DISTRIBUTED SYSTEMS

Concepts and Design

Fifth Edition

George Coulouris
Jean Dollimore
Tim Kindberg
Gordon Blair

Esta página se ha dejado intencionadamente en blanco

SISTEMAS DISTRIBUIDOS

Conceptos y Diseño

Quinta edición

Esta página se ha dejado intencionadamente en blanco

SISTEMAS DISTRIBUIDOS

Conceptos y Diseño

Quinta edición

George Coulouris

Universidad de Cambridge

Jean Dollimore

ex integrante de Queen Mary,

Universidad de Londres

Tim Kindberg

2 materia de medios

Gordon Blair

Universidad de Lancaster

Addison-Wesley

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Director editorial: *Marcia Horton*
Editor en jefe: *Michael Hirsch*
Editor ejecutivo: *Matt Goldstein*
Asistente editorial: *Chelsea Campana*
Vicepresidente de marketing: *Patrice Jones*
Gerente de Marketing: *Yezan Alayan*
Coordinador de Marketing: *Kathryn Ferranti*
Vicepresidente, Producción: *Vince O'Brien*
Jefe de redacción: *Jeff Holcomb*
Producción Proyecto Senior Manager: *Marilyn Lloyd*
Supervisor Senior de Operaciones: *Alan Fischer*
Comprador de fabricación: *Lisa McDowell*
Director de arte: *Jayne Conte*
Diseñador de la portada: *Suzanne Duda*
Imagen de portada: *Cielo: © amygdala_imagery; Kite: © Alamy;*
Teléfono móvil: © yasinguneysu / IStock
Medios Editor: *Daniel Sandin*
Media Manager del proyecto: *Wanda Rockwell*
Impresora / Carpeta: *Edwards hermanos*
Cubierta de la impresora: *Lehigh-Phoenix color*
La composición y diseño de los autores utilizando FrameMaker

Biblioteca del Congreso de datos Catalogación por la publicación disponibles a pedido

impresión 1
10 9 8 7 6 5 4 3 2 1-EB-15 14 13 12 11

Addison-Wesley
is an imprint of

PEARSON

www.pearsonhighered.com

ISBN 10: 0-13-214301-1
ISBN 13: 978-0-13-214301-1

CONTENIDO

PREFACIO

Esta quinta edición de nuestro libro de texto aparece en un momento en el Internet y la Web continúan creciendo y tener un impacto en todos los aspectos de nuestra sociedad. Por ejemplo, el capítulo introductorio del libro señala su impacto en áreas de aplicación tan diversos como las finanzas y el comercio, las artes y el entretenimiento y la aparición de la sociedad de la información en general. También se destacan los requisitos muy exigentes de los dominios de aplicación, tales como búsqueda en la web y los juegos multijugador en línea. Desde una perspectiva de sistemas distribuidos, estos desarrollos están colocando nuevas demandas sustanciales en la infraestructura del sistema subyacente en cuanto a la gama de aplicaciones y cargas de trabajo y los tamaños de los sistemas soportados por muchos sistemas modernos.

Nuevo en la quinta edición

Los nuevos capítulos:

La comunicación indirecta: Cubriendo la comunicación de grupo, y la publicación-suscripción estudios de caso sobre JavaSpaces, JMS, WebSphere y colas de mensajes.

Distribuidos Objetos y Componentes: Cubriendo middleware basado en componentes y estudios de caso sobre Enterprise JavaBeans, fractal y CORBA.

El diseño de sistemas distribuidos: Dedicado a un importante estudio de caso sobre la nueva Google infraestructura.

Temas añaden a otros capítulos: La computación en nube, virtualización de red, sistema de virtualización de funcionamiento, el paso de mensajes de interfaz, de igual a igual-estructurado, espacios de tuplas, el acoplamiento débil en relación con los servicios web.

Otros nuevos casos de estudio: Skype, Gnutella, TOTA, L 2 Imbo, BitTorrent, Fin sistema de multidifusión.

Vea la tabla en la página XV para más detalles de los cambios.

(Que conduce a radicalmente diferentes arquitecturas físicas), la necesidad de apoyar los servicios multimedia y el surgimiento del paradigma de la computación en nube, que desafía nuestra perspectiva de los servicios de sistemas distribuidos.

El libro tiene como objetivo proporcionar una comprensión de los principios en que se basan la Internet y otros sistemas distribuidos; su arquitectura, algoritmos y diseño; y la forma en que cumplen las exigencias de las aplicaciones distribuidas contemporáneos. Comenzamos con un conjunto de siete capítulos que en conjunto abarcan los bloques de construcción para un estudio de los sistemas distribuidos. Los dos primeros capítulos ofrecen una visión conceptual de la materia, mostrando las características de los sistemas distribuidos y los retos que deben ser abordados en su diseño: la escalabilidad, la heterogeneidad, la seguridad y el control de fallos siendo la más significativa. Estos capítulos también desarrollan modelos abstractos para la comprensión de la interacción de procesos, el fracaso y la seguridad. Ellos son seguidos por otros capítulos fundamentales dedicados al estudio de las redes, la comunicación entre procesos, invocación remota,

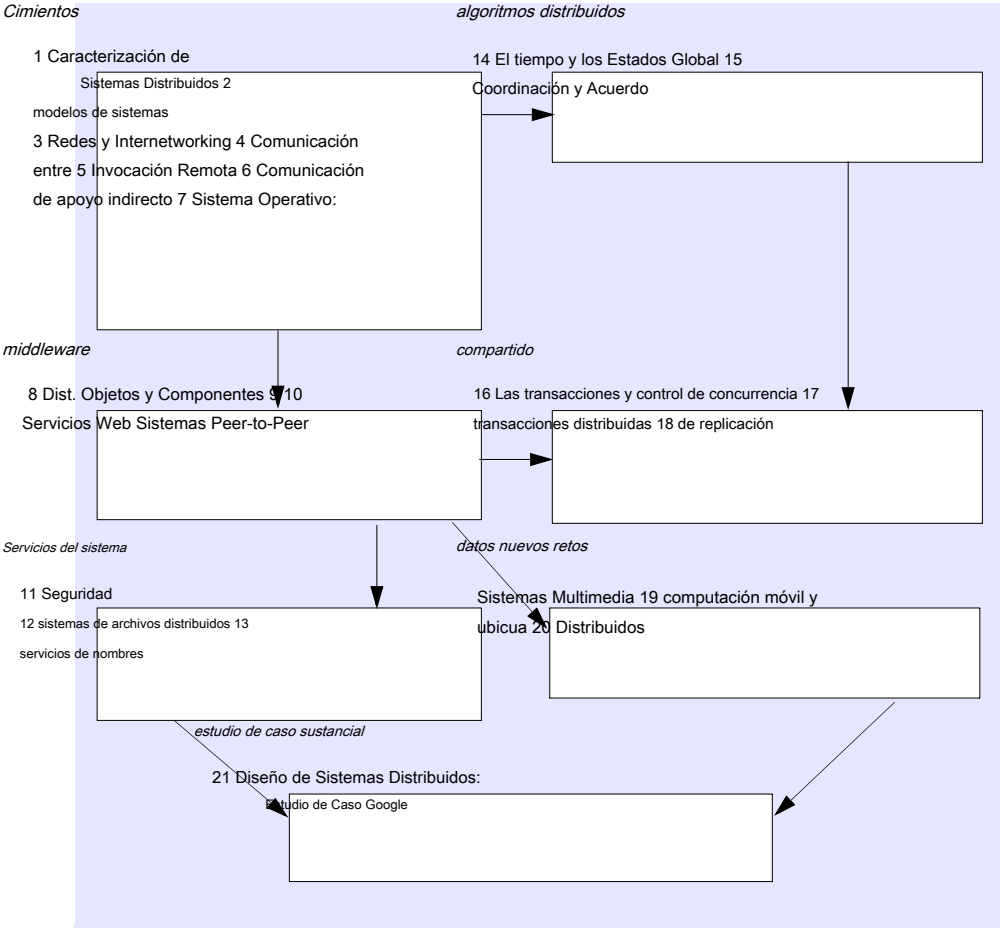
El siguiente conjunto de capítulos cubre el tema importante de middleware, que examina diferentes enfoques para el soporte de aplicaciones distribuidas, incluyendo objetos distribuidos y componentes, servicios web y soluciones alternativas peer-to-peer. a continuación, cubrimos los temas bien establecidos de seguridad, sistemas de archivos distribuidos y denominación distribuido antes de pasar a los aspectos relacionados con los datos importantes, incluyendo las transacciones distribuidas y replicación de datos. Algoritmos asociados con todos estos temas se tratan a medida que surgen y también en capítulos independientes dedicadas a la sincronización, coordinación y concertación.

El libro culmina en los capítulos que abordan las áreas emergentes de los sistemas multimedia móviles y la computación ubicua y distribuidos antes de presentar un estudio de caso importante se centra en el diseño e implementación de la infraestructura de sistemas distribuida que soporta Google tanto en términos de funcionalidad de búsqueda básica y la gama cada vez mayor de servicios adicionales ofrecidos por Google (por ejemplo, Gmail y Google Earth). Este último capítulo tiene un papel importante en la ilustración de cómo todos los conceptos arquitectónicos, algoritmos y tecnologías introducidas en el libro pueden reunirse en un diseño global coherente para un dominio de aplicación dada.

Propósitos y lectores

El libro está pensado para su uso en cursos de postgrado universitarios y de introducción. Se puede igualmente ser utilizado para auto-estudio. Tomamos un enfoque de arriba hacia abajo, para abordar las cuestiones que deben resolverse en el diseño de sistemas distribuidos y describir los enfoques exitosos en forma de modelos abstractos, algoritmos y estudios de casos detallados de sistemas de uso frecuente. Cubrimos el campo con suficiente profundidad y amplitud para que los lectores puedan ir a estudiar la mayoría de los trabajos de investigación en la literatura sobre sistemas distribuidos.

Nuestro objetivo es hacer que el sujeto accesible a los estudiantes que tienen un conocimiento básico de la programación orientada a objetos, sistemas operativos y arquitectura de computadores primaria. El libro incluye la cobertura de aquellos aspectos de las redes informáticas pertinentes a los sistemas distribuidos, incluyendo las tecnologías subyacentes de Internet y de área amplia, área local y redes inalámbricas. Algoritmos y las interfaces se presentan en todo el libro en Java o, en algunos casos, ANSI C. Por razones de brevedad y claridad de la presentación, también se utiliza una forma de pseudo-código derivado de Java / C.



Organización del libro

El diagrama muestra los capítulos del libro de menos de siete áreas temáticas principales. Se pretende proporcionar una guía a la estructura del libro y para indicar las rutas de navegación recomendados para los instructores que deseen proporcionar o lectores que desean lograr, la comprensión de los diversos subcampos de diseño del sistema distribuido.

referencias

La existencia de la World Wide Web ha cambiado la forma en la que un libro como este puede estar relacionado con material de origen, incluyendo trabajos de investigación, las especificaciones y normas técnicas. Muchos de los documentos originales están ahora disponibles en la Web; algunos están disponibles sólo allí. Por razones de brevedad y legibilidad, empleamos una forma especial de referencia al material de banda que vagamente se parece a una URL: **referencias tales como [www.omg.org] Y [me www.rsasecurity.com] Se refieren a la documentación que está disponible**

Sólo en la Web. Ellos se pueden consultar en la lista de referencias al final del libro, pero no se dan las URL completas sólo en una versión en línea de la lista de referencia en el sitio web del libro, www.cdk5.net/refs donde toman la forma de hacer clic en enlaces. Ambas versiones de la lista de referencia incluyen una explicación más detallada de este esquema.

Cambios relativos a la cuarta edición

Antes de embarcarse en la redacción de esta nueva edición, se llevó a cabo un estudio de los profesores que utiliza la cuarta edición. A partir de los resultados, se identificó el nuevo material requerido y un número de cambios a realizar. Además, nos dimos cuenta de la creciente diversidad de sistemas distribuidos, particularmente en términos de la variedad de enfoques arquitectónicos disponibles para los desarrolladores de sistemas distribuidos hoy. Esto requiere cambios significativos en el libro, especialmente en los capítulos anteriores (fundamentales).

En general, esto llevó a nuestro escrito tres capítulos totalmente nuevos, hacer cambios sustanciales a una serie de otros capítulos y haciendo numerosas inserciones en todo el libro de doblar en el nuevo material. Muchos de los capítulos se han cambiado para reflejar la nueva información que se ha vuelto disponible de los sistemas descritos. Estos cambios se resumen en la siguiente tabla. Para ayudar a los profesores que han utilizado la cuarta edición, siempre que sea posible, hemos conservado la estructura adoptada a partir de la edición anterior. Cuando el material se ha eliminado, hemos colocado este en nuestro sitio Web compañero junto con material extraído de las ediciones anteriores. Esto incluye los estudios de caso sobre ATM, la comunicación entre procesos en UNIX, CORBA (una versión acortada de la que permanece en el capítulo 8),

Algunos de los capítulos del libro, como el nuevo capítulo sobre la comunicación indirecta (capítulo 6), cubrir una gran cantidad de material. Los maestros pueden elegir para cubrir el amplio espectro antes de elegir dos o tres técnicas para examinar con más detalle (por ejemplo, la comunicación de grupo, dado su papel fundamental, y la publicación-suscripción o un mensaje de colas, dada su prevalencia en los sistemas distribuidos comerciales).

El ordenamiento capítulo ha sido modificado para acomodar el nuevo material y para reflejar los cambios en la importancia relativa de ciertos temas. Para una mejor comprensión de algunos temas lectores pueden encontrar que es necesario seguir una referencia hacia adelante. Por ejemplo, hay material en el capítulo 9 en técnicas de seguridad XML que harán más sentido una vez que se han absorbido las secciones que hace referencia en el capítulo 11 de seguridad.

Expresiones de gratitud

Estamos muy agradecidos a los siguientes maestros que participaron en nuestra encuesta: Guohua Cao, José Fortes, Bahram Khalili, George blanco, Jinsong Ouyang, Joanne Holliday, George K. Thiruvathukal, Joel Wein, Tao Xie y Zhou Xiaobo.

