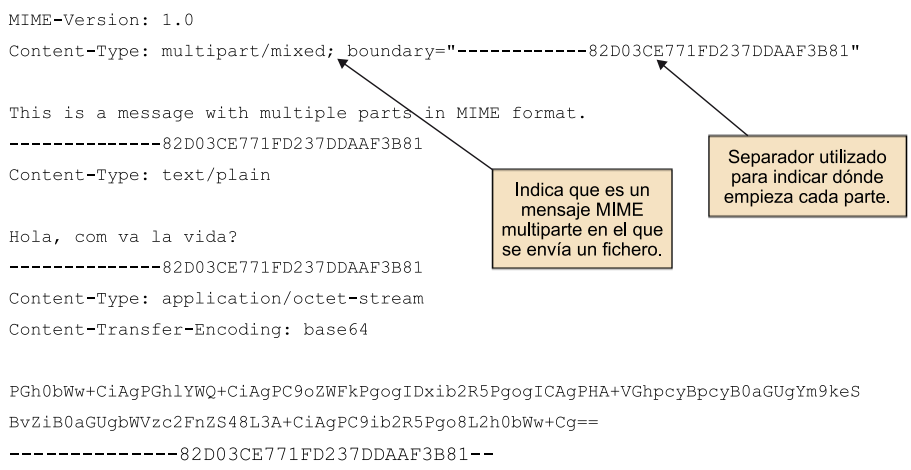
A continuación, pondremos un ejemplo que incluirá los diferentes aspectos vistos sobre el formato MIME:

```
From: marques@uoc.edu
To: puig@upc.cat
Subject: Foto del niño
MIME-version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

(datos            codificados              en             base64
.....................................................datos
codificados en base64)
```

La cabecera `MIME-version` indica la versión de MIME que se está utilizando.

La cabecera `Content-Type:` también se utiliza para que un mensaje pueda contener texto y ficheros adjuntos. En este caso, a la cabecera `Content-Type:` se le añade un indicador de inicio y final de cada parte (`boundary`).



S/MIME (*Secure/Multipurpose Internet Mail Extensions*) es un estándar para cifrado de clave pública y firma de correo electrónico encapsulado en MIME. S/MIME proporciona:

- Autenticación, integridad y no repudio (usando firma digital).
- Privacidad y seguridad de los datos (usando cifrado).

5.3. Protocolos para acceder a los buzones de correo

Los servidores SMTP guardan los mensajes recibidos en los buzones de correo de los destinatarios de los mensajes. Es posible acceder a estos mensajes conectándose al servidor SMTP y ejecutando un programa para leer el correo. La mayoría de los usuarios, sin embargo, leen el correo electrónico desde un ordenador personal, mediante un cliente de correo que les ofrece un conjunto amplio de funcionalidades y que incluye la posibilidad de ver mensajes multimedia y adjunciones.

5.3.1. POP3

POP3[18] es un protocolo del ámbito aplicación para que un agente de usuario de correo electrónico (el cliente) pueda obtener el correo de un servidor de correo remoto. El funcionamiento es muy sencillo. El cliente abre una conexión TCP con el servidor en el puerto 110. Una vez establecida esta conexión, el POP3 pasa por tres fases.

[18]El protocolo POP3 (*Post Office Protocol* versión 3) está especificado en el RFC 1939.

1) Autorización: el usuario envía un usuario y una clave de acceso (en claro) para autenticar al usuario.

2) Transacción: el agente de usuario baja mensajes. En esta fase, el agente de usuario puede marcar los mensajes para que se borren, eliminar marcas de borrar y obtener información sobre el buzón del usuario.

3) Actualización: ocurre después de que el cliente ha ejecutado el pedido `QUIT`, que finaliza la sesión POP3. En este momento, el servidor borra los mensajes que se habían marcado para ser borrados.

Los comandos son palabras clave, que pueden estar seguidas de argumentos, y que acaban con un salto de línea (CRLF). Tanto las palabras clave como los argumentos son caracteres ASCII imprimibles.

Las respuestas consisten en un indicador de estado (`+OK`) o (`-ERR`) que puede estar seguido por una información adicional.

- `+OK`: el servidor indica al cliente que el comando recibido era correcto
- `-ERR`: el servidor indica al cliente que había algo erróneo en el comando recibido.

Ejemplo de una sesión de autenticación

El siguiente es un ejemplo de una sesión de autenticación (una vez establecida la conexión al puerto TCP 110 del servidor de correo).

```
+OK POP3 server ready
USER marques
+OK
PASS uoc
+OK user successfully logged on
```

Conexión a un servidor de correo

Si tenéis acceso a un servidor de correo que no requiera utilizar una conexión segura, os podéis intentar conectar con `telnet servidorCorreo 110`.

El usuario puede configurar su agente de usuario –utilizando POP3– para que aplique la política "baja los mensajes y bórralos" o la política "baja los mensajes y deja copia de los mismos en el servidor".

Ejemplo de bajar y borrar los mensajes

El agente de usuario obtiene una lista de los mensajes (`LIST`), y los va bajando (`RETR`) y borrando (`DELE`) uno a uno. Por claridad, identificamos las líneas emitidas por el agente de usuario con **C**: y por el servidor de correo con **S**:

```
C:  LIST
S:  +OK 2 messages (320 octets)
S:  1 120
S:  2 200
S:  .
C:  RETR 1
S:  +OK 120 octets
S:  < el servidor de POP3 envía el mensaje 1>
S:  .
C:  DELE 1
S:  +OK message 1 deleted
C:  RETR 2
S:  +OK 200 octets
S:  < el servidor de POP3 envía el mensaje 2>
S:  .
C:  DELE 2
S:  +OK message 2 deleted
C:  QUIT
S:  +OK POP3 server signing off
```

Un problema del modo "baja los mensajes y bórralos" es que el correo se baja en un ordenador. Si el usuario posteriormente quiere acceder al correo desde otro ordenador no lo podrá hacer, ya que el servidor ya no contendrá los mensajes. Esto es problemático para usuarios que acostumbran a leer el correo desde diferentes ordenadores, por ejemplo el ordenador de casa, un ordenador portátil, el trabajo, etc.

El servidor POP3 no guarda estado entre sesiones.

5.3.2. IMAP

El protocolo IMAP[19] es un protocolo de acceso a correo como el POP3, pero tiene muchas más funcionalidades. También es más complejo de implementar, tanto en la parte cliente como en la parte del servidor.

(19)El IMAP está especificado en el RFC 3.501.

Los servidores IMAP[20] asocian cada mensaje a una carpeta. Cuando un mensaje llega al servidor, este se asocia a la carpeta INBOX. Posteriormente, el receptor puede mover este mensaje a otra carpeta creada por el mismo usuario. También puede leerlo, borrarlo, etc. El protocolo IMAP proporciona comandos para permitir a los usuarios crear carpetas y para mover mensajes de una carpeta a otra. El IMAP también proporciona comandos que permiten a los usuarios hacer buscas de mensajes que satisfagan ciertos criterios en las carpetas remotas. De todo esto, ya se ve que los servidores IMAP deben mantener información entre sesiones IMAP: los nombres de las carpetas, qué mensajes están asociados a cada carpeta, etc.

(20)IMAP es la abreviatura de *Internet Mail Access Protocol*.

Otra característica interesante del IMAP es que tiene comandos que permiten al agente de usuario obtener partes de un mensaje: puede obtener sólo las cabeceras del mensaje, o sólo una parte de un mensaje multiparte (por ejemplo, el texto del mensaje sin el fichero adjunto). Esto es especialmente útil cuando se utiliza una conexión con poco ancho de banda.

5.3.3. Web

Una manera muy popular de acceder al correo es mediante un navegador web. En este caso, el agente de usuario es el navegador web y el usuario se comunica con su buzón vía HTTP. De esta manera, cuando se quiere leer un mensaje el navegador lo obtiene del servidor de correo usando el protocolo HTTP, y cuando se quiere enviar un mensaje, el navegador lo envía al servidor de correo usando también el protocolo HTTP.

6. Aplicaciones de igual a igual para la compartición de ficheros

La arquitectura de igual a igual es una manera de construir aplicaciones que persigue la utilización de los recursos informáticos y el ancho de banda que aportan los usuarios de la aplicación aunque estos estén conectados de manera intermitente.

Bibliografía recomendada

En el artículo de **Lua y otros** (2005). "A survey and comparison of peer-to-peer overlay network schemes". *IEEE Communications Surveys & Tutorials* (vol. 7, núm. 2) encontraréis un resumen de cómo funcionan las redes superpuestas de igual a igual más populares, así como una comparativa entre estas. También encontraréis más detalle de los sistemas explicados en este apartado.

Las aplicaciones de igual a igual se han popularizado de la mano de las aplicaciones de compartición de ficheros. La primera fue Napster, pero la han ido siguiendo otras aplicaciones. En este material presentaremos varias, que nos tienen que ayudar a entender cómo funcionan estos tipos de aplicaciones. Cada una de estas aplicaciones utiliza su propia organización interna y sus propios protocolos.

Los nodos que forman el sistema o aplicación de igual a igual se organizan formando una red superpuesta (*overlay network*, en inglés) que funciona sobre la red física que conecta los nodos. Hay diferentes maneras de organizar esta red superpuesta, las cuales se pueden dividir en dos categorías: estructuradas y no estructuradas.

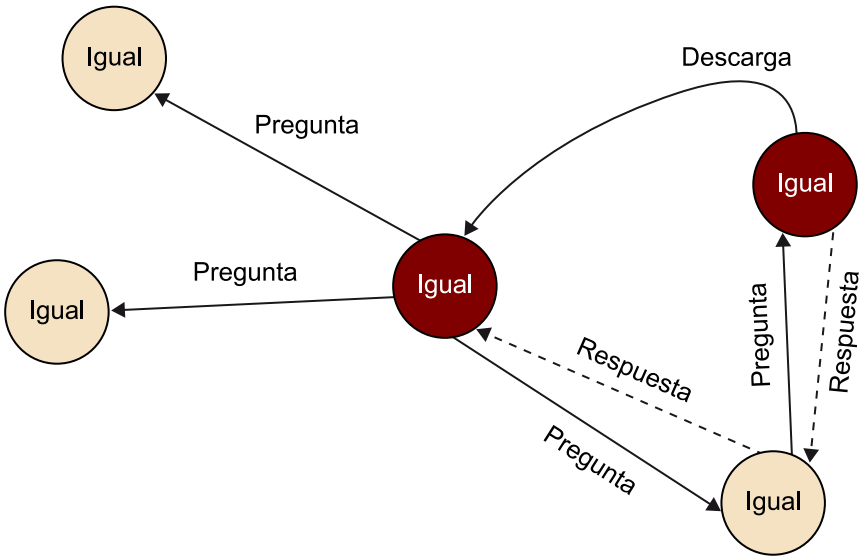
6.1. Redes superpuestas no estructuradas

Un sistema de igual a igual que utilice una red superpuesta del tipo no estructurado es un sistema que está compuesto de iguales que se conectan a la red sin conocer su topología. Estos sistemas usan mecanismos de inundación para enviar consultas mediante la red superpuesta. Cuando un igual recibe la pregunta, envía al igual que la ha originado una lista de todo el contenido que encaja con la pregunta. Aunque las técnicas basadas en la inundación van bien para localizar objetos altamente reproducidos y son resistentes ante las conexiones y desconexiones de los nodos, no tienen un comportamiento muy bueno cuando se hacen buscas de objetos poco reproducidos. De esta manera, las redes superpuestas no estructuradas tienen fundamentalmente un problema: cuando deben gestionar un ritmo elevado de consultas o cuando hay crecimientos repentinos del tamaño del sistema, se sobrecargan y tienen problemas de escalabilidad.

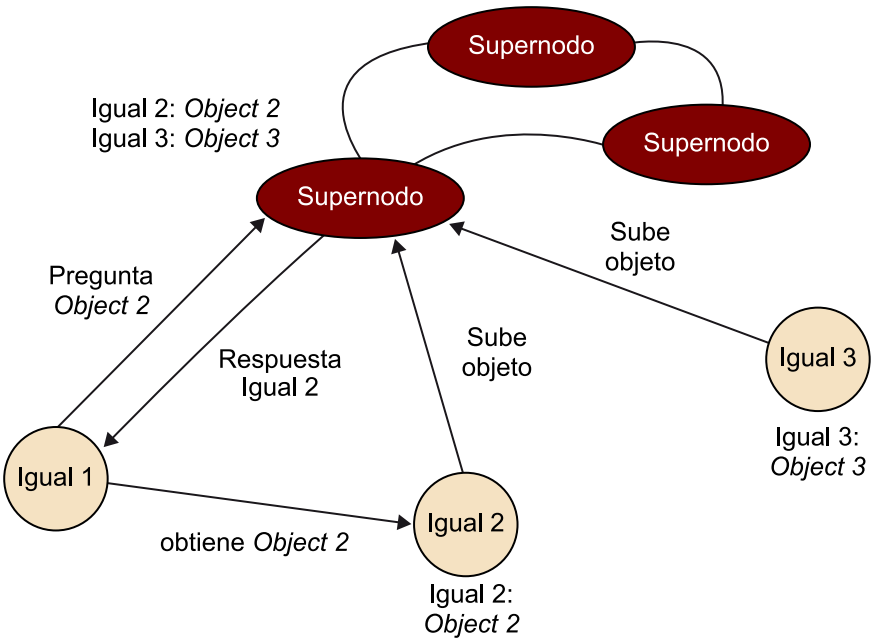
A pesar de que el sistema de direccionamiento basado en claves que utilizan los sistemas de igual a igual estructurados –veremos este tipo de sistemas en el próximo subapartado– pueda localizar de manera eficiente objetos y, además, sea escalable, sufre sobrecargas significativamente mayores que los sistemas no estructurados por contenidos populares. Por este motivo, los sistemas descentralizados no estructurados se usan más.

Algunos ejemplos de sistemas no estructurados son Gnutella, FastTrack/KaZaA, BitTorrent y Overnet/eDonkey2000.

Gnutella: protocolo totalmente descentralizado para hacer buscas sobre una topología de iguales totalmente plana. Ha sido (y todavía lo es) muy usado. Para localizar un objeto, un igual pregunta a sus vecinos. Estos inundan a sus vecinos y así hasta un cierto radio. Este mecanismo es extremadamente resistente ante entradas y salidas de nodos, pero los mecanismos actuales de busca no son escalables y generan cargas no esperadas en el sistema. La figura siguiente muestra un ejemplo de busca.



KaZaA: es un sistema de ficheros descentralizado en el que los nodos se agrupan en torno a superiguales (*super-peers* en inglés) para hacer las buscas más eficientes, tal y como se muestra en la figura siguiente. La comunicación entre los iguales en KaZaA se hace utilizando el protocolo Fast Track, que es un protocolo propietario. Los superiguales son iguales del sistema que se han elegido para que mantengan metainformación que hará las buscas más eficientes. En el momento de una busca, el igual pregunta al superigual al que está conectado. Este, de manera similar a lo que hace Gnutella, hace un *broadcast* a los otros superiguales.



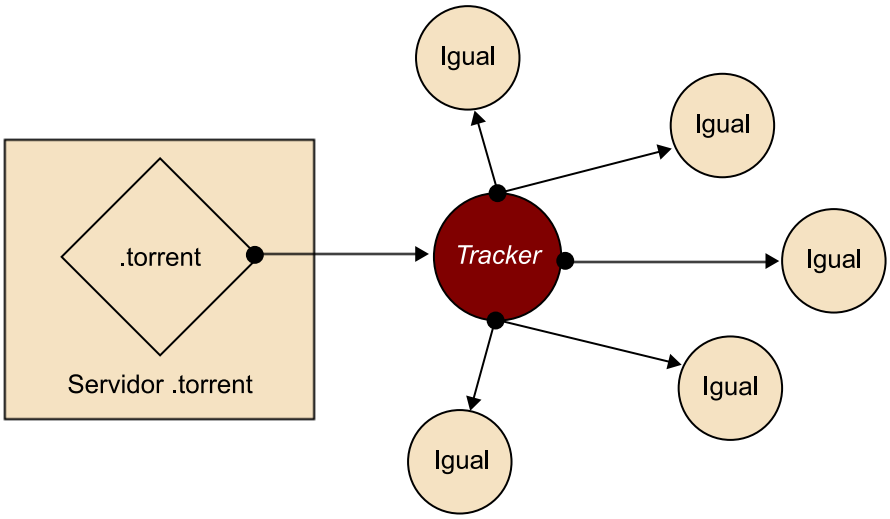
Los iguales se conectan a un superigual. Las consultas se encaminan hacia los superiguales. Las bajadas se hacen entre iguales.

BitTorrent: es un sistema de igual a igual para distribuir grandes volúmenes de datos sin que el originador de la información tenga que soportar todo el coste de los recursos necesarios para servir el contenido. Este tipo de soluciones son útiles para distribuir contenidos que son muy populares. BitTorrent utiliza servidores para gestionar las bajadas. Estos servidores almacenan un fichero que contiene información sobre el fichero: longitud, nombre, información de resumen (*hashing information*, en inglés) y el URL del *tracker*. El *tracker* (podéis ver la figura siguiente) conoce todos los iguales que tienen el fichero (tanto de manera total como parcial) y hace que los iguales se conecten unos con otros para bajar o subir los ficheros. Cuando un nodo quiere bajar un fichero envía un mensaje al *tracker*, que le contesta con una lista aleatoria de nodos que están bajando el mismo fichero. BitTorrent parte los ficheros en trozos (de 256 kB) para saber qué tiene cada uno de los iguales. Cada igual que está bajando el fichero anuncia a sus iguales los trozos que tiene. El protocolo proporciona mecanismos para penalizar a los usuarios que obtienen información sin proporcionarla. De esta manera, a la hora de subir información, un igual elegirá otro igual del que haya recibido datos.

Bibliografía complementaria

Para más información sobre el funcionamiento de BitTorrent, podéis consultar el artículo siguiente:

B. Cohen (2003, junio). "Incentives Build Robustness in BitTorrent". *Proc. First Workshop the Economics of Peer-to-Peer Systems*. Berkeley, CA: University of Berkeley.



eDonkey: es un sistema de igual a igual híbrido organizado en dos niveles para el almacenamiento de información. Está formado por clientes y servidores. Los servidores actúan como concentradores para los clientes y permiten a los usuarios localizar los ficheros que hay en la Red. Esta arquitectura proporciona bajada concurrente de un fichero desde distintas ubicaciones, uso de funciones resumen (*hash*, en inglés) para la detección de ficheros corruptos, compartición parcial de ficheros mientras se bajan y métodos expresivos para hacer buscas de ficheros. Para que un nodo se pueda conectar al sistema, es necesario que conozca un igual que actúe como servidor. En el proceso de conexión, el cliente proporciona al servidor la información sobre los ficheros que comparte. Cuando un cliente busca un fichero, los servidores proporcionan las ubicaciones de los ficheros. De esta manera, los clientes pueden bajar los ficheros directamente de las ubicaciones indicadas.

eMule puede funcionar tanto sobre la red eDonkey como sobre Kad. Kad es una implementación de Kademlia, una red superpuesta estructurada (en el próximo apartado se explica cómo funcionan estos tipos de sistemas).