Nos gustaría agradecer a las siguientes personas que revisaron los nuevos capítulos o proporcionan otro tipo de ayuda sustancial: Rob Allen, Roberto Baldoni, John Bates, Tom Berson, Lynne Blair, Geoff Coulson, Paul Grace, Andrew Herbert, David Hutchison, Laurent Mathy, Rajiv Ramdhany, Richard Sharp, Jean-Bernard Stefani, Rip Sohan, Francois

Los nuevos capítulos:

6 La comunicación indirecta	Incluye eventos y notificación de 4ª edición.
8 y objetos distribuidos componentes	Incluye una versión precisó del estudio de caso CORBA de la 4ª edición.
21 Diseño de Sistemas Distribuidos	Incluye un importante estudio de caso nuevo en Google

Capítulos que han experimentado cambios sustanciales:

1 Caracterización de DS	*reestructuración significativa de material* Nueva Sección 1.2: Ejemplos de sistemas distribuidos Sección 1.3.4: La computación en nube introdujo
2 modelos de sistemas	*reestructuración significativa de material* Nueva Sección 2.2: modelos Sección 2.3 Física: reescritura importante para reflejar el nuevo contenido del libro y perspectivas arquitectónicas asociadas
4 comunicación entre procesos	*varios cambios* comunicación cliente-servidor se trasladó a Nueva Capítulo 5 Sección 4.5: virtualización de red (incluye estudio de caso a través de Skype) Nueva Sección 4.6: Estudio de caso en el estudio de caso sobre MPI IPC en UNIX retira
5 invocación remota	*reestructuración significativa de material* comunicación cliente-servidor se trasladó a aquí progresión introducido desde la comunicación cliente-servidor a través de RPC para RMI Eventos y notificación trasladó al Capítulo 6

Capítulos a que el nuevo material se ha añadido / eliminado, pero sin cambios estructurales:

3 Redes y Internetworking	*varios cambios* Sección 3.5: material sobre ATM retira
Soporte 7 Sistema Operativo:	Nueva Sección 7.7: la virtualización del sistema operativo
9 Servicios Web	Sección 9.2: Discusión añadido el acoplamiento débil
10 Sistemas Peer-to-Peer	Nueva Sección 10.5.3: no estructurados peer-to-peer (incluyendo un nuevo estudio de caso sobre Gnutella)
15 Coordinación y Acuerdo	El material en la comunicación de grupo se trasladó a Ch. 6
18 replicación	El material en la comunicación de grupo se trasladó a Ch. 6
Sección 19 móvil y la computación ubicua	19.3.1: Nuevo material en espacios de tuplas (TOTA y L 2 Imbo)
20 Sistemas Multimedia Distribuidas	Sección 20.6: estudio de casos nuevos añadidos en BitTorrent y Fin del sistema de multidifusión

Los capítulos restantes han recibido sólo modificaciones menores.

Taiani, Peter Triantafillou, Gareth Tyson y el difunto Sir Maurice Wilkes. También nos gustaría agradecer al personal de Google que proporcionó información sobre los fundamentos de diseño de Google Infraestructura, a saber: Mike Burrows, Tushar Chandra, Walfredo Cirne, Jeff Dean, Sanjay Ghemawat, Andrea y John Kirmse Reumann.

Nuestro editor, Rachel Head también proporcionó apoyo excepcional.

sitio web

Al igual que antes, seguimos manteniendo un sitio web con una amplia gama de material diseñado para ayudar a los maestros y lectores. Este sitio web se puede acceder a través de la URL:

www.cdk5.net

El sitio web incluye:

Guía del instructor: Proporcionamos material para los maestros que comprenden el apoyo a:

- obra completa del libro disponibles como archivos de PowerPoint;
- capítulo por capítulo consejos de enseñanza;
- soluciones a los ejercicios, protegido por una contraseña disponible sólo para los maestros.

Lista de referencia: La lista de referencias que se pueden encontrar en la parte final del libro se replica en el sitio web. La versión web de la lista de referencias incluye enlaces activos para el material que está disponible en línea.

lista de erratas: Se mantiene una lista de errores conocidos en el libro, con las correcciones. Los errores serán corregidos cuando se imprimen nuevas impresiones y se proporcionará una lista de erratas separado para cada impresión. (Los lectores pueden reportar los errores aparentes que se encuentran a la dirección de correo electrónico a continuación.)

Material suplementario: Mantenemos un conjunto de material complementario para cada capítulo. Esto consiste en código fuente de los programas en el libro y material de lectura pertinente, que estuvo presente en las ediciones anteriores del libro, pero fue retirado por razones de espacio. Las referencias a este material complementario aparecen en el libro con enlaces como www.cdk5.net/ipc (La dirección URL de material complementario en relación con el capítulo Comunicación entre procesos). Dos capítulos enteros de la 4ª edición no están presentes en éste; que se puede acceder a las direcciones URL:

CORBA Estudio de caso www.cdk5.net/corba

Memoria compartida distribuida www.cdk5.net/dsm

George Coulouris
Jean Dollimore Tim
Kindberg Gordon
Blair
Londres, Bristol y Lancaster, 2011
authors@cdk5.net

1

CARACTERIZACIÓN de sistemas distribuidos

Un sistema distribuido es una en la que los componentes situados en ordenadores en red se comunican y coordinan sus acciones sólo mediante el paso de mensajes. Esta definición lleva a las siguientes características especialmente importantes de los sistemas distribuidos: concurrencia de los componentes, la falta de un reloj global y las fallas de componentes independientes.

Nos fijamos en varios ejemplos de aplicaciones modernas distribuidos, incluyendo búsqueda en la web, juegos multijugador en línea y sistemas de comercio financiero, y también examinar las tendencias subyacentes clave que impulsan los sistemas distribuidos hoy en día: la naturaleza penetrante de las redes modernas, la aparición de la computación móvil y ubicua, la creciente importancia de los sistemas multimedia distribuidos, y la tendencia a la visualización de los sistemas distribuidos como una utilidad. Luego, el capítulo destaca el intercambio de recursos como la motivación principal para la construcción de sistemas distribuidos. Los recursos pueden ser gestionados por los servidores y accesibles a los clientes o pueden ser encapsulados como objetos y acceder a otros objetos de cliente.

Los retos derivados de la construcción de sistemas distribuidos son la heterogeneidad de sus componentes, la apertura (que permite que los componentes que se añaden o reemplazado), seguridad, escalabilidad - la capacidad de trabajar bien cuando la carga o el número de usuarios aumenta - el control de fallos, concurrencia de los componentes, la transparencia y proporcionar calidad de servicio. Por último, la Web se discute como un ejemplo de un sistema distribuido a gran escala y se introducen sus características principales.

1.1 Introducción

Redes de computadoras están por todas partes. Internet es uno, al igual que las muchas redes de las que se compone. redes de telefonía móvil, redes corporativas, redes de fábricas, redes de campus, redes domésticas, redes en el automóvil - todos ellos, tanto por separado como en combinación, compartir las características esenciales que las hacen temas relevantes para el estudio bajo el título *sistemas distribuidos*. En este libro se pretende explicar las características de los ordenadores en red que los diseñadores de sistemas de impacto y ejecutores y presentar los principales conceptos y técnicas que se han desarrollado para ayudar en las tareas de diseño e implementación de sistemas que se basan en ellos.

Definimos un sistema distribuido como uno en el que los componentes de hardware o software ubicados en ordenadores en red se comunican y coordinan sus acciones sólo mediante el paso de mensajes. Esta simple definición cubre toda la gama de sistemas en los que ordenadores conectados en red pueden ser útilmente desplegados.

Los equipos que están conectados por una red pueden espacialmente separados por cualquier distancia. Pueden estar en continentes separados, en el mismo edificio o en la misma habitación. Nuestra definición de sistemas distribuidos tiene las siguientes consecuencias importantes:

concurrencia: En una red de ordenadores, ejecución simultánea de programas es la norma. Puedo hacer mi trabajo en mi equipo, mientras que usted hace su trabajo en el suyo, compartiendo recursos, tales como páginas web o archivos cuando sea necesario. La capacidad del sistema para manejar los recursos compartidos se puede aumentar mediante la adición de más recursos (por ejemplo. Ordenadores) a la red. Vamos a describir las formas en que esta capacidad adicional se puede desplegar de forma útil en muchos puntos en este libro. La coordinación de los programas que se ejecutan simultáneamente que comparten recursos es también un tema importante y recurrente.

No hay reloj mundial: Cuando los programas necesitan cooperar coordinan sus acciones mediante el intercambio de mensajes. Estrecha coordinación depende a menudo de una idea compartida de la hora a la que se producen las acciones de los programas. Pero resulta que hay límites a la precisión con la que los ordenadores de una red pueden sincronizar sus relojes - no existe una única noción global de la hora correcta. Esto es una consecuencia directa del hecho de que la *solamente* la comunicación es mediante el envío de mensajes a través de una red. Ejemplos de estos problemas de temporización y soluciones a los mismos se describen en el Capítulo 14.

fallos independientes: Todos los sistemas informáticos pueden fallar, y es la responsabilidad de los diseñadores de sistemas para planificar las consecuencias de posibles fallos. Los sistemas distribuidos pueden fallar en nuevas formas. Los fallos en la red el resultado en el aislamiento de los equipos que están conectados a él, pero eso no quiere decir que dejen de funcionar. De hecho, los programas en ellos pueden no ser capaces de detectar si la red ha fallado o se ha vuelto inusualmente lenta. Del mismo modo, el fracaso de una computadora, o la terminación inesperada de un programa en algún lugar del sistema (una *choque),* no se hace inmediatamente conocida a los otros componentes con los que se comunica. Cada componente del sistema puede fallar de forma independiente, dejando a los otros aún en marcha. Las consecuencias de esta característica de los sistemas distribuidos serán un tema recurrente a lo largo del libro. La principal motivación para la construcción y el uso de sistemas distribuidos deriva de un deseo de compartir recursos. El término 'recurso' es más bien abstracta, pero es mejor caracteriza a la gama de cosas que se pueden compartir de forma útil en un sistema informático en red. Eso

se extiende a partir de componentes de hardware, como discos e impresoras a entidades definidas por software, tales como archivos, bases de datos y objetos de datos de todo tipo. Incluye la corriente de fotogramas de vídeo que se desprende de una cámara de vídeo digital y la conexión de audio que una llamada de teléfono móvil representa.

El propósito de este capítulo es el de transmitir una visión clara de la naturaleza de los sistemas distribuidos y los retos que deben ser abordados con el fin de asegurar que sean exitosos. Sección 1.2 da algunos ejemplos ilustrativos de sistemas distribuidos, con la Sección 1.3 que cubre las tendencias subyacentes clave que impulsan los acontecimientos recientes. Sección 1.4 se centra en el diseño de sistemas de intercambio de recursos, mientras que la Sección 1.5 describe los principales desafíos que enfrentan los diseñadores de sistemas distribuidos: la heterogeneidad, la apertura, la seguridad, la escalabilidad, el control de fallos, concurrencia, transparencia y calidad de servicio. Sección 1.6 presenta un estudio de caso detallado de un sistema distribuido muy conocida, la World Wide Web, ilustrando cómo su diseño admite el uso compartido de recursos.

1.2 Ejemplos de sistemas distribuidos

El objetivo de esta sección es proporcionar ejemplos de motivación de los sistemas distribuidos contemporáneos que ilustran tanto el papel dominante de los sistemas distribuidos y la gran diversidad de las aplicaciones asociadas.

Como se mencionó en la introducción, las redes están en todas partes y sustentan muchos servicios cotidianos que ahora damos por sentado: la Internet y la World Wide Web asociado, búsqueda en la web, juegos en línea, correo electrónico, redes sociales, comercio electrónico, etc. Para ilustrar más este punto , considere la figura 1.1, que describe un rango seleccionado de sectores clave de aplicación comercial o social destacando algunos de los usos establecidos o emergentes asociados de la tecnología de sistemas distribuidos.

Como puede verse, los sistemas distribuidos abarcan muchos de los desarrollos tecnológicos más importantes de los últimos años y, por tanto, una comprensión de la tecnología subyacente es absolutamente esencial para el conocimiento de la informática moderna. La figura también proporciona una penetración inicial en la amplia gama de aplicaciones en uso hoy en día, desde sistemas relativamente localizadas (como se encuentra, por ejemplo, en un coche o avión) a globalscale sistemas que implican millones de nodos, de los servicios de datos centradas a processorintensive tareas , a partir de los sistemas construidos a partir de sensores muy pequeños y relativamente primitivos a los que incorporan elementos computacionales de gran alcance, de los sistemas integrados a los que apoyan al usuario una experiencia interactiva sofisticada, y así sucesivamente.

Ahora nos fijamos en ejemplos más específicos de los sistemas distribuidos para ilustrar mejor la diversidad y complejidad del hecho provisión de sistemas distribuidos hoy.

1.2.1 Búsqueda Web

Buscar en la Web se ha convertido en una industria de gran crecimiento en la última década, con cifras recientes que indican que el número total de búsquedas ha aumentado a más de 10 mil millones por mes calendario. La tarea de un motor de búsqueda en la web es indexar todo el contenido de la World Wide Web, que abarca una amplia gama de estilos de información, incluyendo páginas web, fuentes multimedia y libros (escaneadas). Esta es una tarea muy compleja, ya que las estimaciones actuales indican que la Web se compone de más de 63 mil millones de páginas y un billón web única

Figura 1.1 dominios de aplicación seleccionados y aplicaciones en red asociados

Finanzas y el comercio	El crecimiento del comercio electrónico como se ejemplifica por empresas como Amazon y eBay, y las tecnologías subyacentes a los pagos como PayPal; la aparición asociada de sistemas de difusión de información también complejos para los mercados financieros de banca en línea y el comercio y.
La sociedad de la información	El crecimiento de la World Wide Web como un repositorio de información y conocimiento; el desarrollo de motores de búsqueda como Google y Yahoo para buscar en este vasto depósito; la aparición de bibliotecas digitales y la digitalización a gran escala de las fuentes de información heredados como los libros (por ejemplo, Google Books); la creciente importancia del contenido generado por el usuario a través de sitios como YouTube, Wikipedia y Flickr; la aparición de las redes sociales a través de servicios como Facebook y MySpace.
Las industrias creativas y entretenimiento	La aparición de juegos en línea como una forma novedosa y altamente interactiva de entretenimiento; la disponibilidad de la música y el cine en el hogar a través de centros de medios en red y más ampliamente en Internet a través de streaming de contenido descargable o; el papel de contenido generado por el usuario (como se mencionó anteriormente) como una nueva forma de creatividad, por ejemplo a través de servicios como YouTube; la creación de nuevas formas de arte y entretenimiento habilitados por emergente (incluyendo tecnologías en red).
Cuidado de la salud	El crecimiento de la informática de la salud como una disciplina con su énfasis en los registros electrónicos de pacientes en línea y otros temas relacionados a la privacidad; el creciente papel de la telemedicina en el apoyo diagnóstico a distancia o servicios más avanzados, como la cirugía a distancia (incluyendo el trabajo de colaboración entre los equipos de salud); la creciente aplicación de la tecnología de redes y sistemas integrados en la vida asistida, por ejemplo para el seguimiento de las personas mayores en sus propios hogares.
Educación	La aparición de e-aprendizaje a través de, por ejemplo, herramientas basadas en la web, tales como entornos virtuales de aprendizaje; soporte asociado para el aprendizaje a distancia; apoyo para el aprendizaje colaborativo o basado en la comunidad.
Transporte y logística	El uso de las tecnologías de localización como el GPS en la ruta de búsqueda de sistemas y sistemas más generales de gestión del tráfico; el coche moderno a sí misma como un ejemplo de un sistema distribuido complejo (también se aplica a otras formas de transporte tales como aviones); el desarrollo de servicios de mapas basados en la web, tales como MapQuest, Google Maps y Google Earth.
Ciencia	La aparición de la cuadrícula como una tecnología fundamental para la e-Ciencia, incluyendo el uso de complejas redes de ordenadores para apoyar el almacenamiento, análisis y procesamiento de (a menudo muy grandes cantidades de datos científicos); el uso asociado de la cuadrícula como una tecnología que permite la colaboración mundial entre grupos de científicos.
Gestión ambiental	El uso de (en red) tecnología de sensores tanto a supervisar y gestionar el medio ambiente natural, por ejemplo para proporcionar una alerta temprana de los desastres naturales como terremotos, inundaciones o tsunamis y para coordinar la respuesta de emergencia; la recopilación y análisis de parámetros ambientales a nivel mundial para comprender mejor los fenómenos naturales complejos como el cambio climático.

direcciones. Teniendo en cuenta que la mayoría de los motores de búsqueda analizar todo el contenido de la web y luego llevar a cabo el procesamiento sofisticado en esta enorme base de datos, esta tarea en sí representa un reto importante para el diseño de sistemas distribuidos.

Google, el líder del mercado en tecnología de búsqueda en la web, ha puesto gran esfuerzo en el diseño de una infraestructura de sistema distribuido sofisticada para apoyar la búsqueda (y de hecho otras aplicaciones y servicios de Google como Google Earth). Esto representa una de las más grandes y complejas instalaciones de sistemas distribuidos en la historia de la informática y por lo tanto exige un examen minucioso. Lo más destacado de esta infraestructura incluyen:

- una infraestructura física subyacente que consiste en un gran número de equipos conectados en red instalados en centros de datos de todo el mundo;
- un sistema de archivos distribuido diseñado para soportar archivos muy grandes y muy optimizado para el estilo del uso requerido por la búsqueda y otras aplicaciones de Google (especialmente la lectura de archivos en alta y tasas sostenidas);
- un sistema distribuido de almacenamiento estructurado asociado que ofrece un rápido acceso a grandes bases de datos;
- un servicio de bloqueo que ofrece funciones de sistemas distribuidos, tales como bloqueo distribuido y el acuerdo;
- un modelo de programación que soporta la gestión de grandes cálculos paralelos y distribuidos a través de la infraestructura física subyacente. Para más detalles sobre Google distribuidos de servicios de sistemas y soporte de comunicaciones subyacentes se pueden encontrar en el capítulo 21, un estudio de caso convincente de un sistema distribuido moderna en acción.

1.2.2 juegos multijugador masivo en línea (MMOG)

Masivamente juegos multijugador en línea ofrecen una experiencia de inmersión mediante el cual un gran número de usuarios interactúan a través de Internet con un mundo virtual persistente. Los principales ejemplos de este tipo de juegos de Sony incluyen EverQuest II y EVE Online de la compañía finlandesa CCP Games. Tales mundos han aumentado significativamente en la sofisticación y ahora incluyen, arenas de juego complejas (por ejemplo, EVE, en línea consiste en un universo con más de 5.000 sistemas de estrellas) y los sistemas sociales y económicos múltiples. El número de jugadores también está aumentando, con sistemas capaces de soportar más de 50.000 jugadores en línea simultáneas (y el número total de jugadores tal vez diez veces esta cifra).

La ingeniería de MMOGs representa un reto importante para las tecnologías de sistemas distribuidos, sobre todo debido a la necesidad de tiempos de respuesta rápidos para preservar la experiencia del usuario del juego. Otros desafíos incluyen la propagación en tiempo real de los eventos a los muchos jugadores y mantener una visión coherente del mundo compartido. Por tanto, este es un excelente ejemplo de los retos modernos diseñadores de sistemas distribuidos.

Se han propuesto una serie de soluciones para el diseño de juegos multijugador masivos en línea:

- Tal vez sorprendentemente, el mayor juego en línea, EVE Online, utiliza una *Servidor de cliente* arquitectura en la que se mantiene una sola copia del estado del mundo en una

servidor centralizado y se accede por los programas de cliente que se ejecutan en las consolas de los jugadores u otros dispositivos. Para apoyar a un gran número de clientes, el servidor es una entidad compleja en sí misma consiste en una arquitectura de cluster con cientos de nodos informáticos (este enfoque cliente-servidor se analiza con más detalle en la Sección
1.4 y racimo enfoques se discuten en la Sección 1.3.4). La arquitectura centralizada ayuda significativamente en términos de la gestión del mundo virtual y la copia única también alivia las preocupaciones de consistencia. El objetivo es entonces para asegurar una respuesta rápida a través de la optimización de los protocolos de red y asegurar una respuesta rápida a los eventos entrantes. Para apoyar esto, la carga se reparte mediante la asignación de los sistemas de estrellas " individuales a los ordenadores particulares dentro de la agrupación, con sistemas de estrellas altamente cargadas que tengan su propia computadora dedicada y otros que comparten una computadora. Los eventos entrantes se dirigen a los equipos adecuados dentro de la agrupación mediante el seguimiento del movimiento de jugadores entre los sistemas estelares.

- Otros MMOGs adoptan arquitecturas más distribuidos donde el universo se repartió a través de un número (potencialmente muy grande) de servidores que también puede ser distribuido geográficamente. Los usuarios se asignan dinámicamente un servidor en particular sobre la base de los patrones de uso actuales y también los retrasos en la red al servidor (en base a la proximidad geográfica, por ejemplo). Este estilo de arquitectura, que es adoptado por EverQuest, es naturalmente extensible mediante la adición de nuevos servidores.

- La mayoría de los sistemas comerciales adoptan uno de los dos modelos presentados anteriormente, pero los investigadores también están buscando ahora en arquitecturas más radicales que no se basan en principios cliente-servidor, sino que adoptan enfoques completamente descentralizados basados en la tecnología peer-to-peer en el que cada participante aporta recursos (almacenamiento y procesamiento) para acomodar el juego. Continuación del examen de soluciones peerto-pares se difiere hasta que los capítulos 2 y 10).

1.2.3 comercio financiero

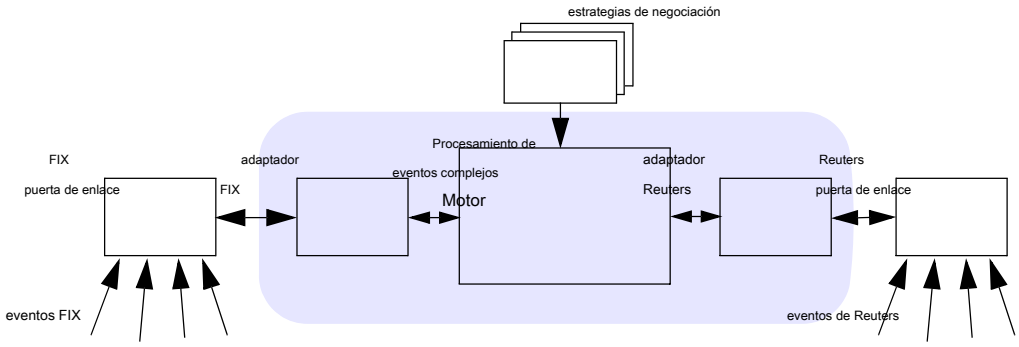
Como último ejemplo, nos fijamos en el apoyo de sistemas distribuidos para los mercados de comercio financiero. La industria financiera ha sido durante mucho tiempo a la vanguardia de la tecnología de sistemas distribuidos con su necesidad, en particular, para el acceso en tiempo real a una amplia gama de fuentes de información (por ejemplo, los precios y las tendencias actuales de las acciones, económico y acontecimientos políticos). La industria emplea automatizadas de control y las aplicaciones comerciales (véase a continuación).

Tenga en cuenta que el énfasis en este tipo de sistemas está en la comunicación y el procesamiento de artículos de interés, conocidos como *eventos* en sistemas distribuidos, con la necesidad también de ofrecer eventos de forma fiable y en tiempo y forma a potencialmente un gran número de clientes que tienen un interés establecida en estos elementos de información. Ejemplos de este tipo de eventos son una gota en un precio de la acción, la liberación de los últimos datos de desempleo, y así sucesivamente. Esto requiere un estilo muy diferente de la arquitectura subyacente de los estilos mencionados anteriormente (por ejemplo cliente-servidor), y tales sistemas emplean típicamente lo que se conoce como

sistemas basados en eventos distribuidos. Se presenta una ilustración de un uso típico de este tipo de sistemas de abajo y regrese a este importante tema con más profundidad en el capítulo 6.

La figura 1.2 ilustra un sistema típico de comercio financiero. Esto muestra una serie de eventos alimentaciones de entrada en una institución financiera dada. Dicho evento se alimenta compartir el

Figura 1.2 Un sistema de comercio financiero ejemplo



siguiente características. En primer lugar, las fuentes son típicamente en una variedad de formatos, tales como eventos de datos de mercado de Reuters y eventos FIX (eventos siguientes el formato específico del protocolo Financial Information eXchange), y de hecho desde diferentes tecnologías de eventos, ilustrando así el problema de la heterogeneidad como se encuentra en sistemas más distribuidos (ver también la Sección 1.5.1). La figura muestra el uso de adaptadores que se traducen formatos heterogéneos en un formato interno común. En segundo lugar, el sistema de comercio debe hacer frente a una variedad de flujos de eventos, todos los que llegan a un ritmo rápido, y con frecuencia requieren procesamiento en tiempo real para detectar patrones que indican las oportunidades comerciales. Esto solía ser un proceso manual, pero las presiones competitivas han llevado al aumento de la automatización en términos de lo que se conoce como complejo de procesamiento de eventos (CEP),

Este enfoque se utiliza principalmente para desarrollar estrategias de negociación algorítmica a medida que cubren tanto para la compra y venta de acciones y participaciones, en particular, en busca de patrones que indican una oportunidad de comercio y luego responden de forma automática mediante la colocación y gestión de pedidos. Como ejemplo, considere la siguiente secuencia de comandos:

CUANDO

MSFT precio se mueve fuera de un 2% de media móvil MSFT SEGUIDO

POR-(

MyBasket se mueve hacia arriba por el 0,5% y

El precio de HPQ se mueve hacia arriba en un

5% o

El precio de MSFT se mueve hacia abajo en un 2%

)

)

Todo dentro de

cualquier período de tiempo 2 minutos

ENTONCES

COMPRA VENTA

MSFT HPQ

Este guión se basa en la funcionalidad proporcionada por Apama [www.progress.com], Un producto comercial en el mundo financiero desarrollado originalmente de la investigación llevada a cabo en la Universidad de Cambridge. El script detecta una secuencia temporal compleja basada en los precios de las acciones de Microsoft, HP y una canasta de precios de las acciones, lo que resulta en las decisiones de compra o venta de acciones particulares.

Este estilo de la tecnología cada vez se utiliza en otras áreas de los sistemas financieros, incluyendo el seguimiento de la actividad comercial para gestionar el riesgo (en particular, el seguimiento de la exposición), para asegurar el cumplimiento con las regulaciones y para monitorear los patrones de actividad que podrían indicar las transacciones fraudulentas. En tales sistemas, los eventos se típicamente interceptados y pasaron a través de lo que es equivalente a un cumplimiento y firewall riesgo antes de ser procesado (véase también la discusión de los cortafuegos en la Sección 1.3.1 más adelante).

1.3 Tendencias en sistemas distribuidos

Los sistemas distribuidos están atravesando un período de cambios significativos y esto se remonta a una serie de tendencias influyentes:

- la aparición de la tecnología de redes omnipresente;
- la aparición de la computación ubicua junto con el deseo de apoyar la movilidad del usuario en sistemas distribuidos;
- la creciente demanda de servicios multimedia;
- la vista de los sistemas distribuidos como una utilidad.

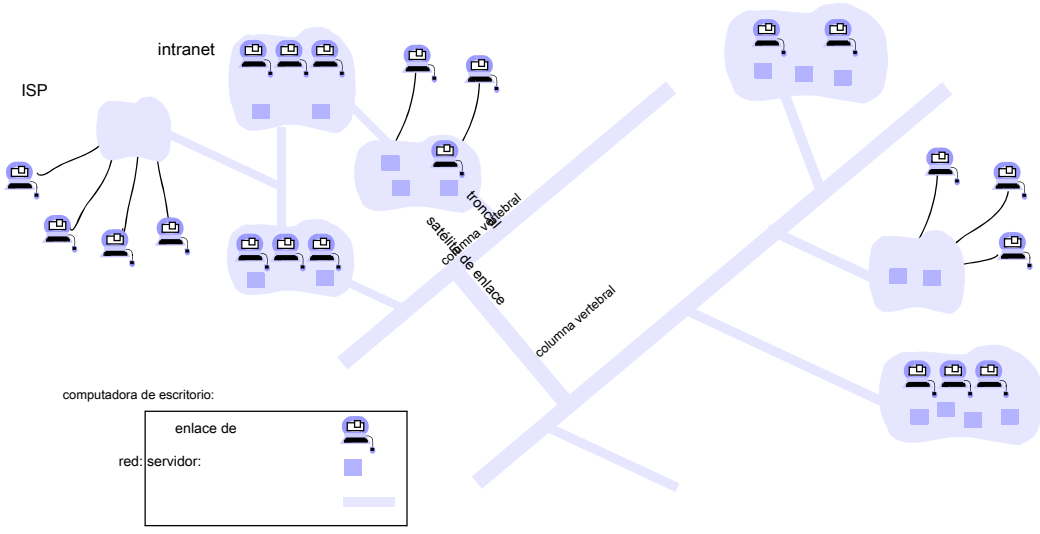
1.3.1 redes generalizado y la moderna Internet

La moderna Internet es una vasta colección interconectada de redes informáticas de muchos tipos diferentes, con la gama de tipos crecientes todo el tiempo y que ahora incluye, por ejemplo, una amplia gama de tecnologías de comunicación inalámbrica como WiFi, WiMAX, Bluetooth (véase el Capítulo 3) y las redes de telefonía móvil de tercera generación. El resultado neto es que la red se ha convertido en un recurso generalizado y dispositivos se pueden conectar (si se desea) en cualquier momento y en cualquier lugar.

Figura 1.3 ilustra una parte típica de la Internet. Programas que se ejecutan en los ordenadores conectados a él interactúan mediante el paso de mensajes, el empleo de un medio común de comunicación. El diseño y construcción de los mecanismos de comunicación de Internet (los protocolos de Internet) es un importante logro técnico, lo que permite un programa que se ejecuta en cualquier lugar para dirigir mensajes a los programas en cualquier otro lugar y resúmenes sobre la miríada de tecnologías mencionadas anteriormente.