6.2. Redes superpuestas estructuradas

La topología de la red superpuesta sobre la cual se construyen estos sistemas está fuertemente controlada y el contenido no va a cualquier lugar, sino a uno determinado que hace que las consultas sean más eficientes. Estos sistemas utilizan tablas de *hash* distribuidas (*distributed hash tables* o DHT, en inglés) como sustratos, en los cuales la ubicación de los objetos (o valores) se hace de manera determinista. Los sistemas que se basan en DHT tienen la propiedad de que asignan los identificadores de nodos de una manera consistente a los iguales dentro de un espacio con muchos identificadores. A los objetos de datos se les asigna un identificador, que se denomina *clave*, elegido dentro del mismo espacio de nombres. El protocolo de la red superpuesta mapea las claves en un único nodo entre los conectados. Estas redes superpuestas soportan el almacenamiento y la recuperación de pares {clave,valor} en la red superpuesta.

Este tipo de sistemas usan un sistema de direccionamiento estructurado, manteniendo tanto la descentralización de Gnutella como la eficiencia y garantía de localizar los resultados de Napster.

Por este motivo, cada igual mantiene una tabla de direccionamiento pequeña. Los mensajes se direccionan de una manera progresiva hacia los iguales a través de caminos de superposición. Cada nodo envía el mensaje al nodo de su tabla de direccionamiento que tiene un identificador más próximo a la clave en el espacio de identificadores. Los diferentes sistemas basados en DHT tienen distintos esquemas de organización para los objetos de datos y su espacio de claves y estrategias de direccionamiento. En teoría, los sistemas basados en DHT garantizan que, como media, se puede localizar cualquier objeto en O(log N) saltos en la red superpuesta, donde N es el número de iguales en el sistema. El camino entre dos nodos en la red física puede ser muy diferente del camino en la red superpuesta DHT. Esto puede provocar que la latencia en las buscas en un sistema de igual a igual basado en una red DHT sea bastante grande y pueda afectar negativamente al rendimiento de la aplicación que funcione por encima.

Ejemplos de redes superpuestas estructuradas

Can, Chord, Tapestry, Pastry, Kademlia, DKS o Viceroy son ejemplos de redes superpuestas estructuradas.

Podéis encontrar más información de estos sistemas en:

CAN

S. Ratnasamy y otros (2001). "A Scalable Content Addressable Network". *Proc. ACM SIGCOMM* (págs. 161-72).

Chord

I. Stoica; R. Morris y otros (2003). "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications". *IEEE/ACM Trans. Net.* (vol. 11, núm. 1, págs. 17-32).

Tapestry

B. Y. Zhao y otros (2004, enero). "Tapestry: A Resilient Global-Scale Overlay for Service Deployment". *IEEE JSAC* (vol. 22, núm. 1, págs. 41-53).

Pastry

A. Rowstron; P. Druschel (2001). "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems". *Proc. Middleware*.

Kademlia

P. Maymounkov; D. Mazieres (2002, febrero). "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". *Proc. IPTPS* (págs. 53-65). Cambridge, MA: EE. UU.

DKS

DKS(N,k,f). *A Familiy of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications*.

Viceroy

D. Malkhi; M. Naor; D. Ratajczak (2002, julio). "Viceroy: A Scalable and Dynamic Emulation of the Butterfly". *Proc. ACM PODC 2002* (págs. 183-92). Monterey, CA: EE. UU.

eMule tiene una versión que funciona sobre la red Kad, que es una implementación de Kadmelia. Kad mejora la localización de ficheros en la Red y la hace más resistente a ataques a los servidores centrales.

7. Mensajería instantánea

La mensajería instantánea es una forma de comunicación textual en tiempo real entre dos o más personas a través de Internet o algún otro tipo de red. La mensajería instantánea se diferencia del correo electrónico en que el usuario percibe sincronismo en la comunicación, aunque muchos de los sistemas de mensajería instantánea también permiten enviar mensajes a personas que en aquel momento no están conectadas. En este caso, el destinatario recibirá el mensaje cuando se vuelva a conectar al sistema.

La mensajería instantánea permite una comunicación efectiva y eficiente y consigue una recepción inmediata de la confirmación o la respuesta. Normalmente, las respuestas se pueden guardar y consultar con posterioridad.

Frecuentemente, combinado con la mensajería instantánea o en torno a la misma también encontramos otras funcionalidades que la hacen aún más completa y popular, como cámaras de vídeo o la posibilidad de hablar directamente mediante Internet. En este apartado, sin embargo, nos centraremos en la mensajería instantánea textual.

7.1. XMPP

El XMPP[21] es un protocolo libre de mensajería instantánea de especificaciones abiertas basado en el XML, y que ha sido estandarizado por la IETF.

[21]XMPP es la abreviatura de *Extensible Messaging Presence Protocol*. El XMPP antes se conocía como *jabber*.

Como se puede ver en la siguiente figura, el XMPP[22] utiliza una arquitectura cliente-servidor descentralizada: de manera similar a como lo hace el SMTP, no hay un servidor central que coordine la mensajería. Los clientes no se comunican directamente unos con otros, lo hacen mediante los servidores.

[22]El puerto estándar para el XMPP es el 5222.

Arquitecturas centralizadas

Otros sistemas de mensajería instantánea como Windows Live Messenger o AOL Instant Messenger utilizan arquitecturas centralizadas.

De manera similar al correo electrónico, cada usuario tiene una dirección única formada por dos campos: un nombre de usuario (único para cada servidor) seguido de la dirección DNS del servidor que hospeda al usuario, separados por el símbolo @, como nombre@dominio.com.

Los pasos para que un mensaje enviado por el cliente 7 (cliente7@server3) de la figura anterior llegue al cliente 3 (cliente3@server1) son los siguientes:

1) El cliente 7 envía el mensaje a su servidor (en este caso, el servidor 3).

a) El servidor 3 pide al DNS la dirección IP correspondiente al servidor 1.

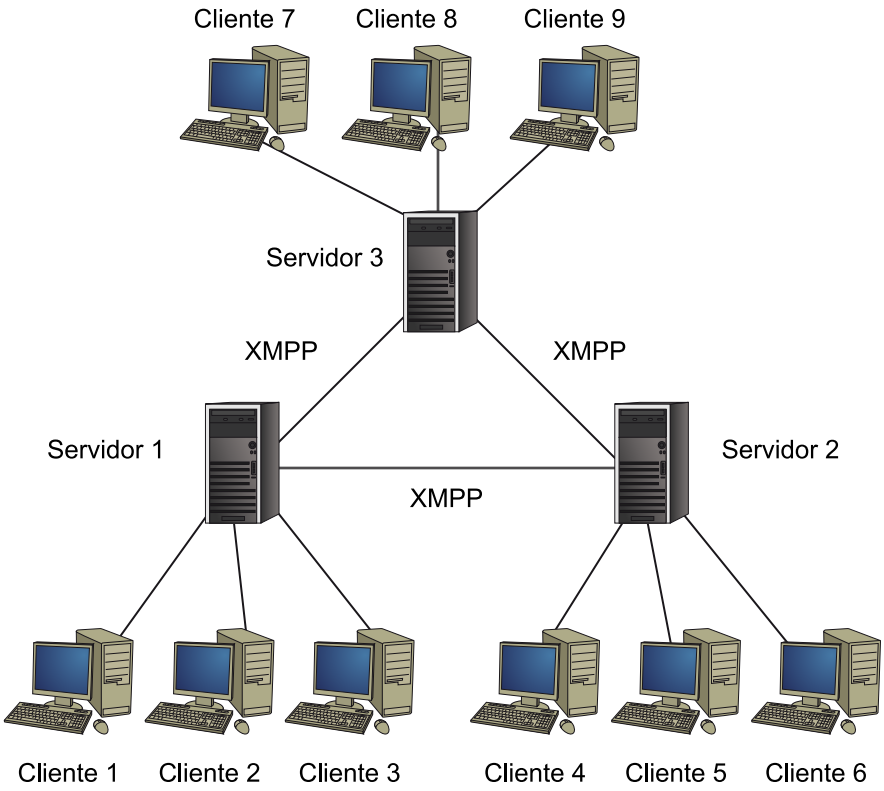
b) Si el servidor 3 bloquea la comunicación hacia el servidor 1, el mensaje se descarta.

2) El servidor 3 obra una conexión con el servidor 1.

a) Si el servidor 1 bloquea los mensajes procedentes del servidor 3, el mensaje se descarta.

b) Si el cliente 3 no está conectado en estos momentos, el servidor 1 almacena el mensaje para entregarlo más adelante.

3) El servidor 1 entrega el mensaje al cliente 3.



Los puntos fuertes del protocolo XMPP son los siguientes.

- **Descentralización**: cualquiera puede tener un servidor XMPP. No hay un servidor centralizado que coordine la mensajería.

- **Estándares libres**: la Internet Engineering Task Force tiene el XMPP aprobado como un estándar de mensajería instantánea y presencia (RFC 3.920 y RFC 3.921).

- **Seguridad**: los servidores XMPP se pueden aislar de la red XMPP pública (por ejemplo, en la intranet de una compañía), y el núcleo de XMPP incluye seguridad (vía SASL y TLS).

- **Flexibilidad**: por encima del XMPP, se pueden añadir funcionalidades a medida.

- Al ser un protocolo abierto, cada usuario puede utilizar un cliente XMPP diferente, siempre y cuando cumpla las especificaciones del XMPP.

Los puntos débiles del protocolo XMPP son los siguientes.

- ***Overhead* de la información de presencia**: en torno al 70% del tráfico entre servidores XMPP es información de presencia, cuyo 60% es redundante. La información de presencia permite saber qué amigos o contactos hay conectados y, por lo tanto, es posible enviar mensajes instantáneos.

- **La inclusión de datos binarios es ineficiente**: XMPP se codifica como un único documento XML. Para incluir datos binarios, estos se deben codificar en base64. Cuando hay que enviar cantidades considerables de datos binarios (por ejemplo, envío de ficheros), lo mejor es hacer el envío de manera externa al XMPP y utilizar mensajes XMPP para coordinarse.

SASL

El *Simple Authentication and Security Layer* (SASL) es un *framework* para la autenticación y la seguridad de los datos en protocolos de Internet. Separa los mecanismos de autenticación de los protocolos de aplicación haciendo posible –en teoría– que cualquier mecanismo de autenticación soportado por el SASL se pueda utilizar en cualquier protocolo de aplicación. Está especificado en el RFC 4.422.

Jingle

Jingle es una ampliación del XMPP para iniciar y gestionar sesiones multimedia entre dos entidades XMPP, de manera que sean interoperables con estándares existentes de Internet.

8. Telnet y Secure Shell: acceso a ordenadores remotos

El protocolo Telnet[23] se usa en Internet o en redes de área local para proporcionar una comunicación bidireccional interactiva en el acceso a ordenadores remotos. Está basado en el protocolo de transporte TCP. En una comunicación Telnet, normalmente se sigue el modelo cliente-servidor, es decir, el sistema usuario establece una conexión con el sistema proveedor, que está esperando peticiones de conexión en un puerto determinado. Se puede utilizar cualquier número de puerto para las conexiones y, de hecho, hay muchas aplicaciones que utilizan el protocolo Telnet para la comunicación, cada una con su propio número.

[23]La especificación de Telnet se publicó en el año 1983 en el estándar RFC 854.

La aplicación básica, sin embargo, es establecer una sesión de trabajo interactiva con el sistema servidor. En este caso, el número de puerto utilizado es el 23. Esta sesión de trabajo en el ordenador remoto se hace vía un terminal virtual. Para resolver el problema del control del terminal en la aplicación de sesiones interactivas, en el protocolo Telnet se utiliza el concepto de terminal virtual de red o NVT. Un NVT es un terminal virtual con una funcionalidad muy básica, definida en la misma especificación del protocolo.

Terminal virtual de red

Un terminal virtual de red, en inglés *Network Virtual Terminal* (NVT), es un dispositivo imaginario por el cual se definen unas funciones de control canónicas, de manera que se puede establecer una correspondencia entre estas funciones y las de cada tipo de terminal real.

Cuando se establece la conexión entre el cliente y el servidor, inicialmente se supone que la comunicación se produce entre dos NVT. Esto quiere decir que tanto el sistema cliente como el sistema servidor deben mapear sus características en las de un NVT y suponer que en el otro extremo de la conexión hay otro NVT. En el modo de operación normal, cada terminal acepta datos del usuario y los envía mediante la conexión establecida al otro terminal, y acepta también los datos que llegan por la conexión y los presenta al usuario. La comunicación se hace en *bytes* de 8 bits.

8.1. Seguridad del protocolo Telnet

Telnet no es un protocolo seguro, por las razones siguientes:

- No cifra ningún dato enviado a través de la conexión (ni las claves de acceso), y esto permite espiar la comunicación y poder utilizar la clave de acceso –así como cualquier otra información enviada– más adelante para usos malintencionados.

- La mayoría de las implementaciones de Telnet no proporcionan autenticación que asegure que la comunicación se lleva a cabo entre los dos ordenadores deseados y no está interceptada por el camino.

8.2. Secure Shell

La solución a la falta de seguridad del protocolo Telnet ha sido el protocolo Secure Shell (SSH).

Puerto TCP 22

El puerto estándar para contactar a un servidor SSH es el puerto TCP 22.

El SSH proporciona la funcionalidad de Telnet con los añadidos siguientes:

a) Permite el cifrado para evitar que sea posible interceptar los datos que se envían.

b) Permite la autenticación con clave pública para asegurar que el ordenador remoto es quien dice ser. El SSH tiene la debilidad de que el usuario debe confiar en la primera sesión con un ordenador cuando todavía no ha adquirido la clave pública del servidor.

9. Aplicaciones multimedia en red

Las aplicaciones multimedia en red tienen unos requerimientos muy diferentes a los de las aplicaciones tradicionales en la red Internet (correo electrónico, transferencia de ficheros, etc.).

Si tenemos en cuenta dos de los requerimientos más relevantes que tienen las aplicaciones de red, como son la tolerancia a pérdidas de datos y las consideraciones temporales, nos damos cuenta de que estos son especialmente importantes para las aplicaciones multimedia en red.

Ved también

Para una descripción completa de los requerimientos de las aplicaciones en red, podéis ver en este mismo módulo el subapartado 1.4, "Requerimientos de las aplicaciones".

Con respecto a la pérdida de datos, se tiene que decir que las aplicaciones multimedia son muy tolerantes: las pérdidas ocasionales de datos lo único que hacen es que haya un salto en la imagen o en el sonido; sin embargo, si se continúa recibiendo información, se puede continuar normalmente la comunicación. Por el contrario, se trata de aplicaciones muy sensibles al retraso, lo cual hace que las consideraciones temporales se deban tener muy en cuenta. Así pues, a lo largo de los subapartados siguientes veremos que los paquetes que llegan con un retraso de más de unos centenares de milisegundos no sirven para nada en una aplicación multimedia y se pueden descartar (un sonido que se ha emitido hace dos segundos con respecto a lo que se está escuchando o una imagen que ya ha pasado no hay que mostrarlos al usuario).

Estas características de alta tolerancia a pérdidas y mucha sensibilidad al retraso hacen que las aplicaciones multimedia sean muy diferentes a las aplicaciones tradicionales, en las que es mucho más importante que lleguen todos los datos (si falta alguna parte, la información recibida será inservible) que el tiempo que estos datos tarden en llegar al destinatario (que puede hacer que la experiencia de usuario no sea muy buena, pero si al final recibe todos los datos, habrá cumplido los requerimientos). En los próximos subapartados veremos algunos ejemplos de aplicaciones multimedia y qué nos podemos encontrar actualmente en Internet para el soporte de estas aplicaciones.

9.1. Ejemplos de aplicaciones multimedia

Hay muchos tipos de aplicaciones multimedia actualmente en la red Internet. En este subapartado describiremos brevemente tres de los grandes tipos de aplicaciones multimedia que nos podemos encontrar: *streaming* de audio / vídeo almacenados, *streaming* en directo de audio / vídeo y audio / vídeo en tiempo real interactivo. Ninguna de estas aplicaciones cubre el caso en el que nos bajamos un contenido y después lo vemos. Este caso tiene que ver con la transferencia de ficheros que se puede hacer con protocolos como HTTP (*Hypertext Transfer Protocol*) y FTP (*File Transfer Protocol*).

9.1.1. *Streaming* de audio y vídeo almacenados

En este tipo de aplicaciones, los clientes piden contenidos de audio y vídeo comprimidos almacenados en servidores. Estos contenidos pueden ser, por ejemplo, programas de televisión, vídeos caseros, canciones, sinfonías o programas de radio. Esta clase de aplicaciones tienen tres características distintivas.

- **Contenidos almacenados**: el contenido multimedia está ya grabado y se almacena en el servidor. Como resultado, el usuario puede detener la reproducción, rebobinar o indexar el contenido. El tiempo de respuesta de un contenido de este tipo puede ir de uno a diez segundos para ser aceptable.

- ***Streaming***: cuando un usuario recibe un contenido con la técnica del *streaming*, la reproducción empieza poco después de haber hecho la petición del contenido. De esta manera, el usuario ve una parte del contenido, mientras que el resto se va recibiendo a medida que se reproduce. En este caso, el tiempo de respuesta es más bajo. Con esta técnica evitamos el retraso que podemos tener si debemos bajarnos todo el contenido.

- **Reproducción continua**: una vez la reproducción del contenido multimedia empieza, tiene que transcurrir tal y como se grabó originalmente. Esto hace que haya unos fuertes requerimientos de retraso en la entrega de los datos en este tipo de aplicación. La aplicación de usuario debe recibir a tiempo los datos del servidor para poder hacer la reproducción de manera correcta. Aunque esta aplicación tiene unos requerimientos de retraso importantes, no son tan fuertes como para el *streaming* en directo o las aplicaciones de tiempo real, que veremos a continuación.

9.2. *Streaming* en directo de audio y vídeo

Este tipo de aplicaciones funcionan como el *broadcast* tradicional de radio y televisión –un emisor transmite a muchos receptores–, sólo que se hace por Internet. Con este tipo de *streaming* se pueden recibir emisiones de radio y televisión en directo desde cualquier punto del mundo.