La Internet es también un sistema distribuido muy grande. Permite a los usuarios, estén donde estén, para hacer uso de los servicios tales como el Wide Web, correo electrónico y transferencia de archivos Mundial. (De hecho, la Web es a veces incorrectamente equipara con la Internet.) El conjunto de servicios es abierto - se puede ampliar mediante la adición de equipos servidores y nuevos tipos de servicio. La figura muestra una colección de intranets - subredes operados por empresas y otras organizaciones y típicamente protegidos por cortafuegos. El papel de una *cortafuegos* es la protección de una intranet mediante la prevención de los mensajes no autorizados puedan salir o entrar. UN

Figura 1.3 Una porción típica de la Internet



servidor de seguridad se implementa mediante el filtrado de los mensajes entrantes y salientes. Los filtros pueden hacerse por origen o destino, o un firewall puede permitir que sólo aquellos mensajes relacionados con el acceso a Internet y correo electrónico a pasar dentro o fuera de la intranet que protege. Proveedores de Servicios de Internet (ISP) son compañías que proporcionan enlaces de banda ancha y otros tipos de conexión a los usuarios individuales y pequeñas organizaciones, lo que les permite acceder a los servicios en cualquier lugar de Internet, así como la prestación de servicios locales, tales como el correo electrónico y alojamiento web. Las intranets están unidos entre sí por cadenas principales. UN *columna vertebral* es un enlace de red con una alta capacidad de transmisión, empleando conexiones por satélite, cables de fibra óptica y otros circuitos de alto ancho de banda.

Tenga en cuenta que algunas organizaciones pueden no desear para conectar sus redes internas a Internet en absoluto. Por ejemplo, la policía y otras agencias de seguridad y de aplicación de la ley es probable que tengan al menos algunos intranets internas que están aislados del mundo exterior (el servidor de seguridad más eficaz posible - la ausencia de cualquier conexión física a Internet). Los cortafuegos también puede ser problemático en sistemas distribuidos al impedir el acceso legítimo a servicios cuando se requiere el intercambio de recursos entre los usuarios internos y externos. Por lo tanto, los cortafuegos a menudo deben complementarse con más mecanismos y políticas de grano fino, como se discutió en el Capítulo 11.

La aplicación de la Internet y los servicios que soporta ha conllevado el desarrollo de soluciones prácticas a los problemas del sistema distribuido muchos (incluyendo la mayoría de los que se definen en la Sección 1.5). Debemos destacar aquellas soluciones en todo el libro, señalando su alcance y sus limitaciones en su caso.

1.3.2 La computación móvil y ubicua

Los avances tecnológicos en la miniaturización de dispositivos y redes inalámbricas han llevado cada vez más a la integración de dispositivos informáticos pequeños y portátiles en los sistemas distribuidos. Estos dispositivos incluyen:

- Computadoras portatiles.
- Los dispositivos portátiles, incluyendo teléfonos móviles, teléfonos inteligentes, dispositivos habilitados para GPS, localizadores, asistentes digitales personales (PDA), cámaras de vídeo y cámaras digitales.
- dispositivos portátiles, tales como relojes inteligentes con funcionalidad similar a una PDA.
- Dispositivos integrados en electrodomésticos como lavadoras, equipos de música, automóviles y refrigeradores.

La portabilidad de muchos de estos dispositivos, junto con su capacidad de conectar convenientemente a las redes en diferentes lugares, hace *informática móvil* posible. La informática móvil es la realización de tareas de computación, mientras que el usuario está en movimiento, o lugares distintos de su entorno habitual visitar. En la informática móvil, los usuarios que se encuentran fuera de su intranet 'casa' (la intranet en el trabajo, o su residencia) están siendo provistos de acceso a los recursos a través de los dispositivos que llevan con ellos. Ellos pueden continuar teniendo acceso a Internet; pueden continuar teniendo acceso a los recursos en su intranet casa; y existe una creciente provisión para los usuarios utilizar los recursos tales como impresoras o incluso los puntos de venta que se encuentran convenientemente cerca mientras se mueven alrededor. Este último también se conoce como *consciente de la ubicación* o *consciente del contexto de computación*. Movilidad presenta una serie de desafíos para los sistemas distribuidos, incluyendo la necesidad de hacer frente a la conectividad variable y de hecho la desconexión, y la necesidad de mantener la operación en la cara de la movilidad del dispositivo (véase la discusión sobre la transparencia de movilidad en la Sección 1.5.7).

Computación ubicua es el aprovechamiento de los muchos dispositivos computacionales pequeños y baratos que están presentes en los entornos físicos de los usuarios, incluyendo el hogar, la oficina e incluso entornos naturales. El término 'omnipresente' se pretende sugerir que pequeños dispositivos informáticos con el tiempo llegar a ser tan omnipresente en los objetos cotidianos que apenas se note. Es decir, su comportamiento computacional estará ligada de forma transparente e íntimamente con su función física.

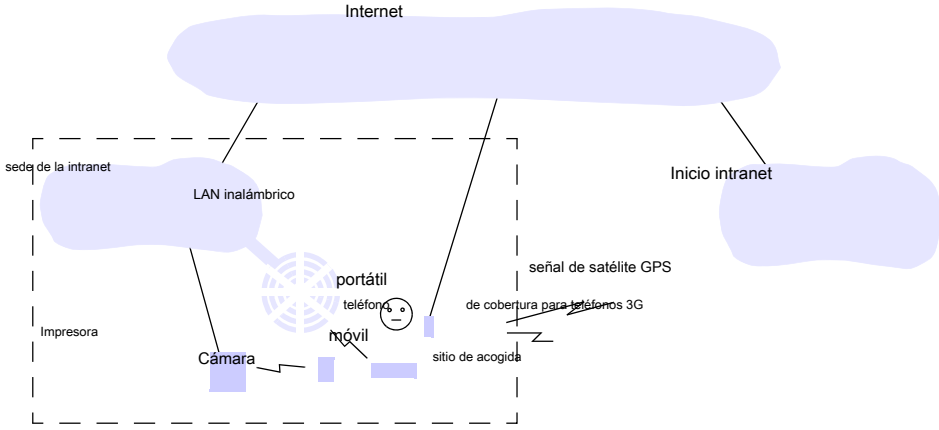
La presencia de los ordenadores de todo el mundo sólo se vuelve útil cuando se puedan comunicar entre sí. Por ejemplo, puede ser conveniente para los usuarios a controlar su lavadora o de su sistema de entretenimiento de su teléfono o un dispositivo de 'control remoto universal' en el hogar. Igualmente, la lavadora podría notificar al usuario a través de una tarjeta de identificación inteligente o teléfono cuando se realiza el lavado.

Ubicua y móvil solapamiento de cómputo, ya que el usuario móvil puede, en principio, se benefician de los equipos que están en todas partes. Pero ellos son distintos, en general. La computación ubicua podría beneficiar a los usuarios mientras permanecen en un solo ambiente, tales como el hogar o en un hospital. Del mismo modo, la informática móvil tiene ventajas incluso si se trata sólo discretos, computadoras y dispositivos convencionales tales como ordenadores portátiles e impresoras.

La Figura 1.4 muestra un usuario que está visitando una organización de acogida. La figura muestra la intranet de inicio del usuario y la intranet huésped en el sitio que el usuario está visitando. Ambos intranets están conectados al resto de Internet.

El usuario tiene acceso a tres formas de comunicación sin hilos. Su portátil tiene un medio de conexión a red LAN inalámbrica del huésped. Esta red proporciona cobertura para una

Figura 1.4 dispositivos portátiles y de mano en un sistema distribuido



unos cientos de metros (un piso de un edificio, por ejemplo). Se conecta al resto de la intranet host a través de un punto de puerta de enlace o de acceso. El usuario también tiene un teléfono móvil (celular), que está conectado a Internet. El teléfono da acceso a la web y otros servicios de Internet, limitado sólo por lo que se puede presentar en su pequeña pantalla, y también puede proporcionar información de ubicación a través de la funcionalidad GPS incorporado. Finalmente, el usuario lleva una cámara digital, que puede comunicarse a través de una red inalámbrica de área personal (con intervalo de hasta aproximadamente 10 m) con un dispositivo tal como una impresora.

Con una infraestructura de sistema adecuado, el usuario puede realizar algunas tareas simples en el sitio de acogida que utilizan los dispositivos que llevan. Mientras viajaba al sitio de acogida, el usuario puede obtener los últimos precios de las acciones de un servidor web usando el teléfono móvil y también puede utilizar las instrucciones integradas en el software GPS y la búsqueda de rutas para llegar a la ubicación del sitio. Durante la reunión con sus anfitriones, el usuario puede mostrarles una fotografía reciente de enviarlo desde la cámara digital directamente a una impresora adecuada habilitado (local) o proyector en la sala de reuniones (descubierto usando un servicio de localización). Esto requiere sólo el enlace inalámbrico entre la cámara y la impresora o proyector. Y pueden, en principio, enviar un documento desde su ordenador portátil a la misma impresora, utilizando los enlaces Ethernet LAN inalámbricas y por cable a la impresora.

Este escenario demuestra la necesidad de apoyar *interoperación espontánea,* por lo que las asociaciones entre los dispositivos se crean de forma rutinaria y destruidos - por ejemplo mediante la localización y el uso de dispositivos del huésped, tales como las impresoras. El principal reto de aplicar a este tipo de situaciones es hacer interoperación rápido y conveniente (es decir, espontánea) a pesar de que el usuario se encuentra en un entorno en el que nunca han visitado antes. Eso significa que permite el dispositivo del visitante a comunicarse en la red de acogida, y asociar el dispositivo con los servicios locales adecuados - un proceso llamado *descubrimiento de servicios.*

La informática móvil y ubicua representan animadas zonas de investigación, y las diversas dimensiones mencionadas anteriormente se tratan en profundidad en el capítulo 19.

1.3.3 sistemas multimedia distribuidos

Otra tendencia importante es el requisito para soportar servicios multimedia en sistemas distribuidos. apoyo Multimedia útil puede definirse como la capacidad de soportar una amplia gama de tipos de medios de manera integrada. Uno puede esperar que un sistema distribuido para apoyar el almacenamiento, transmisión y presentación de lo que se conoce como tipos de medios discretos, tales como imágenes o mensajes de texto a menudo. Un sistema multimedia distribuido debe ser capaz de realizar las mismas funciones para tipos de medios continuos tales como audio y vídeo; es decir, debe ser capaz de almacenar y localizar los archivos de audio o video, para transmitirlos a través de la red (posiblemente en tiempo real, como las corrientes emergen de una cámara de vídeo), para apoyar la presentación de los tipos de medios para el usuario y opcionalmente también para compartir los tipos de medios a través de un grupo de usuarios.

La característica fundamental de tipos de medios continuos es que incluyen una dimensión temporal, y de hecho, la integridad del tipo de medios depende fundamentalmente de la preservación de las relaciones en tiempo real entre los elementos de un tipo de medio. Por ejemplo, en una presentación de vídeo que es necesario para preservar un caudal dado en términos de cuadros por segundo y, para los flujos en tiempo real, un retardo máximo dado o la latencia para la entrega de tramas (esto es un ejemplo de calidad de servicio, discutido en más detalle en la Sección 1.5.8).

Los beneficios de la informática multimedia distribuido son considerables en la que una amplia gama de nuevos servicios y aplicaciones (multimedia) se puede proporcionar en el escritorio, incluyendo el acceso a vivir o emisiones de televisión pregrabados, el acceso a las bibliotecas de cine que ofrecen servicios de vídeo bajo demanda , acceso a bibliotecas de música, la provisión de conferencia de audio y vídeo instalaciones y características de telefonía integrados incluyendo la telefonía IP o tecnologías relacionadas, tales como Skype, una alternativa de igual a igual a la telefonía IP (la infraestructura del sistema distribuido que sustenta Skype se discute en la Sección 4.5 0.2). Tenga en cuenta que esta tecnología es revolucionaria en desafiar a los fabricantes a replantearse muchos dispositivos de consumo. Por ejemplo, ¿cuál es el dispositivo de la base entretenimiento en el hogar del futuro - la computadora, la televisión o la consola de juegos?

La difusión por Internet es una aplicación de la tecnología multimedia distribuido. La difusión por Internet es la capacidad para transmitir medios continuos, normalmente audio o vídeo, a través de Internet. Ahora es un lugar común para los grandes eventos deportivos o musicales que se emitirá de esta manera, a menudo atrayendo a un gran número de espectadores (por ejemplo, el concierto Live 8 en 2005 atrajo a alrededor de 170.000 usuarios simultáneos en su pico).

aplicaciones multimedia distribuidas como lugar de difusión por Internet considerables exigencias a la infraestructura distribuida subyacente en términos de:

- proporcionar soporte para una gama (extensible) de codificación y cifrado formatos, tales como la serie MPEG de normas (incluyendo por ejemplo la norma MP3 populares también conocido como MPEG-1, Capa de Audio 3) y HDTV;
- proporcionando una gama de mecanismos para asegurar que la calidad de servicio deseada se puede cumplir;
- proporcionar estrategias de gestión de los recursos asociados, incluidas las políticas de planificación apropiadas para apoyar la calidad de servicio deseada;
- proporcionar estrategias de adaptación para hacer frente a la situación inevitable en los sistemas abiertos, donde la calidad del servicio que no puede cumplir o sostenida. Además discusión de tales mecanismos se puede encontrar en el capítulo 20.

1.3.4 La computación distribuida como una utilidad

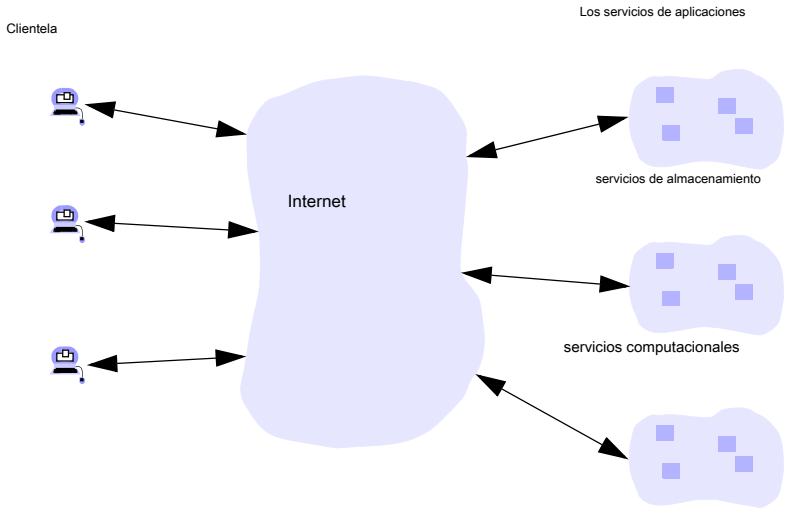
Con la creciente madurez de la infraestructura de sistemas distribuidos, un número de compañías están promoviendo la vista de los recursos distribuidos como una mercancía o utilidad, dibujo la analogía entre los recursos distribuidos y otros servicios públicos como agua o electricidad. Con este modelo, los recursos son proporcionados por los proveedores de servicios apropiados y eficaz alquilan en lugar de propiedad del usuario final. Este modelo se aplica tanto a los recursos materiales y servicios más lógicas:

- Los recursos físicos tales como el almacenamiento y el procesamiento pueden ponerse a disposición de los equipos en red, eliminando la necesidad de poseer dichos recursos por su cuenta. En un extremo del espectro, un usuario puede optar por una instalación de almacenamiento remoto para las necesidades de almacenamiento de archivos (por ejemplo, para datos multimedia como fotografías, música o vídeo) y / o para copias de seguridad. Del mismo modo, este enfoque permitiría a un usuario para alquilar uno o más nodos de cómputo, ya sea para satisfacer sus necesidades básicas de computación o de hecho para realizar computación distribuida. En el otro extremo del espectro, los usuarios pueden acceder sofisticada *centros de datos (* instalaciones en red que ofrecen acceso a repositorios de frecuencia grandes volúmenes de datos a los usuarios u organizaciones) o de hecho la infraestructura computacional utilizando el tipo de servicios que ha de prestar compañías como Amazon y Google. Operativo de virtualización del sistema es una tecnología clave para este enfoque, lo que implica que los usuarios realmente pueden estar provistos de servicios por parte de un Portal en lugar de un nodo físico. Esto ofrece una mayor flexibilidad al proveedor de servicios en materia de gestión de recursos (que opera la virtualización del sistema se analiza con más detalle en el capítulo 7).

- Los servicios de software (tal como se define en la Sección 1.4) también pueden estar disponibles a través de Internet global utilizando este enfoque. De hecho, muchas compañías ahora ofrecen una amplia gama de servicios de alquiler efectiva, incluyendo servicios como el correo electrónico y calendarios distribuidos. Google, por ejemplo, haces una gama de servicios a las empresas bajo la bandera de Google Apps [me www.google.com]. Este desarrollo está habilitada para los estándares acordados para los servicios de software, por ejemplo, según lo dispuesto por los servicios web (véase el capítulo 9). El termino *computación en la nube* se utiliza para capturar esta visión de la informática como una utilidad. Una nube se define como un conjunto de basado en Internet servicios de computación suficientes para apoyar las necesidades la mayoría de los usuarios de aplicaciones, el almacenamiento y, lo que les permite prescindir en gran parte o totalmente con el almacenamiento de datos local y software de aplicación (véase la Figura 1.5). El término también promueve una visión de todo como un servicio, desde la infraestructura física o virtual a través de software, a menudo pagado en una base por-uso en lugar de comprar. Tenga en cuenta que la computación en nube reduce los requisitos en los dispositivos de los usuarios, permitiendo de escritorio muy simple o dispositivos portátiles para acceder a una potencialmente amplia gama de recursos y servicios.

Nubes generalmente se implementan en los equipos del clúster para proporcionar la escala y el rendimiento necesarios requeridos por dichos servicios. UN *equipo de clúster* es un conjunto de ordenadores interconectados que cooperan estrechamente para proporcionar una única capacidad de alto rendimiento de computación e integrada. Sobre la base de proyectos como la EMPRESA (Red de Estaciones de Trabajo) Proyecto en Berkeley [Anderson *et al.* 1995, now.cs.berkeley.edu] Y Beowulf en la NASA [www.beowulf.org], La tendencia es hacia la utilización de hardware común tanto para los ordenadores y de las redes de interconexión. La mayoría de los racimos

Figura 1.5 Computación en la nube



consistir en PCs de las materias primas que ejecuta una versión de un sistema operativo como Linux estándar (a veces de corte hacia abajo), interconectados por una red de área local. Empresas como HP, Sun e IBM ofrecen soluciones blade. *Los servidores blade* son elementos computacionales mínimos que contienen, por ejemplo capacidades de procesamiento y de almacenamiento (memoria principal). Un sistema de la lámina se compone de un número potencialmente grande de servidores blade contenidos dentro de un recinto de cuchilla. Otros elementos tales como la energía, enfriamiento, almacenamiento persistente (discos), redes y muestra, se proporcionan ya sea por el recinto o a través de soluciones virtualizados (discutido en el capítulo 7). A través de esta solución, los servidores blade individuales pueden ser mucho más pequeños y también más barato de producir que los PC de los productos básicos.

El objetivo general de los equipos del clúster es proporcionar una gama de servicios en la nube, incluyendo capacidades de computación de alto rendimiento, almacenamiento masivo (por ejemplo a través de los centros de datos), y los servicios de aplicaciones más ricos, como búsqueda en la web (Google, por ejemplo, se basa en un cúmulo masivo arquitectura informática para implementar su motor de búsqueda y otros servicios, como se discutió en el capítulo 21).

La computación grid (como se discute en el capítulo 9, sección 9.7.2) también se puede ver como una forma de la nube. Los términos son en gran parte sinónimo y, a veces definen malos, pero computación Grid generalmente pueden ser vistos como un precursor para el paradigma más general de la nube con un sesgo hacia el soporte para aplicaciones científicas.

1.4 Enfoque en el intercambio de recursos

Los usuarios están tan acostumbrados a los beneficios de la distribución de los recursos que se pueden pasar por alto fácilmente su significado. Compartimos rutinariamente los recursos de hardware, tales como impresoras, recursos de datos, tales como archivos y recursos con funcionalidad más específicos, como los motores de búsqueda.

Mirado desde el punto de vista de la provisión de hardware, compartimos equipos tales como impresoras y discos para reducir los costos. Pero de mucha mayor importancia para los usuarios es la puesta en común de los recursos de más alto nivel que juegan un papel en sus aplicaciones y en su trabajo diario y las actividades sociales. Por ejemplo, los usuarios tienen que ver con el intercambio de datos en la forma de una base de datos compartida o un conjunto de páginas web - no los discos y procesadores en los que se implementan. Del mismo modo, los usuarios piensan en términos de recursos compartidos, como un motor de búsqueda o un convertidor de moneda, sin tener en cuenta para el servidor o servidores que proporcionan estos.

En la práctica, los patrones de distribución de los recursos varían ampliamente en su alcance y en qué punto los usuarios trabajar juntos. En un extremo, un motor de búsqueda en la Web proporciona una facilidad para los usuarios en todo el mundo, los usuarios que no necesitan entrar en contacto entre sí directamente. En el otro extremo, en *trabajo cooperativo asistido por ordenador* (CSCW), un grupo de usuarios que colaboran directamente comparten recursos tales como documentos en un grupo pequeño, cerrado. El patrón de intercambio y la distribución geográfica de los usuarios particulares determina cuáles son los mecanismos que el sistema debe suministrar para coordinar las acciones de los usuarios.

Usamos el término *Servicio* por una parte distinta de un sistema informático que gestiona una colección de recursos relacionados y presenta su funcionalidad a los usuarios y aplicaciones. Por ejemplo, tenemos acceso a los archivos compartidos a través de un servicio de archivos; enviamos documentos a las impresoras a través de un servicio de impresión; compramos bienes a través de un servicio de pago electrónico. El único acceso que tenemos al servicio se realiza a través del conjunto de operaciones que se exporta. Por ejemplo, un servicio de archivo proporciona *leer escribir* y *borrar* operaciones en los ficheros.

El hecho de que los servicios de restringir el acceso a los recursos a un conjunto bien definido de operaciones se encuentra en práctica de la ingeniería de software estándar de la Parte. Pero también refleja la organización física de los sistemas distribuidos. Recursos en un sistema distribuido se encapsulan físicamente dentro de los ordenadores y sólo se puede acceder desde otros ordenadores por medio de la comunicación. Para el intercambio efectivo, cada recurso debe ser gestionado por un programa que ofrece una interfaz de comunicación que permite el recurso a acceder y actualizar de forma fiable y consistente.

El termino *servidor* es probablemente familiar a la mayoría de los lectores. Se refiere a un programa en ejecución (una *proceso)* en un ordenador en red que acepta solicitudes de los programas que se ejecutan en otros equipos para realizar un servicio y responde apropiadamente. Los procesos que solicitan se denominan *clientela,* y el enfoque general es conocido como *computación cliente-servidor.* En este enfoque, se envían solicitudes de mensajes de clientes a un servidor y las respuestas se envían mensajes desde el servidor a los clientes. Cuando el cliente envía una solicitud de una operación que se lleva a cabo, decimos que el cliente *invoca una operación*

en el servidor. Una completa interacción entre un cliente y un servidor, desde el momento en que el cliente envía su petición a cuando recibe la respuesta del servidor, se le llama *invocación remota.*

El mismo proceso puede ser un cliente y un servidor, ya que los servidores a veces invocan operaciones en otros servidores. Los términos 'cliente' y 'servidor' se aplican únicamente a las funciones que desempeñan en una sola solicitud. Los clientes están activos (hacer peticiones) y los servidores son pasivos (sólo despertarse cuando reciben solicitudes); servidores funcionan de forma continua, mientras que los clientes sólo duran el tiempo que las aplicaciones de la que forman parte.

Tenga en cuenta que, si bien de forma predeterminada los términos 'cliente' y 'servidor' se refieren a *procesos* en lugar de los ordenadores que ejecutan sobre, en el lenguaje cotidiano esos términos también se refieren a los propios ordenadores. Otra distinción, que discutiremos en el capítulo 5,

es que en un sistema distribuido por escrito en un lenguaje orientado a objetos, los recursos pueden ser encapsulados como objetos y acceder a los objetos de cliente, en cuyo caso hablamos de una *objeto de cliente* invocar un método sobre una *objeto de servidor*.

Muchos, pero ciertamente no todos, los sistemas distribuidos pueden ser construidos en su totalidad en la forma de interactuar clientes y servidores. Los Wide Web, correo electrónico y en red impresoras mundo todos se ajustan a este modelo. Se discuten alternativas a los sistemas cliente-servidor en el Capítulo 2.

Un navegador web de ejecución es un ejemplo de un cliente. El navegador web se comunica con un servidor web, para solicitar páginas web de la misma. Consideramos que la web y su arquitectura cliente-servidor asociado con más detalle en la sección 1.6.

1.5 Desafíos

Los ejemplos en la Sección 1.2 están destinados a ilustrar el alcance de los sistemas distribuidos y sugerir los problemas que surgen en su diseño. En muchos de ellos, se encontraron y superar desafíos significativos. A medida que se amplía el alcance y la escala de los sistemas y aplicaciones distribuidas las mismas y otros desafíos son susceptibles de ser encontrado. En esta sección se describen los principales retos.

1.5.1 heterogeneidad

Internet permite a los usuarios acceder a los servicios y aplicaciones se ejecutan a través de una colección heterogénea de computadoras y redes. La heterogeneidad (es decir, la variedad y diferencia) se aplica a todos los siguientes:

- redes;
- hardware de la computadora;
- sistemas operativos;
- lenguajes de programación;
- implementaciones por diferentes desarrolladores.

Aunque la Internet se compone de muchos tipos diferentes de red (ilustrado en la figura 1.3), sus diferencias son enmascarados por el hecho de que todos los ordenadores conectados a ellos utilizan los protocolos de Internet para comunicarse entre sí. Por ejemplo, un ordenador conectado a una red Ethernet tiene una implementación de los protocolos de Internet a través de Ethernet, mientras que un ordenador en un tipo diferente de red necesitará una implementación de los protocolos de Internet para esa red. El capítulo 3 explica cómo se implementan los protocolos de Internet a través de una variedad de diferentes redes.

Los tipos de datos tales como números enteros pueden ser representados de diferentes maneras en diferentes tipos de hardware - por ejemplo, hay dos alternativas para la ordenación de bytes de números enteros. Estas diferencias en la representación debe ser tratado si los mensajes han de ser intercambiados entre los programas que se ejecutan en un hardware diferente.

Aunque los sistemas operativos de todos los ordenadores en Internet tienen que incluir una implementación de los protocolos de Internet, que no necesariamente todos ofrecen la misma interfaz de programación de aplicaciones para estos protocolos. Por ejemplo, las llamadas para intercambiar mensajes en UNIX son diferentes de las llamadas de Windows.

Diferentes lenguajes de programación utilizan diferentes representaciones de personajes y estructuras de datos tales como matrices y registros. Estas diferencias se deben abordar si los programas escritos en diferentes idiomas han de ser capaces de comunicarse entre sí.

Los programas escritos por diferentes desarrolladores no pueden comunicarse entre sí a menos que usen estándares comunes, por ejemplo, para la comunicación de red y la representación de los elementos de datos primitivos y estructuras de datos en los mensajes. Para que esto suceda, las normas deben ser aceptado y adoptado - al igual que los protocolos de Internet.

• middleware El termino *middleware* se aplica a una capa de software que proporciona una abstracción de programación, así como enmascara la heterogeneidad de las redes subyacentes, hardware, sistemas operativos y lenguajes de programación. El Common Object Request Broker (CORBA), que se describe en los capítulos 4, 5 y 8, es un ejemplo. Algunos middleware, como Java Remote Method Invocation (RMI) (véase el capítulo 5), sólo admite un único lenguaje de programación. La mayoría de middleware se implementa a través de los protocolos de Internet, que a su máscara de las diferencias de las redes subyacentes, pero todas las ofertas de middleware con las diferencias en los sistemas operativos y hardware - cómo se hace esto es el tema principal del capítulo 4.

Además de resolver los problemas de la heterogeneidad, middleware proporciona un modelo computacional uniforme para su uso por los programadores de servidores y las aplicaciones distribuidas. modelos posibles incluyen la invocación remota de objetos, notificación de eventos remoto, acceso remoto de SQL y procesamiento de transacciones distribuidas. Por ejemplo, CORBA proporciona invocación objeto remoto, que permite que un objeto en un programa que se ejecuta en un ordenador para invocar un método de un objeto en un programa que se ejecuta en otro equipo. Su implementación oculta el hecho de que los mensajes se pasan a través de una red con el fin de enviar la solicitud de invocación y su respuesta.

Heterogeneidad y código móvil • El termino *código móvil* se utiliza para referirse al código de programa que pueden ser transferidos desde un ordenador a otro y correr en el destino - applets de Java son un ejemplo. Código adecuado para correr en un ordenador no es necesariamente adecuado para correr en otra, porque los programas ejecutables son normalmente específicos tanto para el conjunto de instrucciones y para el sistema operativo anfitrión.

los *máquina virtual* enfoque proporciona una manera de hacer que el código ejecutable en una variedad de equipos host: el compilador para un lenguaje particular, genera código para una máquina virtual en lugar de un código de pedido de hardware en particular. Por ejemplo, el compilador de Java genera código para una máquina virtual Java, que lo ejecuta por la interpretación. La máquina virtual de Java necesita ser implementado una vez para cada tipo de equipo para permitir que los programas Java para funcionar.