Puesto que en el *streaming* de audio y vídeo la información no está almacenada, el usuario no puede tirar adelante la reproducción del contenido recibido. Lo que sí permiten algunas aplicaciones es parar la reproducción e incluso ir hacia atrás (aunque esto último sólo es posible si se ha almacenado el contenido en el disco del usuario mientras se iba reproduciendo).

Aunque para enviar este tipo de *streams* se podrían utilizar conexiones *multicast* (un emisor transmite a diferentes receptores en una única conexión), la verdad es que se hacen múltiples conexiones *unicast* (una conexión para cada receptor).

Se requiere una reproducción continuada, pero los requerimientos de interactividad no son tan críticos como las aplicaciones interactivas en tiempo real y pueden aceptarse retrasos en el inicio de la reproducción de hasta diez segundos.

9.2.1. Audio y vídeo en tiempo real interactivo

Este tipo de aplicaciones permiten que los usuarios interactúen en tiempo real. Actualmente, hay bastantes aplicaciones de este tipo que permiten que los usuarios hagan audio o videoconferencias mediante Internet. Windows Messenger, Google Talk o Skype son algunos ejemplos de aplicaciones que permiten establecer conexiones de audio y/o vídeo. Algunas de estas aplicaciones también permiten llevar a cabo llamadas telefónicas a teléfonos fijos o móviles (haciendo un pequeño pago), como es el caso de Skype.

Direcciones web recomendadas

Aplicaciones de audio y vídeo interactivo:

Windows Messenger: http://www.microsoft.com/windows/messenger/features.asp

Google Talk: http://www.google.com/talk/intl/se/

Skype: http://www.skype.com/intl/es/

El retraso permitido en estas aplicaciones no debe ser mayor que unos centenares de milisegundos. Así pues, retrasos de más de 400 milisegundos harían que una comunicación de voz fuera prácticamente imposible.

9.3. Multimedia en Internet

IP es un protocolo de tipo *best effort*, es decir, que intenta entregar los paquetes tan pronto como puede, pero sin dar ninguna garantía. Los protocolos de transporte TCP/UDP funcionan sobre IP, y por lo tanto tampoco pueden dar garantías de cuándo llegará un determinado paquete a su destino. Por este motivo, el envío de datos multimedia por Internet es una tarea difícil de que ha tenido hasta ahora un éxito limitado.

En las aplicaciones de *streaming* de audio y vídeo almacenados, retrasos de entre cinco y diez segundos pueden ser aceptables. Así pues, si no se produce un pico de tráfico o los paquetes no deben pasar por enlaces congestionados, podremos disfrutar del *streaming* de manera satisfactoria.

Por otra parte, el uso de las aplicaciones de *streaming* interactivo de audio y vídeo hasta ahora no ha sido tan satisfactorio a causa de las restricciones en el retraso de los paquetes y en el denominado *paquet jitter*. El *paquet jitter* es la variabilidad que pueden tener los retrasos de paquetes enviados entre un origen y un destino dentro del mismo *stream*. Puesto que el audio y el vídeo se deben reproducir con la misma temporización con la cual se grabaron, el

jitter se tiene que eliminar antes de hacer la reproducción de los datos. El reproductor debe almacenar los paquetes por un corto periodo de tiempo para eliminar el *jitter* antes de reproducir el *stream*.

Con el fin de eliminar el *jitter*, habitualmente se combinan tres mecanismos:

1) Añadir un número de secuencia a cada paquete de datos enviado. El emisor incrementa este número en cada paquete enviado.

2) Añadir un *timestamp* a cada paquete. El emisor pone en cada paquete el momento en el que se generó.

3) Retrasar la reproducción del paquete en el receptor. El retraso en la reproducción tiene que ser lo bastante grande como para garantizar que los paquetes se han recibido antes del momento de su reproducción. Este retraso puede ser fijo o adaptativo a lo largo de la sesión. Los paquetes que no llegan antes del momento de su reproducción se consideran perdidos y se descartan.

Con estos tres mecanismos, se pueden utilizar dos técnicas de reproducción dependientes del retraso: **retraso fijo** o **retraso adaptativo** en el momento de la reproducción de los datos. El retraso fijo consiste en reproducir el paquete de datos exactamente x milisegundos después de la creación del paquete de datos. Así pues, si el *timestamp* del paquete era t, su reproducción se tiene que hacer en el instante $t + x$, siempre que el paquete haya llegado antes de este momento. El valor de x depende de la aplicación, pero para telefonía por Internet es posible soportar hasta 400 milisegundos. Si es menos, se produce el riesgo de que no lleguen. El retraso adaptativo consiste en adaptarse a las condiciones de pérdida de paquetes y retrasos que tengamos durante la comunicación. La manera de hacer esto es calcular el retraso de la red y la variación del retraso e ir ajustando el retraso en la reproducción de acuerdo con estos datos.

El *streaming* interactivo funciona bien si se dispone de un buen ancho de banda, en el que el retraso y el *jitter* son pequeños.

Las aplicaciones multimedia funcionarían mejor en la red Internet si existiera la posibilidad de reservar una parte del ancho de banda para el envío de este tipo de información. Sin embargo, esto no parece que tenga que suceder, al menos de momento. Por esto, es necesario utilizar algún otro tipo de técnicas, como por ejemplo usar UDP en lugar de TCP para el envío de los datos, introducir retrasos en el envío de los datos o, si se trata de *streaming* de contenidos almacenados, enviar datos de antemano (utilizando técnicas de *buffering*) cuando la conexión lo permite. Incluso se puede enviar información redundante para mitigar los efectos de la pérdida de paquetes.

Puesto que para este tipo de datos no es muy adecuado reenviar los paquetes, lo que se hace es utilizar otras técnicas de recuperación de errores, como son *forward error correction* (FEC) e *interleaving*.

El FEC consiste en enviar datos redundantes al *stream* original. De esta manera, se pueden reconstruir versiones aproximadas de los paquetes originales perdidos. Una manera de hacer FEC es enviar un paquete redundante cada *n* paquetes. Esto hace que si se pierde un paquete de estos *n*, se pueda recuperar. Si se pierden más, entonces esto ya no es posible. De todos modos, si se envían muchos paquetes redundantes hay más retraso. La otra técnica para hacer FEC es enviar los datos redundantes en un *stream* con calidad más baja. Si se pierde un paquete del *stream* original, entonces se toma el paquete del *stream* de baja calidad y se reproduce en el momento que le toca. De esta manera, aunque se mezclen paquetes de alta y baja calidad, no se pierde ningún paquete y la experiencia del usuario es bastante buena.

El *interleaving* consiste en que el emisor envía los paquetes en un orden diferente del de su reproducción, para minimizar las pérdidas en un grupo de paquetes. De esta manera se reducen las pérdidas, pero no es una técnica adecuada para audio y vídeo interactivo, por el retraso en la recepción de los datos; sin embargo, sí funcionaría bien para audio y vídeo almacenados. La ventaja de esta técnica es que no se necesita más ancho de banda, ya que el *stream* enviado es el mismo.

9.3.1. Compresión de audio y vídeo

Para enviar datos multimedia (audio y vídeo) por Internet es necesario digitalizar y comprimir estos datos. La razón por la cual hay que digitalizar los datos es muy sencilla: las redes de ordenadores transmiten bits; así pues, toda la información que se transmite tiene que estar representada con bits. La compresión es importante porque el audio y el vídeo sin comprimir ocupan mucho espacio, con el consumo de ancho de banda que esto implica. Eliminar la redundancia en las señales de audio y vídeo reduce en varios órdenes de magnitud el ancho de banda que se necesita para transmitir la información.

El ancho de banda necesario y la compresión

Una imagen de 1.024 píxeles –en la que cada píxel se presenta con 24 bits, 8 para los colores rojo, azul y verde– necesita 3 MB sin compresión.

Si se aplica una tasa de compresión de 1:10, tendremos que esta misma imagen necesita 300 KB para ser representada.

Suponiendo que tuviéramos que enviar las dos imágenes por un canal de 64 kbps, en el primer caso tardaría 7 minutos en enviarse, mientras que en el segundo el tiempo de transmisión se reduce en un factor de 10.

Compresión de audio

La compresión de tipo **PCM (*pulse code modulation*)** se basa en la recogida de muestras de audio a una frecuencia determinada. El valor de cada muestra es un número real arbitrario. El valor de cada muestra se redondea a un determinado valor finito (tenemos un número de valores finitos para las muestras). Cada uno de estos valores se representa con un número finito de bits, que depende del número de valores que pueden tomar las muestras. Por ejemplo, si se tienen 256 posibles valores de muestras, utilizaríamos un *byte* para representarlas.

Para el caso de la PCM, se toman 8.000 muestras por segundo y cada muestra se representa con 8 bits. Esto nos da una señal digital con una tasa de 64.000 bits por segundo. La señal digital puede descodificarse y convertirse de nuevo en una señal analógica, pero esta señal será diferente de la señal analógica original como consecuencia del muestreo. Si se recogen más muestras y se toman más valores posibles para estas muestras, la señal analógica descodificada será mucho más parecida a la señal original.

Ejemplos en el uso de la PCM

Algunos ejemplos en el uso de la PCM son la codificación de voz con 8.000 muestras/segundo y 8 bits/muestra, lo que da una tasa de 64 kbps. El CD de audio también utiliza la PCM, con 44.100 muestras/segundo y 16 bits/muestra. Esto da una tasa de 705,6 kbps para canales mono y 1,411 Mbps para estéreo.

Aunque la velocidad de las conexiones en Internet ha mejorado (recordemos que un módem proporcionaba una velocidad máxima de 56 kbps), la compresión de audio y vídeo es todavía muy importante. Así pues, podemos encontrar algunos ejemplos de compresión de voz con tasas de bits más bajas como GSM (13 kbps) o G.729 (8 kbps).

Para la compresión de sonido con calidad casi de CD, la técnica más popular es el estándar de compresión MPEG 1 Layer 3, más conocido como MP3. Las tasas de compresión de MP3 suelen ser 96 kbps, 128 kbps y 160 kbps, con poca degradación del sonido.

Compresión de vídeo

El vídeo es una sucesión de imágenes, transmitidas a una tasa constante, como 24 o 30 imágenes por segundo. Una imagen sin comprimir es una sucesión de píxeles, en la que cada píxel se representa con un número de bits que indican color y luminosidad. Hay dos tipos de redundancia en los vídeos que se pueden aprovechar para comprimir: redundancia espacial y redundancia temporal. La redundancia espacial consiste en tener repeticiones dentro de la misma imagen y la temporal, en redundancia entre imágenes consecutivas.

Para vídeo, los estándares de compresión MPEG son también los más populares. Estos incluyen MPEG 1 para la compresión con calidad de CD de vídeo (1,5 Mbps), MPEG 2 para calidad de DVD (3-6 Mbps) y MPEG 4 para compresión orientada a objetos. Las técnicas de compresión MPEG se basan en el hecho de explotar la redundancia temporal entre imágenes además de la compresión

que tiene en cuenta la redundancia espacial que ya proporciona JPEG. Otros estándares de compresión de vídeo son H.261 o Quicktime (este último es un formato propietario de Apple).

9.3.2. Formatos de audio y vídeo

Existen muchos formatos diferentes de audio y vídeo, y aquí presentaremos los más conocidos y utilizados. Los formatos de vídeo más conocidos para trabajar en la red Internet son Quicktime Movie (Apple), Real Media (Real Networks), Windows Media (Microsoft), Audio/Video Interleaved (Microsoft), MPEG (mpg) y Flash Video (Adobe). A continuación, hacemos un resumen de sus características principales.

Bibliografía recomendada

Más información sobre los formatos de audio y vídeo:

J. Niederst (2006). *Web Design in a Nutshell* (3.ª ed.). Sebastopol: O'Reilly.

- **Quicktime Movie**: originalmente estaba pensado como formato de vídeo, pero ahora se puede utilizar para contener cualquier tipo de medio (imágenes, audio, vídeo, Flash, etc.). Es un formato propietario de Apple. Es posible hacer *streaming* de este formato utilizando algún servidor de *streaming* dedicado. Se puede simular también el *streaming* sobre HTTP si se utiliza FastStart, que hace que el vídeo se empiece a reproducir al mismo tiempo que se está bajando.

- **Real Media**: es uno de los formatos más utilizados para hacer *streaming*. Se trata de un formato propietario de la compañía Real Networks y hay que tener un sistema Real Media para poder hacer el *streaming*. También permite hacer *seudostreaming* vía HTTP.

- **Windows Media**: se trata del formato de vídeo creado por Microsoft. Hay dos formatos, Windows Media Video (.wmv) y Advanced Streaming Format (.asf). Se necesita un servidor de tipo Windows Media Server para crear este tipo de ficheros. Permite hacer *streaming* y transmisiones de vídeo en directo.

- **Audio/Video Interleaved**: es un formato de vídeo creado también por Microsoft para su Video For Windows (VFW), la arquitectura multimedia de Windows 95. Ha sido sustituido por el formato Windows Media. Con este formato no se puede hacer *streaming*, hay que bajar el contenido y después reproducirlo.

- **MPEG**: formato estándar definido por el Moving Picture Experts Group (MPEG). Soporta tres tipos de información: audio, vídeo y *streaming* (que significa que soporta audio y vídeo sincronizados). Este formato ofrece una alta compresión con pocas pérdidas.

- **Flash Video**: es un formato de vídeo creado por Flash (.flv) que permite hacer *streaming*. Funciona con la aplicación Flash Player, lo cual hace que no se necesite un reproductor dedicado para poder ver los vídeos. Además,

permite las mismas características de interactividad que ahora ofrecen las animaciones hechas con Flash (ficheros .swf).

Los formatos de audio más conocidos para trabajar en la red Internet son WAV/AIFF, MP3, Quicktime Audio (Apple), MIDI, Real Media Audio (Real Networks), Windows Media Audio (Microsoft) y Flash (Adobe). A continuación hacemos un resumen de sus características principales.

- **WAV/AIFF**: estos dos formatos tienen características muy similares. El *Waveform Audio Format* (.wav) fue originalmente diseñado para los sistemas operativos Windows, mientras que el *Audio Interchange File Format* (.aiff) fue diseñado para sistemas Apple. Ahora ya no se utilizan tanto, ya que hay otros formatos que ofrecen más compresión con una calidad parecida, como por ejemplo el MP3. No permiten hacer *streaming*, pero son los formatos que se utilizan como base para otros formatos (como RealAudio). Si se comprimen, pierden mucha calidad de sonido.

- **MP3**: se trata del formato más popular en la red Internet. Ofrece una muy buena calidad de sonido, al mismo tiempo que permite hacer una alta compresión de datos. Sigue el estándar MPEG-1 Nivel-III y se basa en comprimir la información partiendo de la percepción auditiva de las personas. Se puede hacer *streaming* de este formato y también se puede bajar con HTTP o FTP.

- **Quicktime Audio**: aunque el formato Quicktime está pensado para vídeo, también permite incluir audio. Este formato permite hacer *streaming* y *seudostreaming* con HTTP, además de poder bajarse.

- **MIDI**: quiere decir *Musical Instrument Digital Interface* y se trata de un tipo diferente de formato de audio de los vistos hasta ahora. Se diseñó inicialmente como estándar para que los instrumentos musicales electrónicos se pudieran comunicar entre los mismos. No contiene información de audio, sino órdenes de cómo se tendrían que tocar las diferentes notas, con instrucciones sobre longitud y volumen. Los ficheros MIDI ocupan muy poco y son ideales para conexiones con un ancho de banda bajo. No se puede hacer *streaming* de este formato.

- **Real Media Audio**: fue uno de los primeros formatos que permitían hacer *streaming* de audio mediante la Red. Necesita un servidor dedicado, el RealServer, que permite negociar el ancho de banda y la transmisión RTSP. Se puede hacer *seudostreaming* de este formato con un servidor HTTP, si hay un tráfico limitado.

- **Windows Media Audio**: es el formato de audio para *streaming* creado por Microsoft. Hay dos opciones: Windows Media Audio (.wma), que no permite *streaming*, y Active Streaming File (.asf) para *streaming*. El *codec* para

este formato es propietario y se necesita un servidor dedicado para hacer *streaming*.

- **Flash**: se trata de un formato que permite añadir sonidos de fondo a las páginas HTML y, además, da características de interactividad y animación. En este caso, no hay retrasos en la reproducción y, una vez bajado el fichero Flash, la interactividad y el sonido están permanentemente disponibles.

10. *Streaming* de audio y vídeo almacenados

Las aplicaciones de *streaming* de audio y vídeo se han popularizado en los últimos años por diferentes motivos. Por una parte, los usuarios tienen cada vez más capacidad de almacenamiento y también redes con mayor ancho de banda, que permiten recibir la información multimedia pedida en un corto espacio de tiempo.

Sin embargo, para recibir los datos de *streaming* hay que tener una aplicación dedicada (lo que se conoce como *media player* o reproductor), que sea capaz de ofrecer las características siguientes.

- **Descompresión**: los datos audio y vídeo están casi siempre comprimidos para ahorrar espacio de disco y ancho de banda. El reproductor deberá descomprimir el audio y el vídeo a medida que se reciben.

- **Eliminación del *jitter***: el *paquete jitter* es la variabilidad del retraso que hay entre origen y destino dentro de un mismo *stream*. Puesto que el audio y el vídeo se tienen que reproducir con la misma temporización con la cual se grabaron, el *jitter se* debe eliminar antes de hacer la reproducción. El receptor almacenará los paquetes por un corto periodo de tiempo para que sea posible eliminar el *jitter* antes de reproducir el *stream*.

- **Corrección de errores**: a causa de los errores imprevisibles en la red, una parte de los paquetes se podría perder. Si se pierden demasiados paquetes, entonces la calidad del *stream* recibido es inaceptable. Algunas técnicas para recuperar los datos perdidos son las siguientes.
  - Reconstruir los paquetes perdidos mediante el envío de paquetes redundantes.
  - Hacer que el cliente pida de manera explícita los paquetes perdidos.
  - Interpolar los datos perdidos a partir de los datos recibidos.