Hoy en día, la forma más común de código móvil es los programas de inclusión de JavaScript en algunas páginas web se cargan en los navegadores cliente. Esta extensión de la tecnología Web se discute más adelante en la Sección 1.6.

1.5.2 apertura

La apertura de un sistema de ordenador es la característica que determina si el sistema se puede extender y reimplementada de varias maneras. La apertura de los sistemas distribuidos se determina principalmente por el grado en que los nuevos servicios de intercambio de recursos se pueden agregar y estar disponible para su uso por una variedad de programas de cliente.

La apertura no puede lograrse a menos que la especificación y la documentación de las interfaces de software clave de los componentes de un sistema están a disposición de los desarrolladores de software. En una palabra, las interfaces son clave *publicado*. Este proceso es similar a la estandarización de las interfaces, pero a menudo se pasa por alto los procedimientos de normalización oficiales, que suelen ser engorrosos y lentos.

Sin embargo, la publicación de interfaces es sólo el punto de partida para la adición y extensión de los servicios en un sistema distribuido. El reto para los diseñadores es hacer frente a la complejidad de los sistemas distribuidos que consisten en muchos componentes de ingeniería por diferentes personas.

Los diseñadores de los protocolos de Internet introdujeron una serie de documentos llamados " solicitudes de comentarios, o RFC, cada uno de los cuales es conocido por un número. Las especificaciones de los protocolos de comunicación de Internet fueron publicados en esta serie a principios de 1980, seguido de especificaciones para las aplicaciones que se ejecutan sobre ellos, tales como transferencia de archivos, correo electrónico y telnet a mediados de la década de 1980. Esta práctica ha continuado y constituye la base de la documentación técnica de Internet. Esta serie incluye debates, así como las especificaciones de los protocolos. Las copias se pueden obtener a partir de [www.ietf.org]. Así, la publicación de los protocolos de comunicación de Internet originales ha permitido una gran variedad de sistemas y aplicaciones de Internet, incluyendo la Web para ser construido. RFCs no son los únicos medios de publicación. Por ejemplo, el World Wide Web Consortium (W3C) desarrolla y publica estándares relacionados con el funcionamiento de la Web [www.w3.org].

Sistemas que están diseñados para apoyar el intercambio de recursos de esta manera se denominan *abrir los sistemas distribuidos* hacer hincapié en el hecho de que son extensibles. Ellos pueden ser extendidas a nivel de hardware mediante la adición de los ordenadores a la red y en el nivel de software mediante la introducción de nuevos servicios y la reimplementación de los antiguos, que permite programas de aplicación para compartir recursos. Un beneficio adicional que se cita a menudo para sistemas abiertos es su independencia de los vendedores individuales.

Para resumir:

- Los sistemas abiertos se caracterizan por el hecho de que sus interfaces clave se publican.
- sistemas distribuidos abiertos se basan en la provisión de un mecanismo de comunicación uniforme y las interfaces para el acceso a los recursos compartidos publicados.
- sistemas distribuidos abiertos se pueden construir de hardware y software heterogéneo, posiblemente de diferentes proveedores. Sin embargo, la conformidad de cada componente de la norma publicada debe ser cuidadosamente probado y verificado si el sistema está funcionando correctamente.

1.5.3 Seguridad

Muchos de los recursos de información que están disponibles y se mantienen en sistemas distribuidos tienen un alto valor intrínseco a sus usuarios. por lo tanto, de considerable importancia es su seguridad. Seguridad de los recursos de información tiene tres componentes: la confidencialidad (protección contra la divulgación a personas no autorizadas), integridad (protección contra la alteración o corrupción) y la disponibilidad (protección contra la interferencia con los medios para acceder a los recursos).

Sección 1.1 señaló que aunque Internet permite que un programa en una computadora para comunicarse con un programa en otro ordenador, independientemente de su

ubicación, los riesgos de seguridad están asociados con permitir el libre acceso a todos los recursos en una intranet. Aunque un servidor de seguridad se puede utilizar para formar una barrera alrededor de una intranet, lo que restringe el tráfico que puede entrar y salir, esto no se ocupa de garantizar el uso adecuado de los recursos por los usuarios dentro de una intranet, o con el uso adecuado de los recursos en Internet , que no están protegidos por cortafuegos.

En un sistema distribuido, los clientes envían peticiones a los datos de acceso gestionados por los servidores, lo que implica el envío de información de mensajes a través de una red. Por ejemplo:

1. Un médico puede solicitar el acceso a los datos del paciente del hospital o envíe adiciones a esos datos.

2. En el comercio electrónico y la banca, los usuarios envían sus números de tarjetas de crédito a través de Internet.

En ambos ejemplos, el reto consiste en enviar información sensible en un mensaje a través de una red de una manera segura. Sin embargo, la seguridad no es sólo una cuestión de ocultar el contenido de los mensajes - es también conocer con certeza la identidad del usuario u otro agente en cuyo nombre se envió un mensaje. En el primer ejemplo, el servidor tiene que saber que el usuario es realmente un médico, y en el segundo ejemplo, el usuario tiene que estar seguro de la identidad de la tienda o banco con el que están tratando. El segundo reto consiste en identificar a un usuario remoto u otro agente correctamente. Ambos de estos desafíos pueden ser satisfechas por el uso de técnicas de cifrado desarrollados para este propósito. Se utilizan ampliamente en Internet y se discuten en el Capítulo 11.

Sin embargo, los siguientes dos desafíos de seguridad aún no se han cumplido en su totalidad:

Ataques de denegación de servicio: Otro problema de seguridad es que un usuario puede desear interrumpir un servicio por alguna razón. Esto se puede lograr mediante el bombardeo de servicio con un gran número de peticiones sin sentido, que los usuarios serios no son capaces de utilizarlo como. Esto se llama una *negación de servicio* ataque. Ha habido varios ataques de denegación de servicio en los servicios web conocidos. Actualmente este tipo de ataques se ven contrarrestados por el intento de atrapar y castigar a los autores después del evento, pero que no es una solución general al problema. Contramedidas basadas en mejoras en la gestión de las redes están en desarrollo, y estos serán tocados en el Capítulo 3.

Seguridad del código móvil: código móvil tiene que ser manejado con cuidado. Considere una persona que recibe un programa ejecutable como un archivo adjunto de correo electrónico: los posibles efectos de la ejecución del programa son impredecibles; por ejemplo, puede parecer para mostrar un cuadro interesante pero en realidad se puede acceder a los recursos locales, o tal vez ser parte de un ataque de denegación de servicio. Algunas medidas para asegurar el código móvil se describen en el capítulo 11.

1.5.4 Escalabilidad

Los sistemas distribuidos operan con eficacia y eficiencia en muchas escalas diferentes, que van desde una pequeña intranet a Internet. Un sistema se describe como *escalable* si seguirá siendo eficaz cuando hay un aumento significativo en el número de recursos y el número de usuarios. El número de ordenadores y servidores en Internet ha aumentado de forma espectacular. La figura 1.6 muestra el número creciente de ordenadores y servidores web durante los 12 años de historia de la Web hasta el año 2005 [zakon.org]. Es interesante observar el crecimiento significativo tanto en ordenadores y servidores web en este período, pero también que la

porcentaje relativo se aplana a cabo - una tendencia que se explica por el crecimiento de la computación personal fijo y móvil. Un servidor web puede también ser cada vez alojado en varios equipos.

El diseño de sistemas distribuidos escalables presenta los siguientes desafíos:

Controlar el coste de los recursos físicos: A medida que la demanda de un recurso crece, debería ser posible extender el sistema, a un costo razonable, para cumplir con ella. Por ejemplo, la frecuencia con la que se accede a los archivos en una intranet es probable que aumente el número de usuarios y equipos aumenta. Debe ser posible añadir equipos de servidor para evitar el cuello de botella que se produciría si un solo servidor de archivos tenía que manejar todas las peticiones de acceso a archivos. En general, para un sistema con *norte* los usuarios ser escalable, la cantidad de recursos físicos necesarios para apoyarlos deben ser como máximo O (*norte)* - es decir, proporcional a *norte.* Por ejemplo, si un solo servidor de archivos puede soportar 20 usuarios, a continuación, dos de estos servidores deben ser capaces de soportar 40 usuarios. A pesar de que suena un objetivo obvio, no es necesariamente fácil de lograr en la práctica, como veremos en el capítulo 12.

El control de la pérdida de rendimiento: Considere la gestión de un conjunto de datos cuyo tamaño es proporcional al número de usuarios o recursos en el sistema - por ejemplo, la mesa con la correspondencia entre los nombres de dominio de los ordenadores y sus direcciones de Internet mantenidos por el sistema de nombres de dominio, que se utiliza principalmente para buscar nombres DNS como www.amazon.com. Algoritmos que utilizan estructuras jerárquicas escalan mejor que las que utilizan estructuras lineales. Pero incluso con estructuras jerárquicas un aumento de tamaño se traducirá en una pérdida en el rendimiento: el tiempo necesario para acceder a los datos de estructura jerárquica es O (*log n),* dónde *norte* es el tamaño del conjunto de datos. Para que un sistema sea escalable, la máxima pérdida de rendimiento no debería ser peor que esto.

La prevención de los recursos de software en ejecución a cabo: Un ejemplo de falta de escalabilidad se muestra por los números utilizados como direcciones de Internet (IP) (direcciones de ordenador en Internet). A finales de 1970, se decidió utilizar 32 bits para este propósito, pero como se explicará en el capítulo 3, el suministro de direcciones de Internet disponibles se está acabando. Por esta razón, se ha adoptado una nueva versión del protocolo de Internet con direcciones de 128 bits, lo que requerirá modificaciones a muchos componentes de software. Para ser justo

Figura 1.6 Crecimiento de Internet (ordenadores y servidores web)

Fecha	*Ordenadores*	*Los servidores web*	*Porcentaje*
1993, julio de	1776000	130	0,008
1995, julio de	6642000	23.500	0.4
1997, julio de	19540000	1203096	6
1999, julio de	56218000	6598697	12
2001, julio de	125888197	31299592	25
2003, julio de	~ 200 millones	42298371	21
2005, julio de	353284187	67571581	19

a principios de los diseñadores de Internet, no existe una solución correcta a este problema. Es difícil predecir la demanda que será puesto en un sistema próximos años. Por otra parte, el exceso de compensación para el crecimiento futuro puede ser peor que la adaptación a un cambio cuando nos vemos obligados a - las direcciones de Internet más grandes van a ocupar espacio adicional en los mensajes y en la memoria interna.

Evitar los cuellos de botella de rendimiento: En general, los algoritmos deben descentralizarse para evitar que los cuellos de botella de rendimiento. Se ilustra este punto con referencia al predecesor del sistema de nombres de dominio, en el que la tabla de nombres se mantuvo en un solo archivo maestro que pudiera ser descargado en cualquier equipo que lo necesitaban. Eso estaba bien cuando sólo había unos pocos cientos de ordenadores en Internet, pero pronto se convirtió en un cuello de botella de rendimiento grave y administrativa. El sistema de nombres de dominio eliminado este cuello de botella mediante la partición de la tabla de nombres entre servidores ubicados en todo el Internet y administrados localmente - véanse los capítulos 3 y 13.

Algunos recursos compartidos se accede con mucha frecuencia; por ejemplo, muchos usuarios pueden acceder a la misma página web, causando una disminución en el rendimiento. Nos veremos en el capítulo 2 que el almacenamiento en caché y replicación pueden utilizarse para mejorar el rendimiento de los recursos que se utilizan muy fuertemente.

Idealmente, el sistema y el software de aplicación no tiene por qué cambiar cuando la escala del sistema aumenta, pero esto es difícil de lograr. La cuestión de la escala es un tema dominante en el desarrollo de sistemas distribuidos. Las técnicas que han tenido éxito son discutidos ampliamente en este libro. Ellos incluyen el uso de datos replicados (Capítulo 18), la técnica asociada de almacenamiento en caché (capítulos 2 y 12) y el despliegue de múltiples servidores para gestionar las tareas común, lo que permite varias tareas similares a ser realizado simultáneamente.

1.5.5 La falta de manipulación

Los sistemas informáticos a veces fallan. Cuando ocurre un fallo en el hardware o el software, los programas pueden producir resultados incorrectos o pueden detenerse antes de que hayan completado el cálculo previsto. Discutiremos y clasificar una serie de posibles tipos de errores que pueden ocurrir en los procesos y las redes que componen un sistema distribuido en el Capítulo 2.

Los fallos en un sistema distribuido son parciales - es decir, algunos componentes fallan mientras que otros siguen funcionando. Por lo tanto la manipulación de los fallos es particularmente difícil. Las siguientes técnicas para hacer frente a los fallos se discuten en el libro:

La detección de fallos: Algunos fallos pueden ser detectados. Por ejemplo, las sumas de comprobación se pueden utilizar para detectar datos corruptos en un mensaje o un archivo. Capítulo 2 explica que es difícil o incluso imposible de detectar algunos otros fallos, como por ejemplo un servidor caído a distancia en Internet. El desafío consiste en gestionar en presencia de fallos que no pueden ser detectados, pero se puede sospechar.

Enmascaramiento de fallos: Algunos fallos detectados se pueden ocultar o hechos menos graves. Dos ejemplos de fallos de ocultación:

1. Los mensajes pueden ser retransmitidos cuando no pueden llegar.

2. Los datos del archivo se pueden escribir en un par de discos de manera que si uno está dañado, el otro todavía puede ser correcta.

Simplemente dejar caer un mensaje que está dañado es un ejemplo de lo que un fallo menos grave

- podría ser retransmitido. El lector probablemente se dará cuenta de que las técnicas descritas por las fallas de ocultación no están garantizados para trabajar en el peor de los casos; por ejemplo, los datos sobre el segundo disco se pueden dañar también, o el mensaje no puede conseguir a través de en un tiempo razonable sin embargo a menudo se retransmite.

Tolerar fallos: La mayor parte de los servicios en el Internet ¿exhibir fracasos - que no sería práctico para ellos para tratar de detectar y ocultar todos los fallos que pudieran producirse en una red tan grande con tantos componentes. Sus clientes pueden ser diseñados para tolerar fallos, lo que implica por lo general los usuarios tolerar ellos también. Por ejemplo, cuando un navegador web no puede ponerse en contacto con un servidor web, que no hace que el usuario espere para siempre, mientras que persiste en el intento - que informa al usuario sobre el problema, dejándolos libres para volver a intentarlo más tarde. Servicios que toleran fallos se discuten en el párrafo sobre la redundancia a continuación.