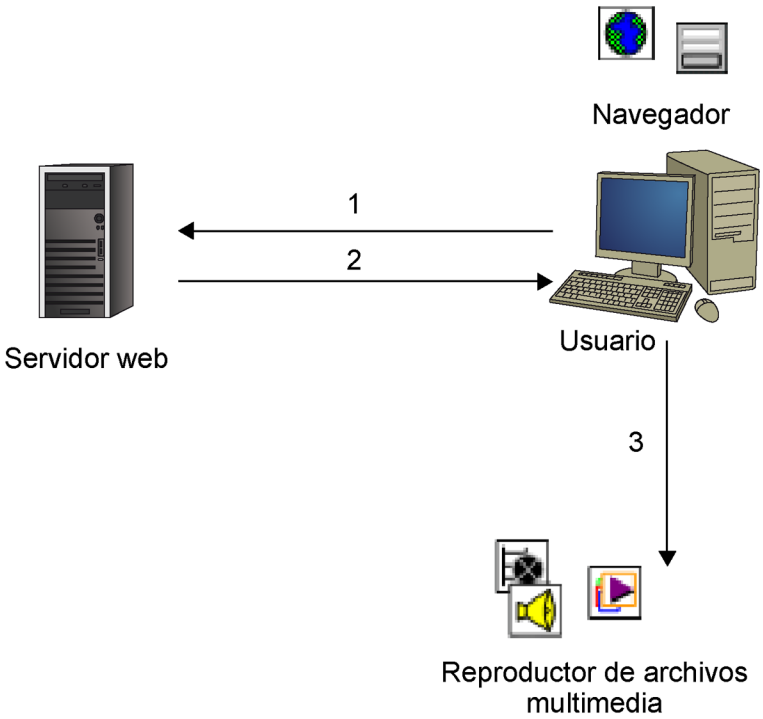
En los siguientes subapartados veremos dos maneras de hacer *streaming* de audio y vídeo almacenados. La primera consiste en utilizar un servidor de web tradicional, en el que el audio y el vídeo están almacenados en el sistema de ficheros. La otra manera de recibir un *stream* es el uso de un protocolo de *streaming* específico, como es el *Real Time Streaming Protocol* (RTSP).

10.1. Acceso vía servidor de web

Desde el punto de vista de un servidor de web, los ficheros de audio y vídeo son objetos accesibles desde el servidor como cualquier otro tipo de fichero, como una página HTML o una imagen.

Así pues, el usuario utilizaría su navegador para acceder al fichero de audio o vídeo. Este acceso establecería una conexión TCP con el servidor y, una vez establecida esta conexión, pediría el recurso (fichero de audio o vídeo) con un mensaje de petición HTTP con la correspondiente orden HTTP (podría ser una orden GET). El servidor genera un mensaje de respuesta, en el que estará encapsulado el fichero pedido. La recepción de este fichero hará que se abra el programa reproductor y, cuando este tenga bastantes datos, empezará a reproducir el fichero recibido.

La figura siguiente muestra de manera muy simple cómo se produciría esta comunicación.



1) El usuario pide con su navegador un archivo multimedia (por ejemplo, haciendo clic en un enlace web). Esta petición genera un mensaje de petición HTTP.

2) El servidor busca el recurso en su sistema de ficheros y lo envía al navegador del cliente. Esta respuesta viaja en un mensaje de respuesta HTTP.

3) El navegador guarda los datos en el disco y estos datos son reproducidos por un reproductor de archivos multimedia, que deberá descomprimir los datos, eliminar los retrasos y solucionar los errores de datos que se hayan podido producir.

En este caso, encontramos el problema de que hasta que el navegador no recibe el objeto multimedia entero, no lo puede pasar al reproductor. Si el archivo es muy grande, el retraso será inaceptable. Por este motivo, lo que se hace normalmente es que el servidor de web se comunique directamente con el reproductor y le envíe los datos. De esta manera, el reproductor recibe los datos y cuando tiene suficientes para reproducir el contenido, empieza a hacerlo, al mismo tiempo que continúa recibiendo datos.

La manera de hacer que el servidor web se conecte directamente con el reproductor es sustituyendo el recurso multimedia por un fichero de metadatos, que contiene la información de dónde se encuentra el recurso multimedia del que se tiene que hacer *streaming* e incluso del tipo de codificación. De esta manera, el navegador recibe un fichero de metadatos, que pasa al reproductor y este se encarga de conectar con la dirección contenida en el fichero de metadatos. Así pues, el inicio de la reproducción puede empezar mucho antes de haber recibido todo el fichero multimedia.

10.2. *Real Time Streaming Protocol* (RTSP)

El protocolo RTSP[24] establece y controla tanto uno como varios *streams* sincronizados de datos multimedia, como pueden ser el audio y el vídeo. No hace el envío de los datos, aunque el envío de información de control en medio de la transmisión de datos es posible. Lo que hace el RTSP es de control remoto a través de la Red para los servidores de datos multimedia.

[24]El RFC que define el RTSP es el 2.326.

En el RTSP no hay conexiones, lo que tenemos son sesiones mantenidas por el servidor. Cada sesión tiene su identificador. Una sesión RTSP no está vinculada a una conexión de nivel de transporte. Esto quiere decir que, durante una sesión RTSP, se pueden abrir y cerrar tantas sesiones de transporte como sea necesario. También se puede utilizar UDP como protocolo de transporte, un protocolo sin conexión.

Los *streams* controlados por el RTSP pueden utilizar RTP, pero el funcionamiento del RTSP no depende del mecanismo de control utilizado para enviar los datos multimedia. El protocolo RTSP es muy parecido al protocolo HTTP/1.1 (la versión 1.1 del protocolo HTTP), lo cual hace que se puedan añadir mecanismos de extensión para el HTTP que también se pueden aplicar al RTSP. Sin embargo, el RTSP es diferente del HTTP en un buen número de aspectos importantes.

Ved también

Sobre el RTP, podéis ver más adelante el subapartado 11.1 de este módulo didáctico.

- El RTSP introduce métodos nuevos y tiene un identificador de protocolo propio (rtsp://).

- Un servidor RTSP tiene que mantener el estado en la mayoría de los casos, justo al contrario de lo que hace el HTTP.

- Tanto el cliente como el servidor RTSP pueden hacer peticiones.

- Los datos los transporta un protocolo diferente del RTSP.

- El URI de petición del RTSP siempre contiene el URI absoluto. En el HTTP, el URI de petición sólo lleva la ruta absoluta al fichero y el nombre del servidor va en una cabecera aparte.

El protocolo soporta las operaciones siguientes.

a) Recuperación de datos del servidor de contenidos multimedia: el cliente puede pedir una descripción de una presentación vía HTTP o cualquier otro método. Si la presentación se está emitiendo en *multicast*, la descripción de esta contiene su dirección *multicast* y los puertos que se utilizarán. Si la presentación se envía en *unicast* sólo a este cliente, el cliente da el destino por motivos de seguridad.

b) Invitación de un servidor de datos a una conferencia: un servidor de datos puede ser invitado a unirse a una conferencia existente, tanto para reproducir datos en la presentación o para grabar todos o una parte de los datos transmitidos en la presentación. Este modo es muy útil para aplicaciones de enseñanza distribuida. Muchos participantes pueden tomar parte en las mismas, pulsando los botones de control remoto.

c) Añadir contenido a una presentación existente: es muy útil especialmente para el caso de presentaciones en directo, ya que el servidor le puede decir al cliente que hay nuevos datos disponibles.

Un fichero de descripción de una presentación sería un ejemplo de lo que en la distribución de contenidos mediante un servidor de web hemos denominado un *fichero de metadatos*. Este fichero contiene una descripción de los *streams* que forman parte de la presentación, incluyendo el lenguaje de codificación y otros parámetros que ayudan al usuario a elegir la mejor combinación de datos. Cada contenido (por ejemplo, el audio y el vídeo se podrían transmitir en *streams* separados) tiene su propia URL, que apunta a un determinado contenido dentro de un determinado servidor. El fichero también describe los métodos de transporte que se soportan. Además de los datos del contenido, la descripción de la dirección destino de la red y el puerto se tiene que determinar. Hay diferentes modos de operación.

1) *Unicast*: los datos se transmiten al origen de la petición RTSP, en el número de puerto seleccionado por el cliente.

2) *Multicast* (el servidor elige dirección): el servidor selecciona la dirección *multicast* y el puerto. Este caso es el típico para transmisiones de datos multimedia en directo.

3) *Multicast* (el cliente elige dirección): si el servidor participa en una conferencia *multicast* existente, la dirección *multicast*, el puerto y la clave de cifrado vienen dados por la descripción de la conferencia.

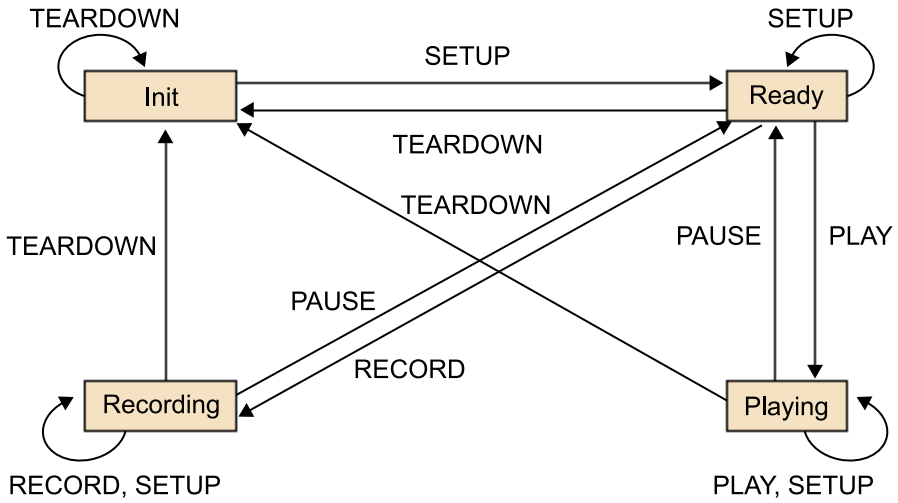
10.2.1. Los estados del RTSP

El RTSP controla *streams* que se pueden enviar por un protocolo separado, independiente del protocolo de control. También puede pasar que el RTSP funcione sobre conexiones TCP, mientras que los datos se envían por UDP. Así pues, la transmisión de datos continúa ocurriendo aunque no se envíen datos de control RTSP. Otra posibilidad es que un *stream* de datos esté controlado por peticiones RTSP y que estas peticiones viajen en diferentes conexiones TCP. Por todas estas razones, es necesario mantener el estado de la sesión RTSP, con el objetivo de correlacionar las peticiones que se refieren al mismo *stream*.

10.2.2. Diagrama de estados del cliente

El cliente puede estar en cuatro estados posibles: *Init*, *Ready*, *Playing* y *Recording*. El estado *Init* indica que el cliente ha enviado una orden SETUP y espera respuesta. El estado *Ready* indica que, o se ha recibido una respuesta afirmativa a SETUP, o se ha recibido una confirmación al envío de la orden PAUSE estando en *Playing* o *Recording*. Los estados *Playing* y *Recording* indican que se ha recibido una confirmación afirmativa a las órdenes PLAY y RECORD, de manera respectiva.

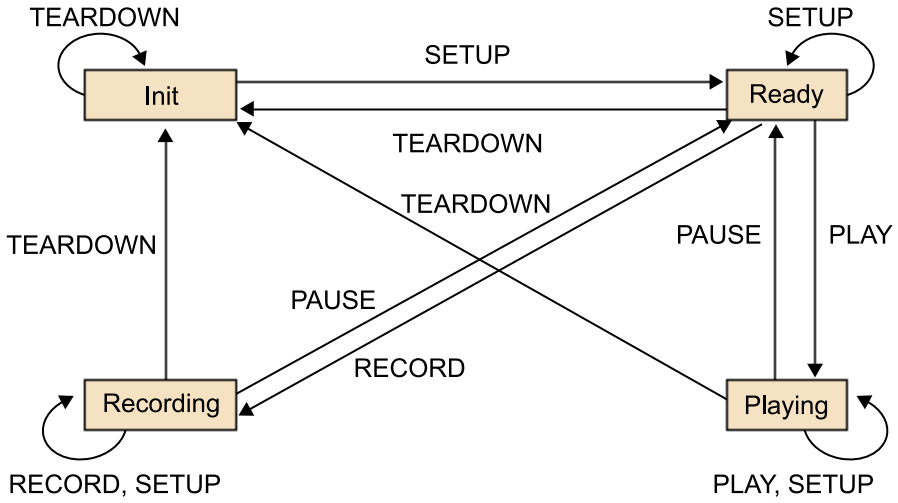
La figura siguiente muestra los estados por los cuales ha pasado el cliente, como respuesta a las órdenes que envía. Las transiciones se producen cuando el cliente recibe una confirmación positiva a la orden enviada (la orden se indica en la flecha correspondiente).



10.2.3. Diagrama de estados del servidor

El servidor puede estar en cuatro estados posibles: *Init*, *Ready*, *Playing* y *Recording*. El estado *Init* indica que el servidor está a la espera de recibir una orden SETUP correcta. Es el estado inicial. El estado *Ready* indica que el último SETUP recibido fue correcto y se envió la confirmación correspondiente, o que en los estados *Playing* y *Recording* la orden PAUSE se recibió y se confirmó. El estado *Playing* indica que se ha recibido la orden PLAY y se confirmó y que se están enviando los datos al cliente. El estado *Recording* indica que el servidor está grabando los datos.

La figura siguiente muestra los estados por los cuales ha pasado el servidor, como respuesta a las órdenes que recibe (y de las cuales envía confirmación al cliente). Las transiciones se producen cuando el servidor recibe una orden y ha enviado una confirmación positiva (la orden se indica en la flecha correspondiente).



10.2.4. Métodos RTSP

Los métodos RTSP siguientes son los más relevantes a la hora de cambiar de estado y de hacer la reserva de recursos para el *stream* en el lado servidor.

- SETUP: el servidor reserva recursos para un *stream* e inicia una sesión RTSP.
- PLAY y RECORD: se inicia la transmisión de datos de un *stream* inicializado con SETUP.
- PAUSE: detiene temporalmente la transmisión de datos, sin liberar los recursos del servidor.
- TEARDOWN: libera los recursos asociados al *stream*. La sesión RTSP se borra del servidor.

Los métodos RTSP que modifican el estado de la sesión utilizan el campo Sesión de la cabecera RTSP, que veremos en el subapartado siguiente. Para consultar la lista entera de métodos, podéis ver el RFC del RTSP.

10.2.5. El protocolo RTSP

En este subapartado veremos el protocolo RTSP, incluyendo el formato del mensaje de petición y respuesta y los campos de cabecera. Muchas de las características del protocolo se heredan del HTTP, pero aquí haremos una descripción completa. El URL del protocolo RTSP tiene la forma siguiente:

Esquemas RTSP

En RTSP, los URL pueden empezar con RTSP o RTSPU. El uso de RTSP indica que funciona sobre un nivel de transporte fiable (TCP), mientras que RTSPU indica que el protocolo funciona sobre un nivel de transporte no fiable (UDP).

```
rtsp://host:port/cami/al/recurso
```

Ejemplo:

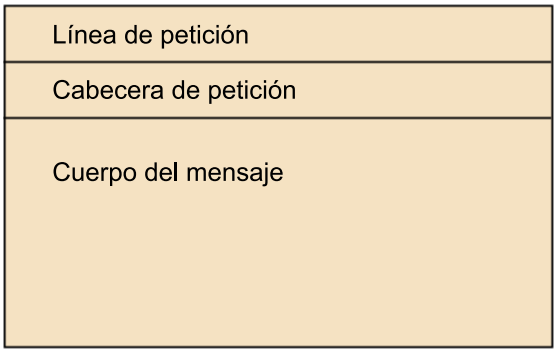
```
rtsp://servidor.multimedia.com:554/programa_tv.html
```

Los URL tendrían que evitar utilizar direcciones IP en lugar del nombre del *host*, y pueden referirse a un único *stream* o a una serie de *streams* agregados.

El RTSP es un protocolo basado en mensajes de texto. Esto facilita su implementación y ampliación (sólo hay que definir un nuevo método o una nueva cabecera). A continuación, veremos el formato de los mensajes de petición y respuesta de RTSP.

10.2.6. El mensaje de petición RTSP

La figura siguiente muestra el formato del mensaje de petición. Este mensaje consta de tres partes bien diferenciadas: la línea de petición, la cabecera de petición y el cuerpo del mensaje.



La línea de petición tiene la forma siguiente:

```
OrdenRTSP [Espacio] URI-petición [Espacio] Versión-RTSP [Salto línea]
```

La orden RTSP indica qué orden del protocolo estamos utilizando. El URI-petición identifica el recurso al cual queremos acceder, y la versión RTSP indica la versión del protocolo que estamos utilizando. [Espacio] es un espacio en blanco y [Salto línea] está representado por los caracteres Salto de línea (ASCII 10) y Retorno de carro (ASCII 13). El resto de los espacios en blanco se han puesto por claridad.

Ved también

Los valores de las órdenes RTSP, así como el formato de la versión RTSP, están explicados con detalle en el anexo 1.

```
Nombre-campo: [Espacio] valor-campo [Salto línea]
```

La cabecera de petición tiene tres partes: la cabecera general, la cabecera propiamente de petición y la cabecera de entidad. El formato de todos los campos de las cabeceras es el siguiente:

- La cabecera general representa información general que es independiente de si el mensaje es de petición o de respuesta.

- La cabecera de petición contiene campos que sólo tienen sentido para un mensaje de petición.

- La cabecera de entidad contiene información sobre el cuerpo del mensaje (que también se conoce como entidad).

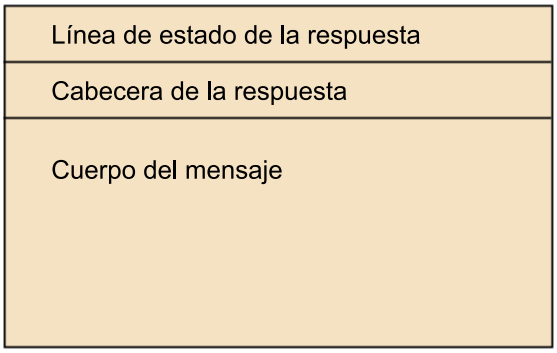
El cuerpo del mensaje lleva la información, si la hay, asociada a la petición.

Ved también

Los valores que pueden tomar las cabeceras de entidad están explicados con detalle en el anexo 1.

10.2.7. El mensaje de respuesta RTSP

La figura siguiente muestra el formato del mensaje de respuesta. Este mensaje consta de tres partes bien diferenciadas: la línea de estado de la respuesta, la cabecera y el cuerpo del mensaje.



La línea de estado de la respuesta tiene la forma siguiente:

```
Versión-RTSP [Espacio] Código-Estado [Espacio] por Descripción-Estado [Salto línea]
```

La versión RTSP define la versión del protocolo que se está utilizando, el código de estado define el éxito o error en la petición y la descripción del estado es una descripción textual del código de estado. [Espacio] es un espacio en blanco y [Salto línea] está representado por los caracteres Salto de línea (ASCII 10) y Retorno de carro (ASCII 13). El resto de los espacios en blanco se han puesto por claridad.