La recuperación de fallos: La recuperación implica el diseño de software para que el estado de los datos permanentes se puede recuperar o 'deshace' después de un servidor se ha estrellado. En general, los cálculos realizados por algunos programas estarán incompletas cuando se produce un fallo, y los datos permanentes que actualicen (archivos y otros materiales almacenados en el almacenamiento permanente) puede no estar en un estado coherente. La recuperación se describe en el Capítulo 17.

Redundancia: Los servicios pueden ser hechas para tolerar fallos mediante el uso de componentes redundantes. Considere los siguientes ejemplos:

1. Siempre debe haber al menos dos rutas diferentes entre dos routers en Internet.

2. En el sistema de nombres de dominio, nombre de cada mesa se replica en al menos dos servidores diferentes.

3. Una base de datos puede ser replicado en varios servidores para asegurar que los datos sigue siendo accesible tras el fracaso de cualquier servidor único; los servidores pueden ser diseñados para detectar fallos en sus compañeros; cuando se detecta un fallo en un servidor, los clientes son redirigidos a los servidores restantes.

El diseño de técnicas eficaces para mantener réplicas de los datos que cambian rápidamente hasta la fecha sin excesiva pérdida de rendimiento es un reto. Enfoques se discuten en el Capítulo 18.

Los sistemas distribuidos proporcionan un alto grado de disponibilidad en la cara de los fallos de hardware. los *disponibilidad* de un sistema es una medida de la proporción de tiempo que está disponible para su uso. Cuando uno de los componentes en un sistema distribuido falla, sólo el trabajo que estaba utilizando el componente que ha fallado se ve afectada. Un usuario puede mover a otro equipo si el que estaban usando falla; un proceso de servidor se puede iniciar en otro equipo.

1.5.6 concurrencia

Ambos servicios y aplicaciones proporcionan los recursos que pueden ser compartidos por los clientes en un sistema distribuido. Por lo tanto, existe la posibilidad de que varios clientes intentarán

acceder a un recurso compartido al mismo tiempo. Por ejemplo, una estructura de datos que registra las ofertas para una subasta se puede acceder con mucha frecuencia cuando se pone cerca de la hora límite.

El proceso que administra un recurso compartido podría tomar una solicitud de un cliente a la vez. Pero ese enfoque limita el rendimiento. Por lo tanto, los servicios y aplicaciones en general, permiten que múltiples solicitudes de clientes para ser procesados simultáneamente. Para hacer esto más concreto, supongamos que cada recurso se encapsula como un objeto y que invocaciones se ejecutan en hilos concurrentes. En este caso es posible que varios hilos puede estar ejecutando simultáneamente dentro de un objeto, en cuyo caso sus operaciones en el objeto pueden entrar en conflicto entre sí y producir resultados inconsistentes. Por ejemplo, si dos ofertas concurrentes en una subasta son 'Smith: $ 122' y 'Jones: $ 111', y las operaciones correspondientes son intercalados sin ningún control, entonces podrían quedar almacenado como 'Smith: $ 111' y 'Jones: $ 122' .

La moraleja de esta historia es que cualquier objeto que representa un recurso compartido en un sistema distribuido debe ser responsable de asegurar un funcionamiento óptimo en un entorno concurrente. Esto se aplica no sólo a los servidores, sino también a los objetos de aplicaciones. Por tanto, cualquier programador que toma una implementación de un objeto que no fue diseñado para ser utilizado en un sistema distribuido debe hacer todo lo necesario para que sea seguro en un entorno concurrente.

Para que un objeto sea seguro en un entorno concurrente, sus operaciones deben estar sincronizados de tal manera que sus datos se mantiene constante. Esto se puede lograr mediante técnicas estándar tales como semáforos, que se utilizan en la mayoría de sistemas operativos. En este tema y su extensión a las colecciones de objetos compartidos distribuidos se analizan en los capítulos 7 y 17.

1.5.7 Transparencia

La transparencia se define como el ocultamiento del usuario y el programador de la aplicación de la separación de componentes en un sistema distribuido, de modo que el sistema se percibe como un todo y no como una colección de componentes independientes. Las implicaciones de la transparencia son una gran influencia en el diseño del software del sistema.

El Manual de Referencia ANSA [ANSA 1989] y la Organización Internacional para la Normalización del modelo de referencia para el procesamiento distribuido abierto (RM-ODP) [ISO 1992] identifican ocho formas de transparencia. Hemos parafraseado las definiciones originales ANSA, en sustitución de su transparencia migración con nuestra propia transparencia movilidad, cuyo alcance es más amplio:

la transparencia de acceso permite que los recursos locales y remotos para ser accedidas usando operaciones idénticas.

transparencia de ubicación recursos permite acceder sin el conocimiento de su ubicación física o la red (por ejemplo, que la construcción o la dirección IP).

la transparencia de concurrencia permite que varios procesos para operar simultáneamente usando recursos compartidos sin interferencia entre ellos.

la transparencia de replicación permite que varias instancias de los recursos que se utilizarán para aumentar la fiabilidad y rendimiento sin el conocimiento de las réplicas por los usuarios o programadores de aplicaciones.

La falta de transparencia permite la ocultación de errores, permitiendo a los usuarios y programas de aplicación para completar sus tareas a pesar del fracaso de los componentes de hardware o software.

transparencia movilidad permite el movimiento de los recursos y los clientes dentro de un sistema sin afectar el funcionamiento de los usuarios o programas.

rendimiento transparencia permite que el sistema puede reconfigurar para mejorar el rendimiento ya que las cargas varían.

escalar la transparencia permite que el sistema y las aplicaciones para expandirse en escala sin cambio a la estructura del sistema o los algoritmos de aplicación.

Los dos más importantes son las transparencias acceso y transparencia de ubicación; su presencia o ausencia afecta más fuertemente la utilización de los recursos distribuidos. Ellos se refieren a veces juntos como *transparencia de la red*.

Como ejemplo de transparencia de acceso, considere una interfaz gráfica de usuario con carpetas, que es lo mismo si los archivos dentro de la carpeta local o remoto. Otro ejemplo es un API para archivos que utiliza las mismas operaciones para acceder a ambos archivos locales y remotos (véase el Capítulo 12). Como un ejemplo de la falta de transparencia de acceso, considere un sistema distribuido que no permite acceder a archivos en un ordenador remoto a menos que haga uso del programa ftp para hacerlo.

nombres o direcciones URL de recursos web son la ubicación transparente debido a que la parte de la URL que identifica un nombre de dominio del servidor web se refiere a un nombre de equipo en un dominio, en lugar de una dirección de Internet. Sin embargo, la movilidad URL no son transparentes, porque la página web personal de una persona no puede moverse a su nuevo lugar de trabajo en un dominio diferente - todos los enlaces en otras páginas todavía apuntará a la página original.

En general, los identificadores, tales como direcciones URL que incluyen los nombres de dominio de los ordenadores impiden la transparencia de replicación. Aunque el DNS permite que un nombre de dominio para referirse a varios ordenadores, que recoge sólo uno de ellos cuando se busca un nombre. Desde un esquema de replicación en general, tiene que ser capaz de acceder a todos los sistemas participantes, sería necesario acceder a cada una de las entradas DNS por su nombre.

Como ilustración de la presencia de la transparencia de red, considere el uso de una dirección de correo electrónico, como *Fred.Flintstone@stoneit.com.* La dirección consiste en el nombre de un usuario y un nombre de dominio. El envío de correo a un usuario, no se trata de conocer su ubicación física o la red. Tampoco el procedimiento para enviar un mensaje de correo electrónico dependerá de la ubicación del receptor. Por lo tanto el correo electrónico en Internet ofrece tanto la ubicación y la transparencia de acceso (es decir, la red de transparencia).

transparencia fracaso también puede ser ilustrado en el contexto de correo electrónico, que finalmente se entrega, incluso cuando los servidores o enlaces de comunicación fallan. Las fallas están enmascarados por intentar retransmitir mensajes hasta que se entregan correctamente, incluso si se toma varios días. Middleware convierte generalmente los fracasos de redes y procesos en excepciones a nivel de programación (véase el capítulo 5 para una explicación).

Para ilustrar la transparencia movilidad, consideremos el caso de los teléfonos móviles. Supongamos que tanto la persona que llama y el destinatario de la llamada están viajando en tren en diferentes partes de un país, moviéndose

de un entorno (celular) a otro. Consideramos que el teléfono de la persona que llama como el cliente y el teléfono de la parte llamada como un recurso. Los dos usuarios de teléfonos que hacen la llamada no son conscientes de la movilidad de los teléfonos (el cliente y los recursos) entre las células.

La transparencia se esconde y hace anónimos los recursos que no son de interés directo para la tarea en cuestión para los usuarios y programadores de aplicaciones. Por ejemplo, generalmente es deseable que los recursos de hardware similares a asignarse indistintamente para realizar una tarea - la identidad de un procesador utilizado para ejecutar un proceso generalmente se oculta para el usuario y permanece en el anonimato. Como se ha señalado en la Sección 1.3.2, esto puede no ser siempre lo que se requiere, por ejemplo, un viajero que se conecta un ordenador portátil a la red local en cada oficina visitó deben hacer uso de los servicios locales, tales como el servicio de envío de correo, utilizando servidores diferentes en cada lugar. Incluso dentro de un edificio, lo normal es que los arreglos para que un documento se imprime en una impresora llamada en particular: por lo general una que está cerca del usuario.

1.5.8 Calidad de servicio

Una vez que los usuarios disponen de la funcionalidad que requieren de un servicio, tales como el servicio de archivos en un sistema distribuido, podemos pasar a preguntar acerca de la calidad del servicio prestado. Las principales propiedades funcionales de los sistemas que afectan a la calidad del servicio experimentada por los clientes y usuarios *fiabilidad, seguridad* y *actuación. Adaptabilidad* para satisfacer las cambiantes configuraciones del sistema y la disponibilidad de recursos ha sido reconocido como un aspecto importante de la calidad del servicio.

problemas de fiabilidad y seguridad son fundamentales en el diseño de la mayoría de los sistemas informáticos. El aspecto de rendimiento de calidad de servicio se definió originalmente en términos de capacidad de respuesta y el rendimiento computacional, pero se ha redefinido en términos de capacidad para cumplir con las garantías de puntualidad, como se explica en los párrafos siguientes.

Algunas aplicaciones, como aplicaciones multimedia, manejar *los datos de tiempo crítico* - flujos de datos que se requieren para ser procesados o transferido desde un proceso a otro a una tasa fija. Por ejemplo, un servicio de películas podría consistir en un programa cliente que está recuperando de una película a partir de un servidor de vídeo y presentarla en la pantalla del usuario. Para un resultado satisfactorio los cuadros sucesivos de vídeo tienen que ser mostrada al usuario dentro de unos límites de tiempo especificados.

De hecho, la QoS abreviatura efectivamente se ha requisado para referirse a la capacidad de los sistemas para cumplir tales plazos. Su logro depende de la disponibilidad de los recursos informáticos y de red necesarios en los momentos adecuados. Esto implica un requisito para que el sistema proporcione los recursos informáticos y de comunicación tienen garantías suficientes para permitir que las aplicaciones para completar cada tarea en el tiempo (por ejemplo, la tarea de mostrar un cuadro de video).

Las redes que se utilizan comúnmente en la actualidad tienen un alto rendimiento - por ejemplo, BBC iPlayer realiza generalmente aceptablemente - pero cuando las redes están fuertemente cargados su rendimiento puede deteriorarse, y se proporcionan sin garantías. QoS se aplica a los sistemas operativos, así como redes. Cada recurso crítico debe reservarse por las aplicaciones que requieren calidad de servicio, y debe ser gestores de recursos que proporcionan garantías. Las peticiones de reserva que no se pueden cumplir son rechazados. Estas cuestiones se abordarán en el capítulo 20.

1.6 Estudio de caso: La World Wide Web

La red mundial [me www.w3.org , Berners-Lee, 1991] es un sistema en evolución para la publicación y el acceso a recursos y servicios a través de Internet. A través de los navegadores web comúnmente disponibles, los usuarios recuperar y visualizar documentos de muchos tipos, escuchan a los flujos de audio y vídeo corrientes de vista, e interactúan con un conjunto ilimitado de servicios.

La web comenzó la vida en el Centro Europeo de Investigación Nuclear (CERN), Suiza, en 1989 como un vehículo para el intercambio de documentos entre una comunidad de físicos conectados por Internet [Berners-Lee, 1999]. Una característica clave de la Web es que proporciona una *hipertexto* estructura entre los documentos que almacena, lo que refleja el requisito de los usuarios a organizar sus conocimientos. Esto significa que los documentos contienen *(enlaces* o *hipervínculos)* - referencias a otros documentos y recursos que también están almacenados en la Web. Es fundamental para la experiencia del usuario de la web que cuando se encuentran con una imagen o un pedazo de texto dentro de un documento dado, esto con frecuencia se acompaña de enlaces a documentos relacionados y otros recursos. La estructura de enlaces puede ser arbitrariamente compleja y el conjunto de recursos que se pueden añadir es ilimitado - la 'web' de enlaces es de hecho en todo el mundo. Bush [1945] concebido de estructuras hipertextuales hace más de 50 años; fue con el desarrollo de Internet que esta idea podría manifestarse en una escala mundial.

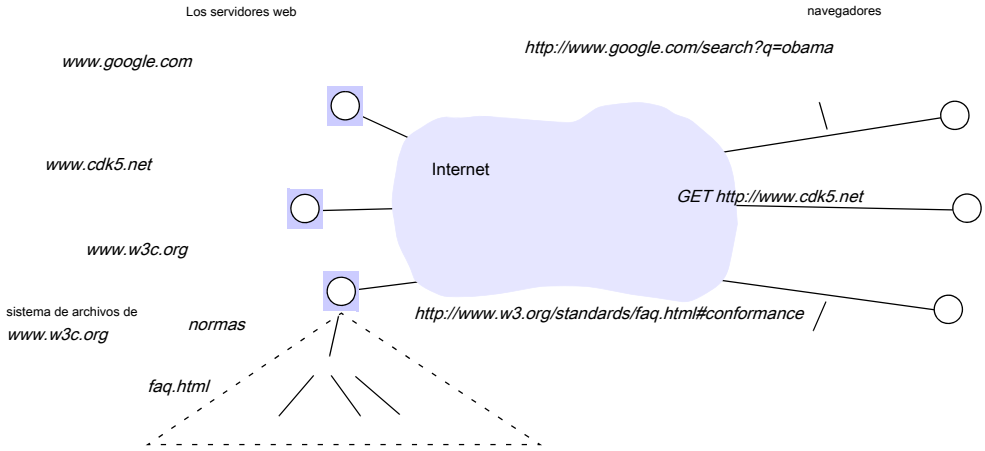
La Web es una *abierto* sistema: puede ser extendido y aplicado de nuevas formas sin alterar su funcionalidad existente (véase la Sección 1.5.2). En primer lugar, su funcionamiento se basa en estándares de comunicación y estándares de documentos o de contenidos que se publican libre y ampliamente implementadas. Por ejemplo, hay muchos tipos de navegador, cada uno en muchos casos implementados en varias plataformas; y hay muchas implementaciones de servidores web. Cualquier navegador conformes puede recuperar recursos desde cualquier servidor conforme. Así, los usuarios tienen acceso a los navegadores en la mayoría de los dispositivos que utilizan, desde teléfonos móviles a los ordenadores de sobremesa.