Ved también

Los valores de los códigos y las descripciones de estado están explicados con detalle en el anexo 1.

La cabecera de respuesta tiene tres partes: la cabecera general, la cabecera propiamente de respuesta y la cabecera de entidad. El formato de todos los campos de las cabeceras es el siguiente:

```
Nombre-campo: [Espacio] valor-campo [Salto línea]
```

La cabecera general representa información general que es independiente de si el mensaje es de petición o de respuesta. La cabecera de respuesta contiene campos que sólo tienen sentido para un mensaje de respuesta. La cabecera de entidad contiene información sobre el cuerpo del mensaje (que también se conoce como entidad).

Ved también

Los valores que pueden tomar estas tres cabeceras están explicados con detalle en el anexo 1.

El cuerpo del mensaje lleva la información, si la hay, asociada a la respuesta.

10.2.8. Ejemplo de uso del protocolo RTSP

En este subapartado veremos un ejemplo de uso del protocolo RTSP, con las peticiones y respuestas que nos encontraríamos en una comunicación real.

En este ejemplo, el cliente establece una conexión TCP en el puerto 554 del servidor y envía la orden OPTIONS, indicando la versión del protocolo RTSP que utilizará. El servidor confirma esta orden con un mensaje 200 OK (indicando que ha ido bien). Este código de respuesta es igual al equivalente en HTTP.

Petición del cliente:

```
OPTIONS rtsp://servidor.multimedia.com:554 RTSP/1.0
Cseq: 1
```

Respuesta del servidor:

```
RTSP/1.0 200 OK Cseq: 1
```

En un segundo paso, el cliente envía una orden de tipo DESCRIBE, que indica al servidor el contenido concreto que queremos. El servidor vuelve a responder con un 200 OK, incluyendo una descripción completa del contenido, en algún formato estandarizado –por ejemplo, *Session Description Protocol* (SDP) o *Multimedia and Hypermedia Experts Group* (MHEG).

SDP y MHEG

El *Session Description Protocol* (SDP) está definido en el RFC 4.566.

Multimedia and Hypermedia Experts Group (MHEG): http://www.mheg.org.

Petición del cliente:

```
DESCRIBE rtsp://servidor.multimedia.com:554/videos/
video.html RTSP/1.0
Cseq: 2
```

Petición del cliente:

```
SETUP rtsp:// servidor.multimedia.com:554/videos/
video.html RTSP/1.0
Cseq: 3
Transport: rtp/udp;unicast;client_port=5067-5068
```

Respuesta del servidor:

```
RTSP/1.0 200 OK Cseq: 3
Session: 12345678
Transport: rtp/udp;client_port=5067-5068;server_port=6023-
6024
```

Como se puede ver, además de las líneas de petición y respuesta, los mensajes llevan cabeceras que indican el número de secuencia del mensaje (Cseq) o las características de la entidad enviada (Content-type o Content-length).

En el siguiente paso de la negociación, el cliente envía una orden SETUP en la que le dice al servidor los mecanismos de transporte que quiere utilizar y el orden de preferencia. El cliente indica que quiere utilizar UDP como protocolo de transporte y los puertos 5067 y 5068 para la transferencia de datos. El servidor confirma los datos de transporte y puertos del cliente y añade el identificador de sesión y sus puertos de transferencia.

Después de establecer la conexión, el cliente está listo para empezar a recibir el *stream* y envía la orden PLAY. Esta orden sólo contiene el URL del contenido y el identificador de la sesión enviado previamente por el servidor. El servidor confirma la orden y se empieza a enviar el *stream*.

Si el cliente decide parar el *stream*, lo puede hacer con la orden TEARDOWN. El servidor envía la confirmación de que ha recibido la orden y se para el envío RTSP.

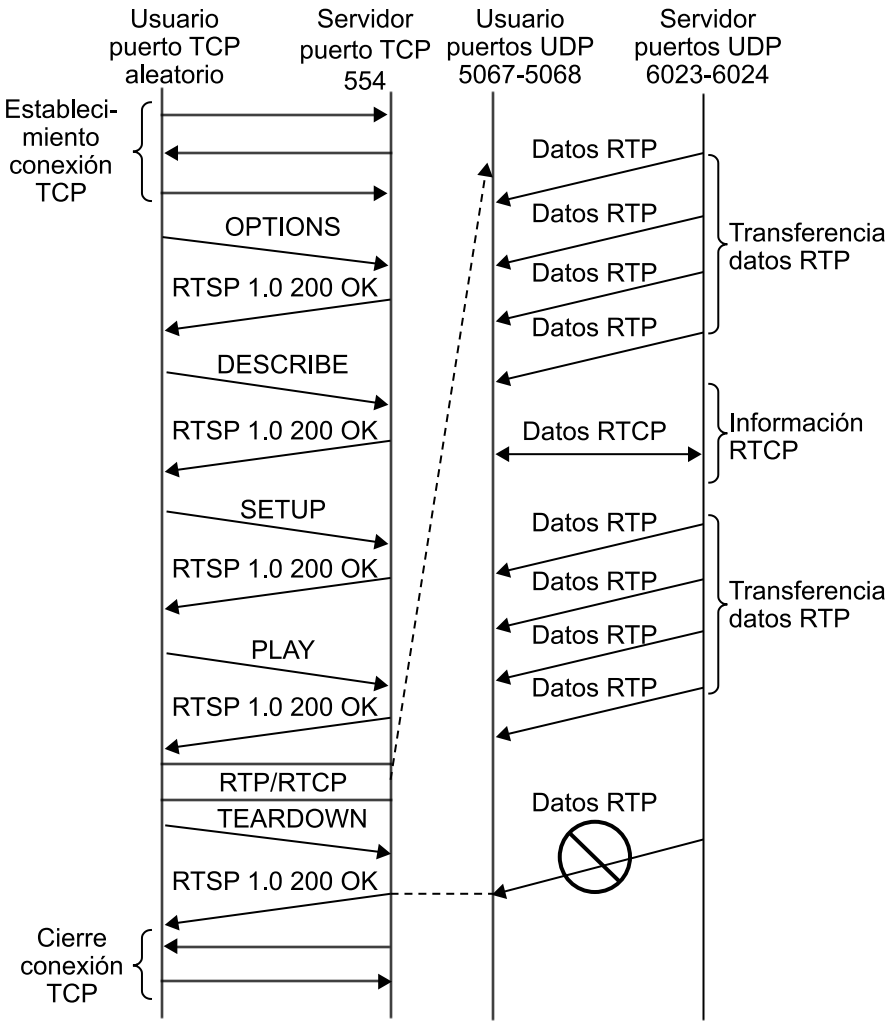
Petición del cliente:

```
PLAY rtsp:// servidor.multimedia.com:554/videos/
video.mpg RTSP/1.0
Cseq: 4
Session: 12345678
```

Respuesta del servidor:

```
RTSP/1.0 200 OK Cseq: 4
```

La figura siguiente resume los intercambios anteriores:



11. Protocolos para aplicaciones interactivas en tiempo real

11.1. *Real Time Transport Protocol* (RTP)

El *Real Time Transport Protocol* (RTP)[25] es un protocolo definido por la Internet Engineering Task Force (IETF). Este protocolo se puede utilizar para enviar cualquier tipo de formato: PCM, GSM o MP3 para audio y MPEG o H.263 para vídeo. Este protocolo es complementario a otros protocolos de tiempo real, como SIP o H.323.

[25]El RFC que define el RTP y el RTCP es el 3.550.

Ved también

Los protocolos SIP o H.323 se verán más adelante dentro de este mismo apartado.

11.1.1. Descripción del RTP

El RTP funciona sobre el protocolo de transporte UDP. El emisor encapsula un trozo de datos (*chunk*, en inglés) dentro del paquete RTP, que también se encapsula dentro de un datagrama UDP, el cual viaja en un paquete IP. El receptor extrae los datos RTP del datagrama UDP y pasa los datos al reproductor para que este descodifique el contenido y lo reproduzca.

Ved también

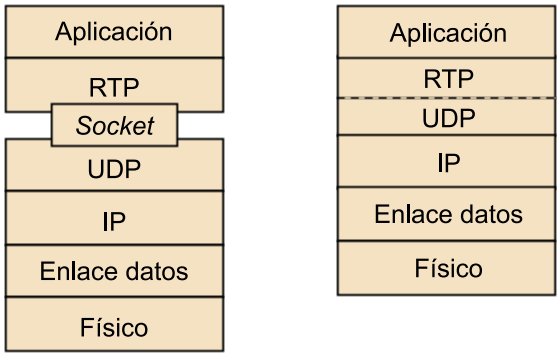
En este subapartado 11.1 hablaremos del RTP, y en el 11.2 hablaremos del protocolo de control del RTP, el RTCP.

Si una aplicación incluye RTP, será más fácil que pueda interoperar con otras aplicaciones multimedia. Por ejemplo, si dos fabricantes ofrecen un *software* de telefonía sobre IP e incorporan RTP en su producto, la interconexión entre sus productos será más factible.

El RTP no ofrece ningún mecanismo que permita asegurar que los datos llegan a su destino a tiempo o con la calidad de servicio adecuada. Tampoco garantiza que los paquetes lleguen en orden, ya que el protocolo RTP sólo se reconoce en los extremos y los direccionadores toman los paquetes IP que contienen RTP como si fueran cualquier otro paquete IP y no los diferencian del resto.

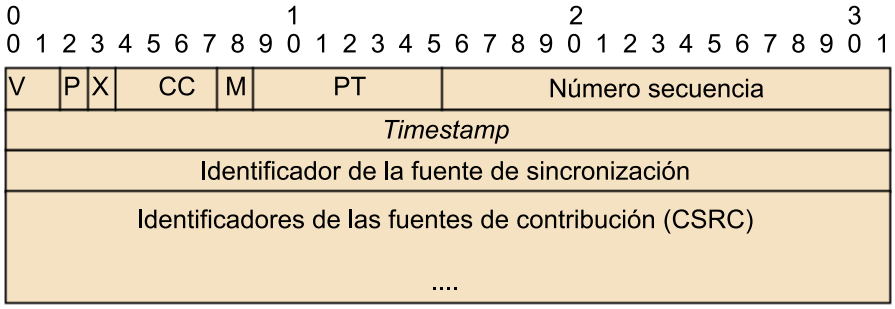
El RTP permite asociar a cualquier dispositivo (una cámara, un micrófono, etc.) su propio *stream* de paquetes. Así pues, si dos usuarios quieren hacer una videoconferencia, se pueden abrir dos *streams* de audio y vídeo, uno en cada sentido para cada tipo de datos. La realidad es que los *streams* de audio y vídeo están entrelazados, se envían como un único *stream* y se establece una única conexión para enviar los dos *streams* (audio y vídeo).

El RTP forma parte del nivel de aplicación (estaría por encima del UDP, que es el nivel de transporte), pero también se puede ver como un subnivel dentro del nivel de transporte. En la figura siguiente se puede ver esta distinción de forma gráfica:



11.1.2. La cabecera del RTP

El formato de la cabecera RTP se puede ver en la figura siguiente.



Los primeros 12 octetos están presentes en todos los paquetes RTP, mientras que la lista de identificadores CSRC (*Contributing Source*) sólo está presente cuando hay un mezclador.

Mezclador RTP

Un mezclador RTP se encarga de cambiar el formato del *stream* con el fin de adecuarlo al ancho de banda del receptor.

Los campos tienen el significado siguiente.

- Versión (V): 2 bits. Este campo identifica la versión RTP. La versión actual es la 2.

- Relleno (P): 1 bit. Si este bit está activado, el paquete contiene unos o más octetos de relleno adicional en los datos. El último octeto de relleno indica cuántos octetos se tienen que descartar, incluyendo el mismo. Esta técnica puede ser necesaria para algunos algoritmos de cifrado que necesitan un tamaño de bloque fijo.

- Extensión (X): 1 bit. Si está activado, la cabecera fija sólo puede llevar una extensión.

- Número de CSRC (CC): 4 bits. Este campo contiene el número de identificadores CSRC que hay en la cabecera fija.

- Marcador (M): 1 bit. La interpretación de este campo está definida por un perfil, que indica por qué se debe utilizar en un determinado contexto.

- Tipo de contenido (PT): 7 bits. Este campo identifica el formato de los datos que se están enviando con RTP.

- Número de secuencia: 16 bits. Este número se incrementa en uno cada vez que se envía un paquete de datos RTP y puede utilizarse para detectar pérdidas de datos y restaurar la secuencia del paquete. El valor inicial debe ser aleatorio.

- Timestamp: 32 bits. Indica el momento de tiempo del primer octeto enviado en este paquete de datos RTP.

- SSRC: 32 bits. Este campo identifica la fuente de sincronización. Tiene que ser aleatorio para que dos fuentes de sincronización de una sesión RTP no tengan el mismo identificador.

- Lista CSRC: de 0 a 15 elementos, 32 bits por elemento. Identifica las fuentes que contribuyen a los datos contenidos en este paquete RTP.

11.2. *Real Time Control Protocol* (RTCP)

El protocolo de control del RTP (RTCP) se basa en la transmisión periódica de paquetes de control a todos los participantes en una sesión, utilizando el mismo mecanismo de distribución que los paquetes de datos enviados con RTP. El protocolo que haya por debajo tiene que ofrecer la posibilidad de multiplexar paquetes de datos y de control, por ejemplo, por medio de números de puerto UDP diferentes. El RTCP lleva a cabo cuatro funciones básicas:

1) Da información sobre la calidad de los datos distribuidos. Esto forma parte del rol del RTP como protocolo de transporte y está muy relacionado con las funcionalidades de control de congestión ofrecidas por otros protocolos de transporte.

2) Mantiene un identificador persistente a nivel de transporte de una fuente RTP, que se denomina *nombre canónico* o *CNAME*, ya que, en el transcurso de la comunicación, el identificador SSRC puede cambiar si se detecta un conflicto o se reinicia un programa. Con el CNAME, si el identificador SSRC cambia, se pueden recuperar los participantes de la sesión.

3) Las dos funcionalidades anteriores necesitan que todos los participantes envíen paquetes RTCP, aunque se tiene que controlar la tasa de envío por si hay un número elevado de participantes. Si cada participante envía los paquetes de control a todos los otros, cada uno puede observar independientemente el número de participantes. Este número sirve para calcular la tasa a la cual se envían los paquetes.

4) La última función es opcional, y consiste en comunicar un mínimo de información de control de la sesión, como que se muestre la identificación de un participante en la interfaz de usuario.

11.2.1. Los paquetes RTCP

Los paquetes RTCP pueden transportar diferentes tipos de información de control.

- SR: informe del emisor (*Sender Report*, en inglés), para transmitir y recibir estadísticas de los participantes que son emisores activos.

- RR: informe del receptor (*Receiver Report*, en inglés), para recibir estadísticas de los participantes que no son emisores activos. También se puede combinar con el SR para los emisores activos que tienen que informar sobre más de 31 fuentes de datos (el máximo que se puede indicar en un paquete de tipo SR).

- SDES: elementos de descripción de la fuente, incluyendo el CNAME.

- BYE: indica que se ha dejado de participar.

- APP: funcionalidades específicas de la aplicación.

Cada paquete RTCP empieza con una parte fija similar a la que tienen los paquetes de datos RTP, seguida de una serie de elementos estructurados que pueden tener una estructura variable de acuerdo con el tipo de paquete, pero que deben tener una longitud múltiple de 32 bits. Se incluyen campos para indicar la alineación y un campo de longitud para hacer que los paquetes RTCP sean apilables. Los paquetes RTCP pueden enviarse de manera consecutiva, sin tener que ponerles ningún separador, y formar de este modo un paquete RTCP compuesto, que se puede enviar en un único paquete del nivel inferior (por ejemplo, UDP). Cada paquete RTCP dentro del paquete compuesto se tiene que procesar de manera independiente del resto de los paquetes. Sin embargo, para llevar a cabo las funciones del protocolo, se imponen las restricciones siguientes:

a) La recepción de estadísticas (en SR o RR) se tiene que enviar tan frecuentemente como lo permitan las restricciones de ancho de banda para maximizar la resolución de las estadísticas. Por este motivo, cada paquete RTCP compuesto tiene que llevar obligatoriamente un paquete de tipo informe.

b) Los nuevos receptores deben recibir el CNAME tan pronto como sea posible para identificar la fuente y poder asociar el contenido multimedia.

c) El número de tipos de paquetes que pueden aparecer inicialmente en el paquete compuesto se tiene que limitar para aumentar el número de bits constantes en la primera palabra del paquete y la probabilidad de validar con éxito los paquetes RTCP contra paquetes de datos RTP que tengan la dirección mal o que no estén relacionados.

Todos los paquetes RTCP se deben enviar obligatoriamente en paquetes compuestos de al menos dos paquetes individuales, con el formato siguiente.

- Prefijo de cifrado: si el paquete RCTP se tiene que cifrar, entonces se debe poner ante un prefijo aleatorio de 32 bits, que debe recalcularse para cada paquete compuesto. Si es preciso relleno, se tiene que poner en el último paquete del paquete compuesto.

- SR o RR: el primer paquete del paquete compuesto debe ser de tipo informe, para facilitar la validación de la cabecera. Esto se tiene que hacer siempre, aunque no se hayan enviado o no se hayan recibido datos. En este caso, se tiene que enviar un paquete RR vacío.

- RR adicionales: si el número de fuentes de las que se debe reportar es mayor que 31, entonces se tienen que enviar paquetes RR adicionales hasta completar los datos del primer paquete SR o RR enviado.

- SDES: un paquete de tipo SDES que contenga un CNAME se tiene que incluir siempre en el paquete compuesto.

- BYE o APP: el resto de los paquetes pueden aparecer en cualquier orden e incluso repetidos, excepto los de tipos BYE, que tienen que ir al final para un SSRC/ CSRC dado.

Un participante RTP tiene que enviar un único paquete compuesto por intervalo de envío de informes para calcular correctamente el ancho de banda de RTCP por participante.

Ved también

El formato específico de cada paquete RTCP se define en el anexo 2 de este módulo didáctico.

11.2.2. Uso del ancho de banda en el protocolo RTCP

Para el funcionamiento del RTCP, se puede ver que, en una conexión con un emisor y muchos receptores (que pueden comunicarse con *multicast*), los datos enviados con RTCP pueden exceder en gran manera los datos enviados con RTP por el emisor. Por este motivo, el RTCP modifica la tasa de envío de los paquetes RTCP a medida que aumenta el número de participantes en la sesión.

El RTCP intenta mantener su tráfico en el 5% del ancho de banda de la sesión. Veamos esto con un ejemplo.

Ancho de banda RTSP

Si un emisor transmite a una velocidad de 2 Mbps, el RTCP intenta limitar su tráfico en el 5% de estos 2 Mbps, que serían 100 kbps.

El protocolo da el 75% de esta tasa, 75 kbps, a los receptores, y el resto, los 25 kbps restantes, al emisor. Los 75 Kbps de los receptores se reparten entre los mismos de manera igualitaria. De este modo, si hay R receptores, cada receptor puede enviar tráfico RTCP a una tasa de 75 kbps/*R* y el emisor envía a una tasa de 25 kbps. Los participantes (emisores o receptores) determinan el periodo de transmisión de un paquete RTCP calculando dinámicamente el tamaño medio de paquete RTCP (en toda la sesión) y dividiéndolo entre el tamaño medio de paquete RTCP que puede enviar a su tasa asignada.

Para resumir, el periodo en el que un emisor puede enviar un paquete RTCP es:

Y en resumen, el periodo en el que un receptor puede enviar un paquete RTCP es:

$$T = \frac{\text{número emisores}}{0{,}25 * 0{,}05 * \text{anchura banda sesión}} \bullet (\text{medida mediana paquete RTCP})$$

11.3. *Session Initiation Protocol* (SIP)

Hay muchas aplicaciones en Internet que necesitan la creación y gestión de sesiones, entendiendo sesión como un intercambio de datos entre distintos participantes. La implementación de las sesiones se hace difícil por el comportamiento de los que participan en las mismas: los usuarios cambian de localización, pueden tener distintas maneras de identificarse e intercambian diferentes tipos de datos, a veces de manera simultánea. Hay muchos protocolos que permiten trabajar con sesiones multimedia en tiempo real con las que es posible transmitir texto, audio o vídeo.

El *Session Initiation Protocol* (SIP)[26] trabaja en cooperación con estos protocolos, y permite que los diferentes agentes de usuario (aplicaciones que utilizan a los usuarios para comunicarse) puedan encontrarse y ponerse de acuerdo con el tipo de sesión que quieren compartir. Para localizar a los participantes de una sesión y para otras funciones, el SIP permite la creación de una infraestructura de *hosts* (denominados *proxy servers*), a la que las aplicaciones de usuario pueden enviar peticiones de registro, invitación a las sesiones y otras peticiones. El SIP es una herramienta que permite gestionar sesiones independientemente de los protocolos de transporte que haya por debajo y sin ninguna dependencia del tipo de sesión establecida.

[26]El RFC donde se define el SIP es el 3.261.

Aplicaciones que utilizan el SIP

Windows Live Messenger y Yahoo Messenger son dos aplicaciones que utilizan SIP.

11.3.1. Funcionalidad del SIP

El SIP es un protocolo de control de nivel de aplicación que puede establecer, modificar y finalizar sesiones multimedia (conferencias). Como ejemplo de sesión, tenemos las llamadas telefónicas mediante Internet. El SIP también permite invitar a participantes a sesiones existentes, como por ejemplo las conferencias *multicast*. También se pueden añadir y quitar contenidos multimedia de una sesión existente. Con SIP, los usuarios pueden tener un único identificador externo, independientemente de su localización de red.

El SIP considera cinco aspectos diferentes para el establecimiento y la finalización de comunicaciones multimedia.

- Localización del usuario: hace referencia al sistema final que se hará para la comunicación.

- Disponibilidad del usuario: indica si el usuario quiere participar en las comunicaciones.

- Capacidad de los usuarios: indica el medio y los parámetros del medio que se utilizarán.

- Inicio de la sesión: llamada, establecimiento de la sesión y parámetros a los dos extremos de la comunicación.

- Gestión de la sesión: incluye la transferencia y finalización de sesiones, modificando los parámetros de la sesión y llamando a los servicios.

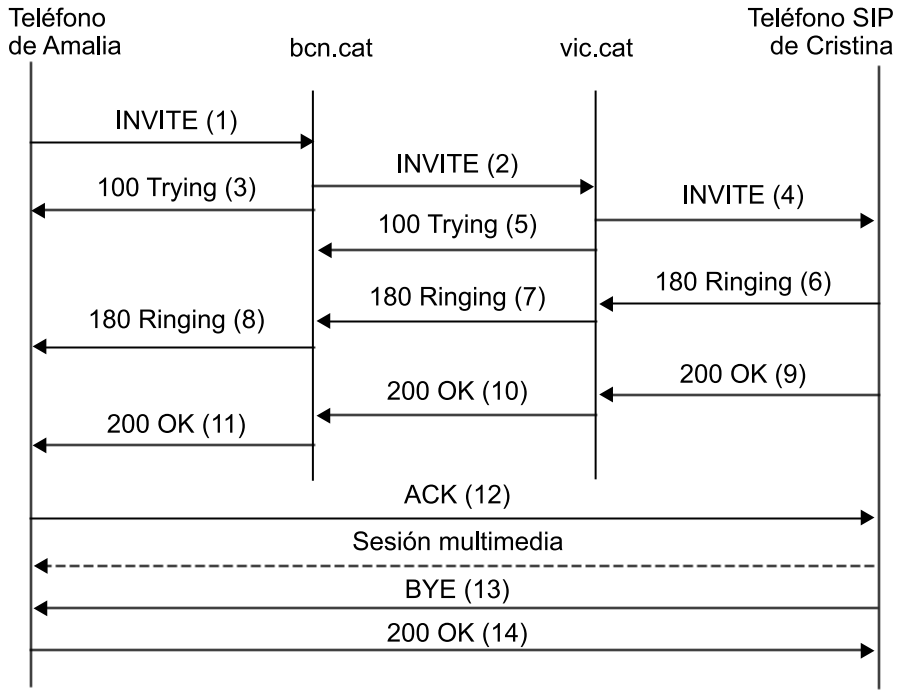
El SIP no es un sistema completo de comunicaciones, sino que se trata de un componente que se puede complementar con otros protocolos de Internet con el fin de implementar una aplicación multimedia entera. Por ejemplo, se pueden utilizar los protocolos RTP y RTCP para el transporte y control del envío de datos o el protocolo RTSP para el control del *streaming* de datos multimedia.

El SIP no ofrece servicios, sino primitivas que se pueden utilizar para ofrecer diferentes tipos de servicio. Por ejemplo, el SIP tiene una primitiva que permite encontrar a un usuario y enviarle datos opacos a su localización. Si estos datos consisten en una descripción de sesión definida, por ejemplo con SDP, los extremos de la comunicación pueden ponerse de acuerdo con los parámetros de la sesión.

El SIP tampoco ofrece servicios de control de las conferencias, ni define cómo se tiene que gestionar una conferencia. Sin embargo, se puede utilizar para iniciar una conferencia que utiliza otro protocolo para el control de conferencias.

La naturaleza de los servicios que se pueden ofrecer con SIP hace que la seguridad sea muy importante. Por este motivo, el SIP ofrece toda una serie de mecanismos de seguridad que incluyen la prevención de la denegación de servicio, la autenticación de usuarios y *proxies*, la protección de la integridad y los servicios de cifrado y privacidad.

La figura siguiente muestra un ejemplo de funcionamiento del SIP:



En este ejemplo, Amalia y Cristina quieren establecer una sesión SIP. Cada intercambio tiene al lado un número entre paréntesis (*n*), para hacer de referencia en la explicación del mismo. También se muestran los dos servidores *proxy* que actúan en nombre de Amalia y Cristina para facilitar el establecimiento de la sesión.

Amalia llama a Cristina utilizando su identidad SIP, una especie de URI, denominado *URI SIP*. Un URI SIP es semejante a una dirección de correo electrónico; lo definiremos más adelante, pero contiene un *host* y un nombre de usuario. En este caso, este URI es sip:amalia@bcn.cat, en el que bcn.cat es el dominio del proveedor de servicio de Amalia. El URI SIP de Cristina es sip:cris@vic.cat. Amalia puede haber escrito la dirección de Cristina o puede haberla elegido de un enlace o una agenda. También existen los URI SIP seguros, que se denominan *SIPS*, como por ejemplo sips:cris@vic.cat. En este caso, se utilizaría el *Transport Layer Security* (TLS)[27] para proveer de una conexión segura y cifrada para transportar los mensajes SIP.

[27]El RFC del TLS es el 4.346.

El SIP se basa en un mecanismo de petición/respuesta muy similar al modelo de transacciones HTTP. Cada transacción consiste en una petición que invoca un método del servidor y, como mínimo, una respuesta. En este ejemplo, la transacción empieza cuando Amalia envía una orden INVITE dirigida al URI SIP de Cristina.

A continuación, se muestra la información que llevaría asociada la orden INVITE:

```
INVITE sip:cris@vic.cat SIP/2.0
```

```
Via: SIP/2.0/UDP pc01.bcn.cat;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Cristina <sip:cris@vic.cat>
From: Amalia <sip:amalia@bcn.cat>;tag=1928301774
Call-ID: a84b4c76e66710@pc01.bcn.cat
CSeq: 314159 INVITE
Contact: <Sip:amalia@pc01.bcn.cat> Content-Type: application/sdp Content-Length: 142

... Resto de datos SDP ...
```

El contenido del mensaje es el siguiente:

- La primera línea contiene el método, con el URI SIP origen y la versión de SIP.

- Las líneas siguientes son cabeceras, que contienen información sobre el origen y el destino de la comunicación (From, To), la dirección donde Amalia quiere recibir las respuestas, indicada en Via, y algunos identificadores únicos. Cseq es un número de secuencia que se incrementa con cada nueva petición. Contact contiene el URI SIP que representa la ruta directa hacia Amalia, formada por un nombre de usuario y el nombre de dominio. El número Max-Forwards indica el número máximo de saltos que puede dar esta petición y, finalmente, Content-Type y Content-Length contienen información sobre el cuerpo del mensaje.

Ved también

Se puede encontrar una lista completa de las cabeceras SIP en el anexo 3 de este módulo didáctico.

Los parámetros específicos de la sesión no se definen con SIP, sino con otro protocolo, como el SDP, que no veremos aquí.

Puesto que el teléfono de Amalia no sabe dónde encontrar a Cristina o el servidor SIP de bcn.cat, envía la orden INVITE a su servidor de dominio, bcn.cat (1). El servidor bcn.cat es un tipo de servidor SIP conocido como *proxy server*. Su función es redireccionar las peticiones SIP en nombre de quien las envía. En este ejemplo, *el proxy server* recibe la petición INVITE (2) y responde con una respuesta 100 (Trying) (3). Esto indica que la petición se ha recibido bien y que se está intentando enviar INVITE a su destino. La respuesta SIP tiene la forma de un código de 3 cifras y una frase descriptiva (como HTTP).

El *proxy server* bcn.cat encuentra la dirección SIP del dominio vic.cat. De esta manera, consigue la dirección del *proxy server* de vic.cat y le reenvía la orden INVITE (2). Este contesta con un 100 (Trying) (5), para indicar que está trabajando en servir la petición. El *proxy server* consulta una base de datos para encontrar la dirección de Cristina y le envía la petición INVITE (4) a su teléfono SIP. Cuando el teléfono recibe la petición, suena para informar a Cristina de que está recibiendo una llamada. El teléfono SIP indica que se está haciendo la llamada enviando un mensaje de tipo 180 (Ringing) (6), (7), (8), que llega hasta el teléfono de Amalia. Cuando la respuesta llega al teléfono de Amalia,

este hace alguna indicación de que está sonando el teléfono en el otro extremo. En este caso, Cristina responde a la llamada, lo cual hace que se envíe una respuesta de tipo 200 OK (9), (10), (11). Amalia envía un ACK (12). En este momento, hay que negociar los parámetros de la sesión con el protocolo SDP. A continuación, se muestra cómo sería el mensaje de respuesta enviado por el teléfono SIP de Cristina:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy03.vic.cat
;branch=z9hG4bKnashds8;received=192.0.2.3
Vía: SIP/2.0/UDP proxy01.bcn.cat
;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc01.bcn.cat
;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Cristina <sip:cris@vic.cat>;tag=a6c85cf
From: Amalia <sip:amalia@bcn.cat>;tag=1928301774
Call-ID: a84b4c76e66710@pc01.bcn.cat
CSeq: 314159 INVITE
Contact: <Sip:cristina@192.0.2.4> Content-Type: application/sdp Content-Length: 131

... Resto de datos SDP ...
```

El contenido del mensaje es el siguiente:

- La primera línea contiene la respuesta, con la versión de SIP, el código de respuesta y la descripción.

- Las líneas siguientes son cabeceras. Via, From, Call-ID y Cseq se copian de la petición INVITE (ahora hay distintas cabeceras de tipo Via para indicar la comunicación mediante los *proxy servers*). Contact contiene el URI SIP que representa la ruta directa hacia Cristina, formada por un nombre de usuario y nombre de dominio. Content-Type y Content-Length contienen información sobre el cuerpo del mensaje.

La sesión multimedia entre Cristina y Amalia ha empezado y pueden enviar los datos multimedia en el formato acordado. En general, estos paquetes seguirán una ruta diferente de la de los mensajes SIP. Durante la comunicación, se pueden cambiar las características de la sesión volviendo a enviar una orden INVITE.

Para acabar la sesión, Cristina envía una orden BYE (cuelga) (13). Este BYE se envía directamente al teléfono de Amalia, sin pasar por los *proxies*. Amalia responde con un 200 OK (14). En este caso, no hay ACK.

11.3.2. El protocolo SIP

En este subapartado veremos el protocolo SIP, incluyendo el formato de los mensajes de petición y respuesta y los campos de cabecera. Muchas de las características del protocolo se heredan de HTTP, pero no se trata de una extensión. Por ejemplo, el SIP puede utilizar UDP para enviar sus peticiones y respuestas. Las direcciones (URI SIP) de los usuarios que utilizan este protocolo tienen la forma:

URI SIP

En SIP, los URI (direcciones) pueden empezar con SIP o SIPS. El uso de SIPS indica que se tiene que establecer una conexión segura (con TLS) para llevar a cabo la comunicación.

```
sip:user:password@host:port;uri-parameters?headers
Ejemplo:
sip:amalia@bcn.cat
```

Los URI para SIPS siguen el mismo formato, cambiando SIP por SIPS al inicio del URI.

El resto de los campos del URI son los siguientes.

- user: el identificador de un recurso en la máquina (*host*) a la que se está dirigiendo. En este contexto, *host* se refiere a un dominio.

- password: el *password* asociado al usuario. Aunque se puede poner, su uso no está recomendado, porque significa enviarlo "en claro" y es un riesgo de seguridad.

- host: el ordenador que ofrece el recurso SIP. Este campo puede contener un nombre de dominio o una dirección IP numérica. Se recomienda utilizar el nombre de dominio.

- port: el número de puerto al que se tiene que enviar la petición.

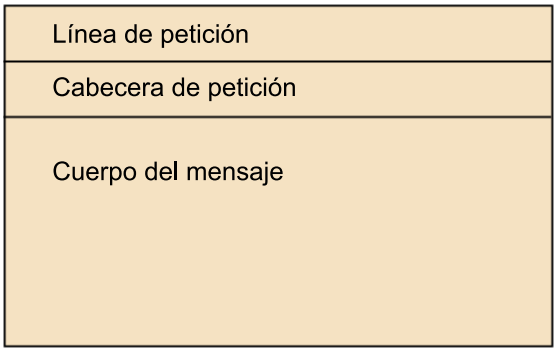
- uri parameters: parámetros que afectan a la petición que lleva el URI. Los parámetros se representan como nom-param=valor-param. Están separados de host:port con punto y coma y entre estos también se separan con punto y coma.

- headers: campos de cabecera que se pueden incluir en la petición representada por el URI.

SIP es un protocolo basado en mensajes de texto. Esto facilita su implementación y ampliación. A continuación, veremos el formato de los mensajes de petición y respuesta SIP.

11.3.3. El mensaje de petición SIP

La figura siguiente muestra el formato del mensaje de petición. Este mensaje consta de tres partes bien diferenciadas: la línea de petición, la cabecera de petición y el cuerpo del mensaje.



La línea de petición tiene la forma siguiente:

```
OrdenSIP [Espacio] URI-petición [Espacio] Versión-SIP [Salto línea]
```

El orden SIP indica qué orden del protocolo estamos utilizando. El URI-petición identifica el recurso al que queremos acceder, y la versión SIP indica la versión del protocolo que estamos utilizando. [Espacio] es un espacio en blanco y [Salto línea] está bien representado por los caracteres Salto de línea (ASCII 10) y Retorno de carro (ASCII 13). El resto de los espacios en blanco se han puesto por claridad. Las órdenes SIP son: REGISTER, INVITE, ACK, CANCEL, BYE y OPTIONS. La funcionalidad de cada método es la siguiente.

- REGISTER: permite registrar un recurso, para que después se puedan llevar a cabo comunicaciones SIP.
- INVITE: permite iniciar una sesión por parte de un usuario.
- ACK: indica que se ha aceptado la petición enviada con INVITE y que la sesión se ha establecido.
- CANCEL: permite cancelar una petición ya enviada. Se genera un mensaje de error y se tiene que dejar de procesar la petición. Sólo se tendría que utilizar para cancelar un INVITE.
- BYE: permite finalizar la sesión.
- OPTIONS: permite pedir las características de un programa cliente o de un *proxy server*.

La cabecera de petición tiene tres partes: la cabecera general, la cabecera propiamente de petición y la cabecera de entidad. El formato de todos los campos de las cabeceras es el siguiente:

```
Nombre-campo: [Espacio] valor-campo [Salto línea]
```

La cabecera general representa información general que es independiente de si el mensaje es de petición o de respuesta. La cabecera de petición contiene campos que sólo tienen sentido para un mensaje de petición. La cabecera de entidad contiene información sobre el cuerpo del mensaje (que también se conoce como entidad).

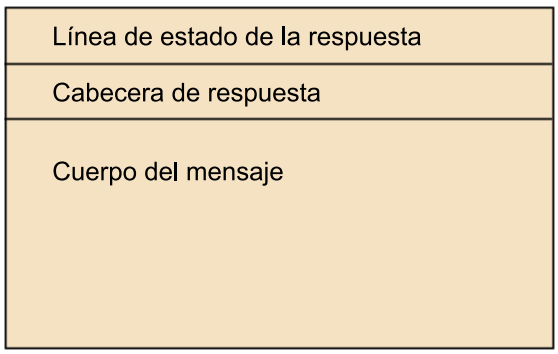
Ved también

Los valores que pueden tomar estas tres cabeceras están explicados con detalle en el anexo 3.

El cuerpo del mensaje lleva la información, si la hay, asociada a la petición.

11.3.4. El mensaje de respuesta SIP

La figura siguiente muestra el formato del mensaje de respuesta. Este mensaje consta de tres partes bien diferenciadas: la línea de estado de la respuesta, la cabecera y el cuerpo del mensaje.



La línea de estado de la respuesta tiene la forma siguiente:

```
Versión-SIP [Espacio] Código-Estado [Espacio] por Descripción-Estado [Salto línea]
```

La versión SIP define la versión del protocolo que se está utilizando, y el código de estado define el éxito o error en la petición y la descripción del estado, que es una descripción textual del código de estado. [Espacio] es un espacio en blanco y [Salto línea] está representado por los caracteres Salto de línea (ASCII 10) y Retorno de carro (ASCII 13). El resto de los espacios en blanco se han puesto por claridad.

Ved también

Los valores de los códigos y descripciones de estado están explicados con detalle en el anexo 3.

La cabecera de respuesta tiene tres partes: la cabecera general, la cabecera propiamente de respuesta y la cabecera de entidad. El formato de todos los campos de las cabeceras es:

```
Nombre-campo: [Espacio] valor-campo [Salto línea]
```

La cabecera general representa información general que es independiente de si el mensaje es de petición o de respuesta. La cabecera de respuesta contiene campos que sólo tienen sentido para un mensaje de respuesta. La cabecera de entidad contiene información sobre el cuerpo del mensaje (que también se conoce como entidad).

Ved también

Los valores que pueden tomar estas tres cabeceras están explicados con detalle en el anexo 3.

El cuerpo del mensaje lleva la información, si la hay, asociada a la respuesta.

11.4. H.323

Este protocolo está considerado como una alternativa al SIP y es muy popular a la hora de transmitir audio y vídeo en tiempo real. Está dedicado a la transmisión de voz sobre IP (*voice over IP* –VOIP–, en inglés). Con H.323, es posible comunicar un equipo conectado a Internet con un teléfono conectado a la red telefónica.

H.323 incluye distintas especificaciones:

- Negociación de codificaciones de audio y vídeo entre los extremos de la comunicación.
- Cómo se envían los trozos de datos de audio y vídeo. H.323 obliga a utilizar RTP.
- Cómo los equipos se comunican con sus *gatekeepers*.
- Cómo los equipos conectados a Internet se conectan con los teléfonos conectados a la red telefónica.

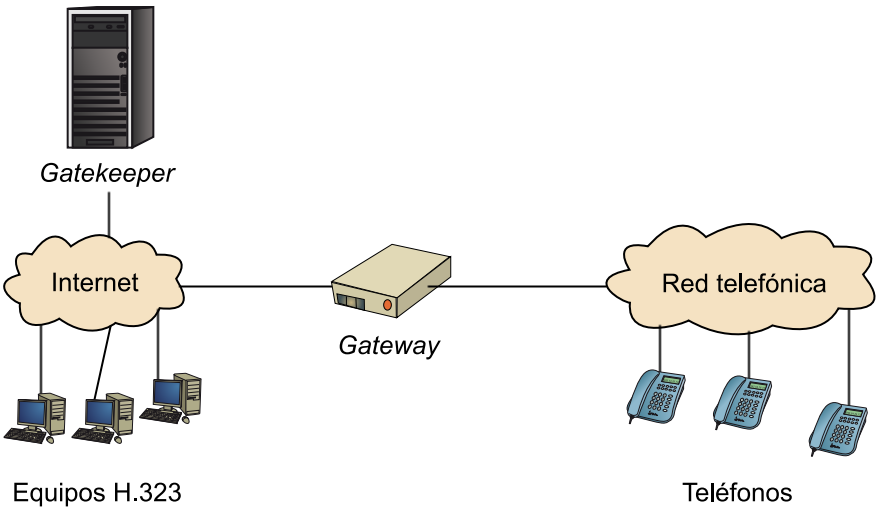
El requerimiento mínimo de H.323 es que se debe soportar el audio, pero las características de vídeo son opcionales. De esta manera, los fabricantes pueden ofrecer unos equipos sencillos con las características de audio y dejar para equipos más sofisticados la parte de vídeo. Para transmitir los datos de audio, se tiene que utilizar un estándar que utiliza PCM (G.711) para generar voz digitalizada tanto en 56 kbps como 64 kbps. Aunque el vídeo es opcional, en caso de que el terminal lo tenga, debe soportar como mínimo el estándar QCIF H.261.

Formatos de datos utilizados en H.323

G.711 – Audio http://www.itu.int/rec/T-REC-G.711/en.

QCIF H.261 – Vídeo http://www.itu.int/rec/T-REC-H.261/en.

La figura siguiente muestra un ejemplo de arquitectura de un sistema H.323:



El *gateway* es un elemento opcional que sólo se necesita si queremos conectar equipos que están en una red diferente, como la red telefónica.

El *gatekeeper* es un elemento opcional en la comunicación entre terminales H.323. Sin embargo, es el elemento más importante de una red H.323, ya que hace de punto central de todas las llamadas que hay dentro de una zona y proporciona servicios a los terminales registrados y control de las llamadas. Se podría decir que el *gatekeeper* hace de conmutador virtual.

Para acabar este subapartado, se describen las diferencias principales entre SIP y H.323.

- H.323 es un conjunto integrado de protocolos para hacer conferencias multimedia que incluye: señalización, registro, control de admisión, transporte y *codecs*. Por otra parte, SIP sólo se encarga de iniciar y gestionar la sesión, sin hacer ninguna imposición en el transporte o los formatos de audio o vídeo soportados.

- H.323 fue definido por ITU (telefonía), mientras que SIP fue definido por IETF (Internet). Esto hace que el punto de vista de los dos sea muy diferente.

- H.323 es un estándar complejo, formado por muchas partes, mientras que SIP es mucho más simple y, por lo tanto, más fácil de implementar.

11.5. Skype

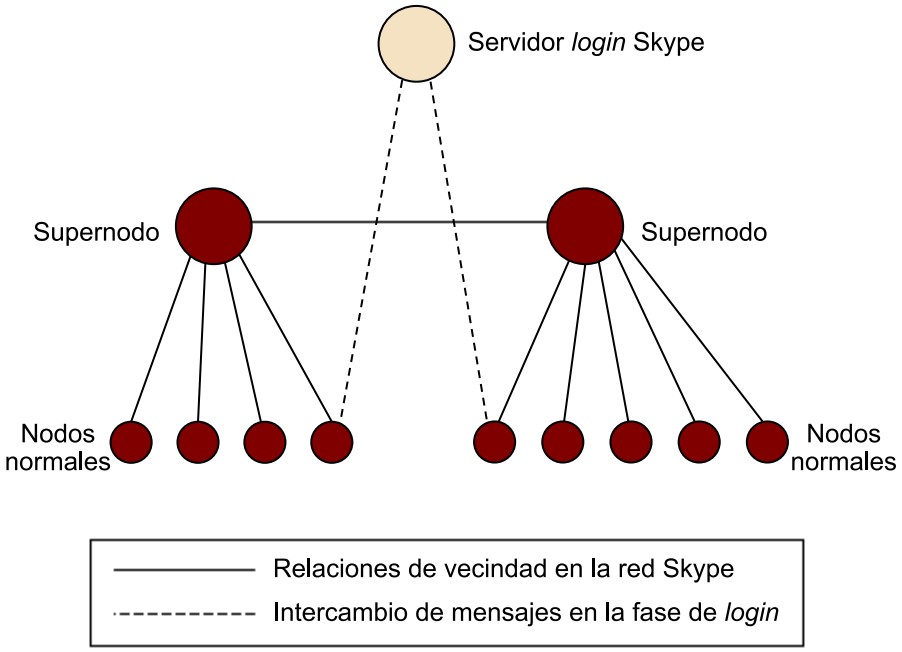
Skype es un sistema de telefonía *peer-to-peer* (P2P) que funciona sobre la red Internet. Es la competencia de estándares de transmisión de voz sobre IP (VoIP) como SIP o H.323. Fue desarrollada por el equipo que hizo KaZaa en el año 2003.

KaZaa

KaZaa es una aplicación P2P de descarga de contenidos que tuvo mucho éxito hace unos años. Utiliza el mismo protocolo P2P que Skype, FastTrack.

Se trata de una aplicación muy utilizada desde que se creó, tanto en sus servicios gratuitos como de pago. Ofrece muchas funcionalidades, incluyendo audio y videoconferencias gratuitas, el uso de tecnologías P2P (descentralizado) para no tener problemas con cortafuegos o con la traducción de direcciones (en inglés, *Network Address Translation*, NAT) y las múltiples medidas para que no se pueda reconocer el protocolo o el *software* que la implementan.

La identificación del usuario que llama está enmascarada cuando se hace una llamada. La diferencia principal entre Skype y otros servicios es que utiliza tecnología P2P, mientras que el resto utiliza arquitecturas cliente-servidor. La implementación de P2P que utiliza es FastTrack, implementada en la aplicación KaZaa. La arquitectura que implementa FastTrack es una red superpuesta (*overlay network*, en inglés) con dos tipos de nodos: los nodos normales y los supernodos. Un nodo normal es un ordenador en el que se ha instalado la aplicación Skype y que permite hacer llamadas y enviar mensajes de texto. Los supernodos son *end-points* de la red Skype. Cualquier nodo que tenga una IP pública y bastantes recursos (memoria, ancho de banda, etc.) puede convertirse en un supernodo. Un nodo normal tiene que hacer *login* en la red Skype con su usuario y palabra de paso y, de esta manera, estará conectado a un supernodo. Todavía tenemos otro elemento en la red, el servidor de *login*, que es el que almacena los usuarios y las palabras de paso en toda la red Skype (los nombres de usuario son únicos). Aparte de este servidor, el resto de las operaciones en la red (busca de usuarios, envío de mensajes y llamadas, etc.) se hace de manera totalmente descentralizada.



El directorio de Skype está completamente descentralizado y distribuido entre los nodos de la red, lo que significa que es fácilmente escalable cuando se tiene un gran número de usuarios sin necesidad de una estructura compleja. Las llamadas se envían a través de otros usuarios de Skype en la red para facilitar la comunicación cuando hay cortafuegos o NAT. Esto hace que los usuarios que se conectan directamente a la red (sin NAT) tengan más tráfico para direccionar las llamadas de otros usuarios. Puesto que se trata de una red superpuesta, cada cliente Skype tiene que mantener una lista de nodos accesibles. Esta lista es la lista de *hosts* y contiene la dirección IP y el número de puerto de los supernodos. En sistemas Windows, se guarda en el registro.

Es posible desarrollar programas que utilicen la API de Skype para acceder a su red. Sin embargo, el código es cerrado y el protocolo propietario, lo que dificulta su interoperabilidad con otros sistemas.

12. Anexos

12.1. Anexo 1. El protocolo RTSP

12.1.1. Órdenes

La tabla siguiente describe las órdenes RTSP:

Orden	Descripción
OPTIONS	Permite al cliente pedir información sobre las opciones de comunicación con el servidor.
DESCRIBE	Recupera la información del contenido multimedia identificado por el URL que se envía con la orden.
ANNOUNCE	Si lo envía el cliente, se está indicando la descripción de una presentación. Si lo envía el servidor, se actualiza la información de la descripción de la sesión en tiempo real.
SETUP	Especifica los parámetros del mecanismo de transporte. Se puede enviar a media sesión, para cambiar estos parámetros.
PLAY	Se indica al servidor que se debe iniciar el envío de datos.
PAUSE	Se indica al servidor que se quiere parar temporalmente la recepción de datos de la sesión.
TEARDOWN	Se cierra la recepción de datos.
GET_PARAMETER	Pide el valor de un parámetro de la descripción de la sesión.
SET_PARAMETER	Envía el valor de un parámetro de la descripción de la sesión.
REDIRECT	Se indica al cliente que se tiene que conectar a otro servidor.
RECORD	Se inicia la grabación de datos de acuerdo con la descripción de la presentación.

12.1.2. Códigos de estado

Los códigos de estado son de cinco tipos: 1XX, informativos, 2XX, petición lograda, 3XX, redireccionamiento, 4XX, error en la petición del cliente y 5XX, error en el servidor.

Sólo se han puesto los códigos específicos de RTSP, el resto están definidos en el RFC de HTTP.

Tipo de código	Valor	Descripción
Informativo	100	El cliente puede continuar con la petición.
Éxito	250	Poco espacio de almacenamiento. Puede suceder con la orden RECORD.

Tipo de código	Valor	Descripción
Redirección	3XX	Los mismos que HTTP.
Error parte cliente	405	El recurso no puede ejecutar el método pedido.
Error parte cliente	451	El servidor no soporta alguno de los parámetros indicados en la petición.
Error parte cliente	452	La conferencia no se ha encontrado.
Error parte cliente	453	No hay suficiente ancho de banda para recibir el contenido.
Error parte cliente	454	La sesión no se ha encontrado.
Error parte cliente	455	El método no es válido en el estado actual del protocolo.
Error parte cliente	456	Un parámetro indicado en la cabecera no se puede procesar.
Error parte cliente	457	El rango pedido está fuera de los límites.
Error parte cliente	458	Un parámetro que se quería modificar con SET_PARAMETER es sólo de lectura.
Error parte cliente	459	El parámetro no se puede aplicar al recurso. Sólo se puede aplicar a un *stream*
Error parte cliente	460	El parámetro no se puede aplicar al recurso. Sólo se puede aplicar a una presentación.
Error parte cliente	461	El tipo de transporte no está soportado.
Error parte cliente	462	El cliente no es accesible.
Error parte servidor	551	Una opción enviada a la cabecera no es soportada.

12.1.3. Cabeceras

En la tabla siguiente, se describen las cabeceras del protocolo RTSP que son específicas de este protocolo. El resto están definidas en el RFC de HTTP:

Cabecera	Descripción
Accept	Tipos de contenidos aceptados en la presentación.
Allow	Métodos RTSP soportados por el servidor.
Bandwidth	Ancho de banda calculado del cliente.
Blocksize	Tamaño de datos preferido por el cliente. No incluye las cabeceras de protocolos de niveles inferiores (IP, UDP, RTSP).
Cache-Control	Control del mecanismo de caché de los contenidos.
Conference	Establece una conexión entre esta conferencia y un *stream* existente.
Content-Length	Longitud del contenido enviado en el cuerpo del mensaje.
CSeq	Número de secuencia en la sesión RTSP de pares petición/respuesta.
Expires	Momento en el que el contenido estará obsoleto.
If-Modified-Since	Se utiliza con SETUP y DESCRIBE. Si la descripción no se ha modificado desde el momento indicado en esta cabecera, se envían datos.

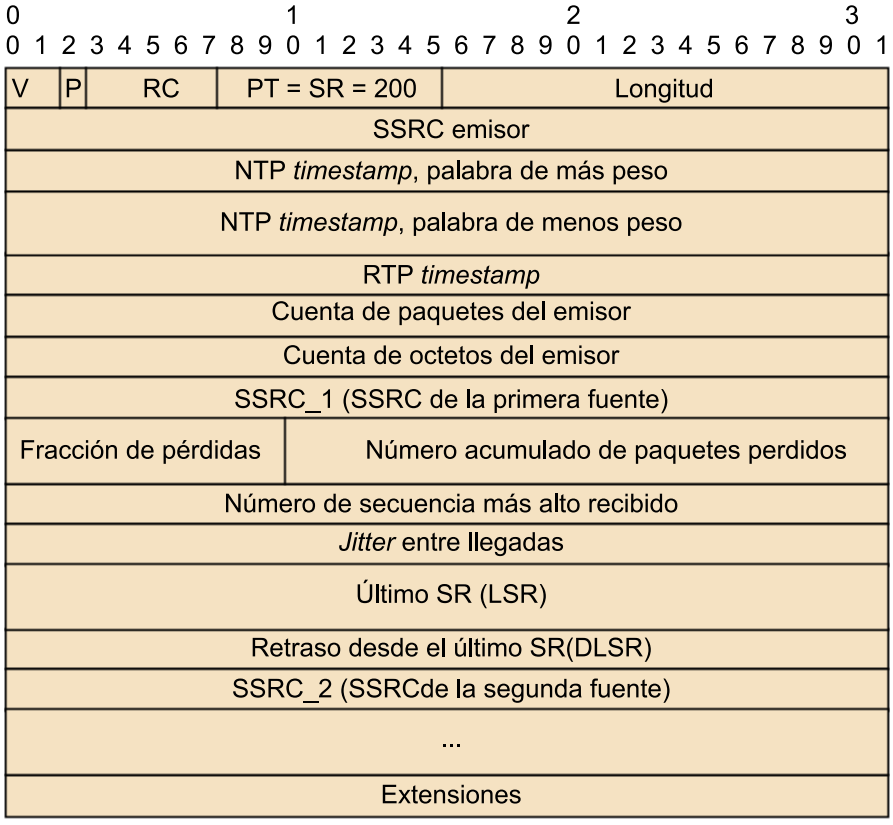
Cabecera	Descripción
Last-Modified	Última vez en la que se modificó la descripción del recurso.
Proxy-Require	Parámetros que debe soportar un *proxy*.
Require	Sirve para preguntar al servidor qué opciones soporta.
RTP-Info	Indica parámetros específicos de RTP en una orden PLAY.
Scale	Permite indicar si el contenido se quiere ver a velocidad normal, más rápida o más lenta.
Session	Identificador de la sesión.
Speed	Pide que la transferencia de datos se haga a una determinada velocidad.
Transporte	Protocolo de transporte utilizado.
Unsupported	Características no soportadas por el servidor.

12.2. Anexo 2. El formato de los paquetes RTCP

En este anexo, se describen los datos que contiene cada uno de los paquetes del protocolo RTCP.

12.2.1. Informe de emisor (SR)

La figura siguiente muestra los campos del paquete de tipo SR:



Este tipo de paquete tiene tres secciones y una cuarta, opcional, que son las extensiones y que están definidas por los perfiles.

La primera sección es la cabecera y tiene ocho octetos. Los campos son los siguientes.

- Versión (V): 2 bits. Identifica la versión de RTP y es la misma que para los paquetes de datos. La versión actual es la 2.

- Relleno (P): 1 bit. Si el bit de relleno está activado, este paquete RTCP tiene octetos de relleno que no forman parte de la información de control, pero que están incluidos en la longitud del paquete. El último octeto de relleno indica cuántos octetos se tienen que ignorar, incluido el mismo. En un paquete compuesto, sólo se debe añadir relleno a los paquetes individuales.

- Contador de recepción de informes (RC): 5 bits. El número de bloques de recepción de informes que hay en este paquete. 0 también es válido.

- Tipo de paquete (PT): 8 bits. Contiene una constante con valor 200 para identificar que se trata de un paquete de tipo SR.

- Longitud: 16 bits. Longitud del paquete en palabras de 32 bits menos 1, incluyendo cabecera y relleno.

- SSRC: 32 bits. Identificador de la fuente de sincronización del creador de este paquete SR.

La segunda sección, la sección del emisor, tiene 20 octetos de longitud y está presente en cualquier paquete de tipo SR. Hace un resumen de los datos transmitidos por este emisor. Los campos son los siguientes.

- *NTP timestamp*: 64 bits. Indica el tiempo en el que este informe se envió. Se puede utilizar para calcular el tiempo de propagación de la información si se combina con *timestamps* como paquetes de tipo RR.

- *RTP timestamp*: 32 bits. Tiene el mismo valor que el campo anterior, pero con las mismas unidades que los *timestamps* de los paquetes RTP.

- Número de paquetes enviados por el emisor: 32 bits. El número de paquetes RTP de datos transmitidos por el emisor desde que se inició la transmisión hasta el envío de este paquete.

- Número de octetos enviados por el emisor: 32 bits. Número total de octetos de datos (sin incluir cabeceras ni relleno) transmitidos en paquetes de datos RTP por el emisor desde que se inició la transmisión hasta el envío de este paquete.

La tercera sección contiene 0 o más bloques de informes de recepción, dependiendo del número de fuentes recibidas por este emisor desde el último informe. Cada bloque contiene las estadísticas siguientes.

- SSRC_n (identificador de la fuente): 32 bits. El identificador de la fuente SSRC a la cual pertenece la información enviada en este bloque.

- Fracción de pérdidas: 8 bits. La fracción de paquetes RTP perdidos por la fuente SSRC_n desde que se envió el último paquete de tipo SR o RR.

- Número acumulado de paquetes perdidos: 24 bits. El número total de paquetes perdidos desde el inicio de la recepción.

- Número de secuencia más alto recibido: 32 bits. Los 16 bits más altos contienen el número de secuencia más alto recibido y el resto son extensiones al número de secuencia.

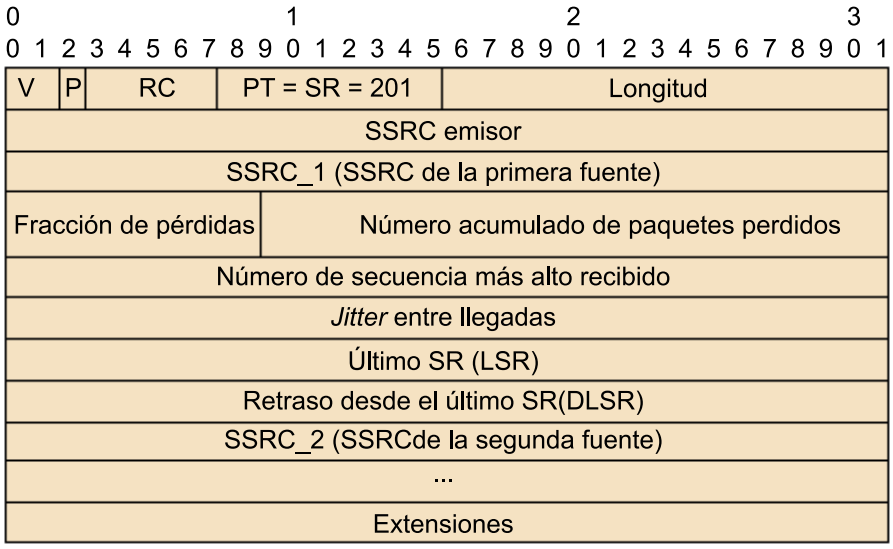
- *Jitter* entre llegadas: 32 bits. Una estimación estadística de la variancia del tiempo entre llegadas de los paquetes de datos RTP, medido en unidades de *timestamp* y expresado como un entero sin signo.

- Último SR (LSR): 32 bits. La mitad de los 64 bits recibidos en el NTP *timestamp* del último paquete de tipo SR recibido desde la fuente SSRC_n.

- Retraso desde el último SR (DLSR): 32 bits. El retraso entre el último SR recibido desde SSRC_n y el envío de este paquete de información.

12.2.2. Informe de receptor (RR)

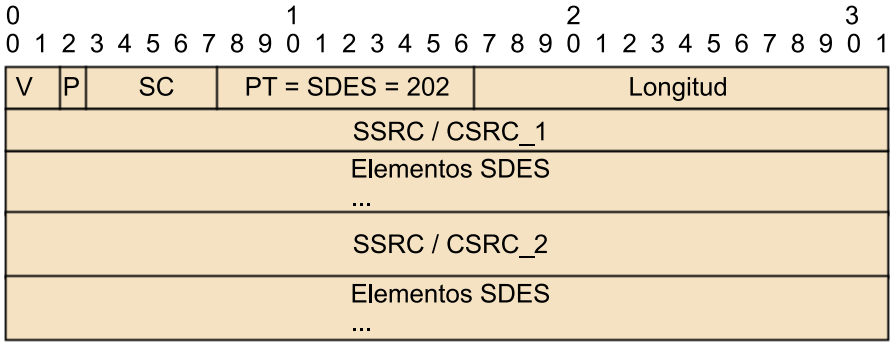
La figura siguiente muestra los campos del paquete de tipo RR:



El formato de este tipo de paquete es igual al del paquete SR, excepto por el hecho de que el tipo de paquete tiene la constante 201 y la información del emisor (*timestamps* NTP y RTP y número de paquetes y octetos enviados) no está.

12.2.3. Descripción de elementos de la fuente (SDES)

La figura siguiente muestra los campos del paquete de tipo SDES:



Este tipo de paquete tiene una estructura en tres niveles compuesta por una cabecera y cero o más porciones, cada una de estas formada por elementos que describen la fuente identificada por la porción.

La primera sección es la cabecera y tiene ocho octetos. Los campos son los siguientes.

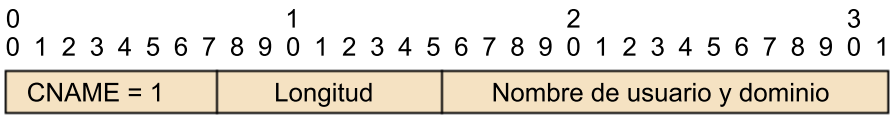
- Versión (V), relleno (P) y longitud tienen el mismo significado que los definidos por el paquete SR.

- Número de fuentes emisoras (SC): 5 bits. El número de porciones SSRC/CRSC por las cuales son contenidas en un paquete SDES.

- Tipo de paquete (PT): 8 bits. Contiene la constante 202.

- Cada porción contiene un identificador de SSRC/CSRC, seguido de una lista de elementos SDES. Cada elemento consiste en un campo de 8 bits que indica el tipo, un campo de 8 bits que es un contador del número de octetos que indica la longitud del campo (sin la cabecera) y el texto del elemento, que no puede pasar de 255 octetos.

Como ejemplo de elemento SDES, veremos cómo se representa el campo CNAME.



El campo CNAME tiene que ser único y hace enlace entre un identificador SSRC (que puede cambiar) y el identificador de la fuente, que debe ser constante.

12.2.4. Paquete de cierre (BYE)

La figura siguiente muestra los campos del paquete de tipo BYE:

El paquete BYE indica que uno o más fondos ya no están activos. Los campos son los siguientes.

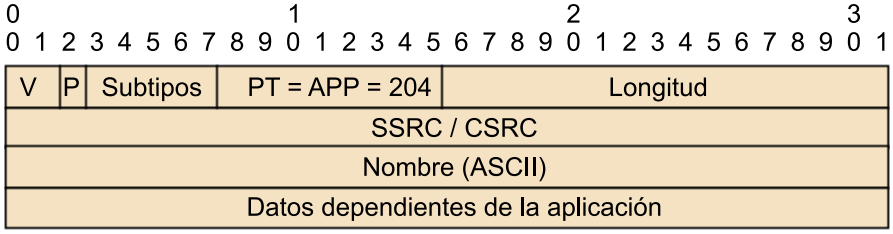
- Versión (V), relleno (P) y longitud tienen el mismo significado que los definidos por el paquete SR.

- Número de fuentes emisoras (SC): 5 bits. El número de porciones SSRC/CRSC por las cuales son contenidas en un paquete SDES.

- Tipo de paquete (PT): 8 bits. Contiene la constante 203.

12.2.5. Paquete definido por la aplicación (APP)

La figura siguiente muestra los campos del paquete de tipo APP:



El paquete APP está pensado para que las aplicaciones añadan funcionalidad. Los campos son:

- Versión (V), relleno (P) y longitud tienen el mismo significado que los definidos por el paquete SR.

- Subtipo: 5 bits. Puede utilizarse para definir un subtipo de aplicaciones.

- Tipo de paquete (PT): 8 bits. Contiene la constante 204.

- Nombre: 8 octetos. Nombre escogido por la persona que ha definido los paquetes de tipo APP, para diferenciarlos de otros paquetes de tipo APP.

- Datos dependientes de la aplicación: longitud variable. Este campo puede aparecer o no. Es directamente interpretado por la aplicación y no por RTP. Tiene que ser múltiplo de 32 bits.

12.3. Anexo 3. El protocolo SIP

12.3.1. Códigos de estado

Los códigos de estado son de 5 tipos: 1XX, respuestas provisionales, 2XX, petición lograda, 3XX, redireccionamiento, 4XX, error en la petición del cliente, 5XX, error en el servidor y 6XX, errores globales (específicos del SIP).

Muchos de los códigos de estado están heredados del protocolo HTTP, pero también hay códigos específicos del SIP. Los códigos HTTP que no tengan sentido en SIP no se deben utilizar.

Tipo de código	Valor	Descripción
Respuestas provisionales	100	Indica que el siguiente *host* (salto, *hop*, en inglés) ha recibido la petición.
Respuestas provisionales	180	Indica que la petición INVITE se ha recibido en el otro lado y se avisa al usuario.
Respuestas provisionales	181	La llamada se está redirigiendo.
Respuestas provisionales	182	La llamada se ha puesto en la cola, porque el receptor de la llamada no está disponible.
Respuestas provisionales	183	Información sobre el progreso de la llamada.
Éxito	200	La petición ha ido bien.
Redirección	300	Indica que hay múltiples opciones en la localización de la dirección del usuario al que se llama.
Redirección	301	El usuario ha cambiado de dirección y se le tiene que llamar a la nueva dirección indicada en la cabecera Contact.
Redirección	302	El usuario ha cambiado de dirección y se le tiene que llamar a la nueva dirección indicada en la cabecera Contact.
Redirección	305	Se tiene que utilizar un *proxy*.
Redirección	380	La llamada no ha sido posible, pero hay servicios alternativos disponibles.
Redirección	400	Formato de petición erróneo.
Redirección	401	El usuario necesita autenticarse.

Tipo de código	Valor	Descripción
Redirección	402	Reservado para uso futuro.
Error parte cliente	403	El servidor no quiere servir la petición del cliente, aunque tenía un formato correcto.
Error parte cliente	404	El usuario a quien se ha llamado no se encuentra en el servidor.
Error parte cliente	405	El método pedido no se puede ejecutar para la dirección indicada.
Error parte cliente	406	El recurso no es aceptable según los valores enviados en la cabecera Accept.
Error parte cliente	407	El cliente se tiene que autorizar delante de un *proxy*.
Error parte cliente	408	El servidor no ha podido producir una respuesta a tiempo.
Error parte cliente	410	El recurso ya no está disponible en el servidor.
Error parte cliente	413	El tamaño de la entidad enviada por el cliente es demasiado grande para que el servidor la pueda tratar.
Error parte cliente	414	El tamaño del URI es demasiado grande para que el servidor lo pueda tratar.
Error parte cliente	415	El tipo de datos del contenido no está soportado por el servidor.
Error parte cliente	416	La petición no se puede servir, porque el esquema indicado en el URI del recurso no se reconoce.
Error parte cliente	420	El servidor no reconoce las extensiones indicadas en la cabecera. Proxy-Require
Error parte cliente	421	Los programas necesitan una extensión específica para poder servir la petición.
Error parte cliente	423	El servidor rechaza la petición, porque expira demasiado pronto.
Error parte cliente	480	Se ha podido llegar hasta el servidor, pero el usuario al que se ha llamado no está disponible temporalmente.
Error parte cliente	481	La transacción no existe.
Error parte cliente	482	Se ha detectado un bucle en la llamada.
Error parte cliente	483	Se ha pasado el máximo de *hosts* indicados en la cabecera MaxHops.
Error parte cliente	484	La dirección del usuario al que se ha llamado es incompleta.
Error parte cliente	485	La dirección es ambigua.
Error parte cliente	486	El usuario no acepta la llamada.
Error parte cliente	487	La petición se ha acabado con un CANCEL o un BYE.
Error parte cliente	488	La petición no es aceptable, por algún parámetro indicado. a la sesión
Error parte cliente	491	Hay otra petición pendiente.
Error parte cliente	493	El cuerpo del mensaje lleva un tipo MIME que el servidor no puede tratar.
Error parte servidor	500	Error interno del servidor, que hace que no pueda servir la petición.
Error parte servidor	501	El servidor no tiene implementada la funcionalidad necesaria para servir la petición.

Tipo de código	Valor	Descripción
Error parte servidor	502	El servidor recibe un error cuando hace de *proxy* o *gateway*.
Error parte servidor	503	El servicio está temporalmente no disponible.
Error parte servidor	504	El servidor no recibe una respuesta de un servidor externo a tiempo.
Error parte servidor	505	El servidor no soporta la versión del protocolo indicada por el cliente.
Error parte servidor	513	El mensaje es demasiado largo para que el servidor lo pueda procesar.
Error general	600	El usuario llamado no está disponible.
Error general	603	El usuario llamado no quiere aceptar la llamada.
Error general	604	El usuario llamado no existe.
Error general	606	La petición no es aceptable, por algún parámetro indicado en la sesión.

12.3.2. Cabeceras

En la tabla siguiente se describen las cabeceras del protocolo SIP que son específicas de este protocolo. El resto están definidas en el RFC de HTTP:

Cabecera	Descripción
Alert-Info	Tono de llamada alternativo.
Allow	Métodos soportados.
Authentication-Info	Información por autenticación mutua
Authorization	Credenciales de autorización del cliente.
Call-ID	Identificación de una llamada o de una serie de los registros de un cliente.
Call-Info	Información adicional sobre los participantes en la llamada.
Contact	Contiene una URI que puede tener diferentes significados según el contexto.
Content-Disposition	Indicación de la organización del cuerpo del mensaje, en términos de las partes que pueda tener y los formatos MIME.
Content-Encoding	Codificación preferida por el contenido.
Content-Length	Longitud del contenido enviado en el cuerpo del mensaje.
Cseq	Número de secuencia asociado a una petición.
Date	Fecha y hora en los que se creó el mensaje.
Error-Info	Información adicional sobre el error expresado en el código de respuesta.
Expires	Momento en el que el contenido estará obsoleto.
From	Iniciador de la petición.
In-Reply-To	Referencia a los identificadores de la llamada asociados a la llamada actual.
Max-Forwards	Límite al número de *proxies* o *gateways* que pueden reenviar la petición.

Cabecera	Descripción
Min-Expires	Intervalo de refresco mínimo.
Organization	Nombre de la organización a la cual pertenece quien genera la petición.
Priority	Prioridad de la llamada desde el punto de vista del cliente.
Proxy-Authorization	Identificación del cliente frente a un *proxy* que pide autenticación.
Proxy-Require	Requerimientos que debe tener el *proxy*.
Record-Route	El *proxy* llena esta cabecera para indicar que las futuras peticiones tienen que pasar por el mismo.
Reply-To	URI de retorno que no debe ser en absoluto el mismo que el de la cabecera From.
Require	Características esperadas del programa de usuario.
Retry-After	Indicación de cuándo el servicio volverá a estar disponible.
Route	Descripción de los *proxies* por los cuales tendría que pasar la petición.
Server	Descripción del *software* que utiliza el servidor.
Subject	Tema de la llamada.
Supported	Extensiones soportadas por el programa de usuario.
Timestamp	Indicación de cuándo el programa cliente efectuó la llamada.
To	Receptor de la llamada.
Unsupported	Descripción de las características no soportadas.
Via	Indicación del camino que ha seguido la petición y que se tiene que utilizar para las respuestas.
WWW-Authenticate	Valor de respuesta de autenticación.

Resumen

En este módulo, se ha empezado describiendo las arquitecturas de aplicaciones más habituales en Internet: cliente servidor y de igual a igual, así como los requerimientos de las aplicaciones orientadas al envío de datos o de mensajes.

A continuación, se han descrito las aplicaciones distribuidas más habituales en Internet y diferentes estándares que están relacionados: Web y HTTP, DNS, SMTP, FTP, XMPP, Telnet y SSH, Gnutella, KaZaA, BitTorrent y eDonkey.

Después, se han introducido los requerimientos de las aplicaciones multimedia en red.

Finalmente, se han visto una serie de técnicas y protocolos para trabajar con contenidos multimedia en Internet, tanto por *streaming* de audio y vídeo: RTSP; como por aplicaciones interactivas en tiempo real: RTP, RTCP, SIP, H3.23 y Skype.

Bibliografía

Kurose, J. F.; Ross, W. K. *Computer Networking* (5.ª edición). Boston: Addison Wesley. ISBN 0-321-49770-8.

Este libro proporciona una visión completa de los diferentes aspectos relacionados con las redes de ordenadores. En este módulo interesa el capítulo 2 ("*Application Layer*"), en el que se pueden encontrar los conceptos básicos de la Web, el correo electrónico, la transferencia de ficheros y el servicio de directorio de Internet. También hay un apartado de igual a igual que puede complementar la visión de los sistemas de igual a igual dada en este módulo.

Keagy, S. (2000). *Integrating Voice and Data Networks*. Indianapolis: Cisco Press.

En este libro se describe el protocolo SIP.

Niederst, J. (2006). *Web Design in a Nutshell* (3.ª ed.). Sebastopol: O'Reilly.

En este libro hay dos secciones dedicadas a los formatos de audio y vídeo que se transmiten por Internet.

Hersent, O.; Gurle, D.; Petit, J. P. (1999). *IP Telephony: Packet-Based Multimedia Communications Systems*. Indianapolis: Addison Wesley Professional.

En este libro, se describen los protocolos SIP y H.323 para la implementación de telefonía sobre IP.

Artículos

Baset, S.; Schulzrinne, H. (2004). "An Analysis of the Skype Peer-To-Peer Internet Telephony Protocol".

Este artículo describe la arquitectura y el protocolo de Skype basados en la captura de los datos enviados por la red que han hecho los autores.

Lua y otros (2005). "A survey and comparison of peer-to-peer overlay network schemes". *IEEE Communications Surveys&Tutorials* (vol. 7, núm. 2).

En este artículo encontraréis un resumen de cómo funcionan las redes superpuestas de igual a igual más populares, así como una comparativa entre las mismas. También encontraréis más detalle de los sistemas explicados en este módulo.