En segundo lugar, la Web está abierto con respecto a los tipos de recursos que pueden ser publicados y compartidos en él. En su forma más simple, un recurso en la web es una página web o algún otro tipo de *contenido* que se pueden presentar al usuario, tales como archivos multimedia y documentos en formato de documento portátil. Si alguien inventa, por ejemplo, un nuevo formato de imagen de almacenamiento, a continuación, las imágenes en este formato de inmediato pueden ser publicados en la Web. Los usuarios requieren un medio de visualización de imágenes en este nuevo formato, pero los navegadores están diseñados para dar cabida a la nueva funcionalidad contenido presentación en forma de aplicaciones 'ayuda' y 'plug-ins'.

La Web se ha movido más allá de estos recursos de datos simples, a servicios, tales como compras electrónicas de mercancías. Se ha evolucionado sin cambiar su estructura básica. La web se basa en tres principales componentes tecnológicos estándar:

- el lenguaje de marcado de hipertexto (HTML), un lenguaje para especificar el contenido y el diseño de páginas a medida que se muestran por los navegadores web;
- Los localizadores uniformes de recursos (URL), también conocidos como identificadores uniformes de recursos (URI), que identifican los documentos y otros recursos almacenados como parte de la Web;
- una arquitectura de sistema cliente-servidor, con las normas estándar para la interacción (Protocolo de transferencia de hipertexto al - HTTP) por el cual los navegadores y otros clientes traen documentos y otros recursos de los servidores web. La figura 1.7 muestra algunos servidores web y navegadores hacer peticiones a ellos. Es una característica importante que los usuarios pueden localizar y gestionar sus propios servidores web en cualquier lugar en Internet.

Figura 1.7 Los servidores web y navegadores web



Ahora discutimos estos componentes a su vez, y al hacerlo, explicar el funcionamiento de los navegadores y servidores web cuando un usuario obtiene páginas web y los clics en los enlaces dentro de ellos.

• **HTML** El HyperText Markup Language [www.w3.org ll] Se utiliza para especificar el texto y las imágenes que componen el contenido de una página web, y para especificar la forma en que se presentan y formateados para su presentación al usuario. Una página web contiene elementos tales estructurados como los encabezamientos, párrafos, tablas e imágenes. HTML también se utiliza para especificar enlaces y recursos, que están asociados con ellos.

Los usuarios pueden producir HTML a mano, usando un editor de texto estándar, pero más comúnmente usar un editor HTML-consciente 'WYSIWYG' que genera HTML a partir de un diseño que crean gráficamente. Una pieza típica de texto HTML siguiente:

```
<IMG SRC = “http://www.cdk5.net/WebExample/Images/earth.jpg”>                1
<P>                                                                          2
Bienvenido a la Tierra! Los visitantes también pueden estar interesados en echar un vistazo a la   3
<A HREF = "http://www.cdk5.net/WebExample/moon.html"> Luna </A>.             4
</ P>                                                                        5
```

Este texto HTML se almacena en un archivo que un servidor web puede acceder - digamos el archivo *earth.html*. Un navegador recupera el contenido de este archivo desde un servidor web - en este caso un servidor en un equipo llamado *www.cdk5.net*. El navegador lee el contenido devuelto por el servidor y lo hace en el texto con formato e imágenes presentados en una página web en la forma familiar. Sólo el navegador - no el servidor - interpreta el texto HTML. Sin embargo, el servidor hace informar al navegador del tipo de contenido que está regresando, para distinguirlo de, digamos, un documento en formato de documento portátil. El servidor puede inferir el tipo de contenido de la extensión de nombre de archivo '.html'.

Tenga en cuenta que las directivas de HTML, conocida como *etiquetas,* están encerrados por corchetes angulares, como < *P>*. Línea 1 del ejemplo identifica un archivo que contiene una imagen de presentación. Su URL es *http://www.cdk5.net/WebEx*
Las líneas 2 y 5 son directivas para empezar y acabar un párrafo, respectivamente. Las líneas 3 y 4 contienen texto que se mostrará en la página web en el formato de párrafo estándar.

La línea 4 especifica un enlace en la página web. Contiene la palabra 'Luna', rodeado por dos etiquetas HTML relacionados, *< A HREF ...>* y *</ A>*. El texto entre estas etiquetas es lo que aparece en el enlace tal como se presenta en la página web. La mayoría de los navegadores están configurados para mostrar el texto de los enlaces subrayados por defecto, por lo que el usuario verá en ese párrafo es:

Bienvenido a la Tierra! Los visitantes también pueden estar interesados en echar un vistazo a la Luna .

El navegador registra la asociación entre el texto que se muestra del enlace y la dirección URL contenida en el *< A HREF ...>* tag - en este caso:

http://www.cdk5.net/WebExample/moon.html

Cuando el usuario hace clic en el texto, el navegador recupera el recurso identificado por la URL correspondiente y lo presenta al usuario. En el ejemplo, el recurso es un archivo HTML que especifica una página web sobre la Luna.

• **URL** El propósito de un localizador de recursos [www.w3.org III] Es identificar un recurso. De hecho, el término utilizado en los documentos de arquitectura web es Uniform Resource *identificador (* URI), pero en este libro la URL mejor conocido término se utiliza cuando se produzcan confusiones. Navegadores examinan las direcciones URL a fin de acceder a los recursos correspondientes. A veces, el usuario escribe una dirección URL en el navegador. Más comúnmente, el navegador busca la URL correspondiente cuando el usuario hace clic en un enlace o selecciona uno de sus marcadores "; o cuando el navegador se vende a un recurso incrustado en una página web, como una imagen.

Cada URL, en su forma completa, absoluto, tiene dos componentes de nivel superior:

esquema: esquema específico de identificador

El primer componente, el 'esquema', declara el tipo de URL que es esto. Se requieren direcciones URL para identificar una variedad de recursos. Por ejemplo, *mailto: joe@anISP.net* identifica la dirección de correo electrónico de un usuario; *ftp://ftp.downloadIt.com/software/aProg.exe* identifica un archivo que se va a recuperar usando el protocolo de transferencia de archivos (FTP) en lugar del protocolo HTTP usado más comúnmente. Otros ejemplos de esquemas son 'tel' (se usa para especificar un número de teléfono para marcar, que es particularmente útil cuando se navega en un teléfono móvil) y 'etiqueta' (utilizado para identificar una entidad arbitraria).

La Web está abierto con respecto a los tipos de recursos que se pueden utilizar para el acceso, en virtud de los designadores de esquema en el URL. Si alguien inventa un nuevo tipo útil de recursos 'Widget' - tal vez con su propio esquema de direccionamiento para los widgets de localización y su propio protocolo para acceder a ellos - entonces el mundo puede empezar a utilizar URLs de tipo *Reproductor: ....* Por supuesto, los navegadores deben tener la capacidad de utilizar el nuevo protocolo 'Widget', pero esto se puede hacer mediante la adición de un plug-in.

URL HTTP son los más ampliamente utilizados, para acceder a recursos utilizando el protocolo HTTP estándar. Una dirección URL HTTP tiene dos funciones principales: identificar qué servidor Web mantiene el recurso, y para identificar cuál de los recursos en ese servidor se requiere. La figura 1.7 muestra tres navegadores que emiten solicitudes de recursos gestionados por tres servidores web. El navegador más alta es la emisión de una consulta a un motor de búsqueda. El navegador medio requiere la página por defecto de otro sitio web. El navegador más inferior requiere una página web que se especifica en su totalidad, incluyendo un nombre de ruta relativa al servidor. Los archivos para un servidor web determinada se mantienen en uno o más subárboles (directorios) de sistema de archivos del servidor, y cada recurso se identifica por un nombre de ruta relativa al servidor.

En general, las direcciones URL HTTP son de la siguiente forma:

http: // servidor: [? consulta] [puerto] [/ vía de acceso] [#fragment]

donde los artículos que figuran entre corchetes son opcionales. Una URL HTTP completa siempre comienza con la cadena 'http: //' seguido de un nombre de servidor, expresado como un nombre de Sistema de nombres de dominio (DNS) (véase la Sección 13.2). nombre DNS del servidor es seguido opcionalmente por el número de la 'puerto' en el que escucha el servidor para las solicitudes (véase el capítulo 4), que es 80 por defecto. Luego viene un nombre de ruta opcional de los recursos del servidor. Si esto está ausente entonces se requiere la página web por defecto del servidor. Por último, el URL termina opcionalmente en un componente de consulta - por ejemplo, cuando un usuario envía las entradas en una forma tal como la página de un motor de búsqueda consulta - y / o un identificador de fragmento, que identifica un componente del recurso.

Considere las direcciones URL:

http://www.cdk5.net
http://www.w3.org/standards/faq.html#conformance
http://www.google.com/search?q=obama

Estos se pueden desglosar de la siguiente manera:

El nombre DNS del nombre de ruta del servidor		*Consulta*	*Fragmento*
www.cdk5.net	(defecto)	(ninguna)	(ninguna)
www.w3.org	normas / faq.html	(ninguna)	Introducción
www.google.com búsqueda		q = obama	(ninguna)

El primer URL designa la página predeterminada suministrada por *www.cdk5.net*. El siguiente identifica un fragmento de un archivo HTML cuyo camino es el nombre *normas / faq.html* en relación con el servidor *www.w3.org*. identificador de fragmento (especificado después de que el carácter '#' en la URL) es *introducción,* y un navegador buscará que el identificador de fragmento en el texto HTML después de que se haya descargado todo el archivo. El tercer URL especifica una consulta a un motor de búsqueda. La ruta identifica un programa llamado 'búsqueda' y la cadena después de que el '?' carácter codifica una cadena de consulta suministrado como argumentos para este programa. Se discuten las direcciones URL que identifican recursos de programación con más detalle si tenemos en cuenta las características más avanzadas a continuación.

Publicar un recurso: Mientras que la Web tiene un modelo claramente definido para acceder a un recurso de su URL, los métodos exactos de recursos de publicación en la web son dependientes de la implementación del servidor web. En cuanto a los mecanismos de bajo nivel, el método más simple de la publicación de un recurso en la web es colocar el archivo correspondiente en un directorio que el servidor web puede acceder. Conocer el nombre del servidor *S* y un nombre de ruta del archivo *PAG* que el servidor puede reconocer, el usuario construye la URL como *http: // S / P.* El usuario pone esta URL en un enlace de un documento existente o distribuye el URL para otros usuarios, por ejemplo, por correo electrónico.

Es común que este tipo de preocupaciones que se oculta de los usuarios cuando generan contenido. Por ejemplo, 'bloggers' suelen utilizar herramientas de software, ellos implementan como páginas web, para crear colecciones organizadas de las páginas del diario. páginas de productos para el sitio web de una empresa suelen ser creados usando una *sistema de gestión de contenidos,* de nuevo por

interactuando directamente con el sitio web a través de las páginas Web de administración. El sistema de base de datos o archivos en el que se basan las páginas de producto es transparente.

Por último, Huang *et al. [*2000] proporcionan un modelo para la inserción de contenidos en la Web con una mínima intervención humana. Esto es particularmente pertinente cuando los usuarios necesitan para extraer el contenido de una variedad de dispositivos, tales como cámaras, para su publicación en páginas web.

• **HTTP** El Protocolo de transferencia de hipertexto [www.w3.org IV] Define las formas en que los navegadores y otros tipos de clientes interactúan con los servidores web. Capítulo 5 considerará HTTP con más detalle, pero aquí esbozar sus principales características (restringir nuestra discusión a la recuperación de recursos en archivos):

interacciones de petición-respuesta: HTTP es un protocolo 'petición-respuesta'. El cliente envía un mensaje de solicitud al servidor que contiene la dirección URL de los recursos deseados. El servidor busca el nombre de ruta y, si existe, devuelve el contenido del recurso en un mensaje de respuesta al cliente. De lo contrario, envía de vuelta una respuesta de error tales como el familiar '404 no encontrado'. HTTP define un pequeño conjunto de operaciones o *métodos* que se puede realizar en un recurso. Los más comunes son GET, para recuperar los datos desde el recurso, y POST, que facilita datos sobre el recurso.

Tipos de contenido: Los navegadores no son necesariamente capaces de manejar todo tipo de contenido. Cuando un navegador hace una petición, que incluye una lista de los tipos de contenido que prefiere - por ejemplo, en principio, puede ser capaz de mostrar imágenes en formato 'GIF', pero no el formato 'JPEG'. El servidor puede ser capaz de tener esto en cuenta cuando se devuelve el contenido al navegador. El servidor incluye el tipo de contenido en el mensaje de respuesta para que el navegador sabrá cómo procesarlo. Las cuerdas que denotan el tipo de contenido se denominan tipos MIME, y ellos están estandarizados en el RFC 1521 [Liberado y Borenstein 1996]. Por ejemplo, si el contenido es de tipo 'text / html', entonces un navegador interpretará el texto como HTML y mostrarlo; si el contenido es de tipo 'image / gif', entonces el navegador se hacen como una imagen en formato 'GIF'; si el tipo de contenido es 'application / zip', entonces es de datos comprimidos en formato 'zip', y el navegador lanzará una aplicación de ayuda externa para descomprimirlo. El conjunto de acciones que un navegador tomará para un determinado tipo de contenido es configurable y puede cuidar a los lectores a comprobar estos valores por sus propios navegadores.

Uno de los recursos por solicitud: Clientes especificar un recurso por la petición HTTP. Si una página web contiene nueve imágenes, por ejemplo, el navegador emitirá un total de diez solicitudes por separado para obtener todo el contenido de la página. Los navegadores suelen hacer varias solicitudes al mismo tiempo, para reducir el retardo global para el usuario.

control de acceso simple: De manera predeterminada, cualquier usuario con conectividad de red a un servidor web puede acceder a cualquiera de sus recursos publicados. Si los usuarios desean restringir el acceso a un recurso, entonces se puede configurar el servidor para emitir un 'desafío' a cualquier cliente que lo solicite. El usuario que corresponde a continuación, tiene que demostrar que tienen el derecho a acceder al recurso, por ejemplo, tecleando una contraseña.

• **Las páginas dinámicas** Hasta ahora hemos descrito cómo los usuarios pueden publicar páginas web y otros contenidos almacenados en los archivos de la web. Sin embargo, gran parte de la experiencia de la Web de los usuarios es la de interactuar con los servicios en lugar de recuperar los datos. Por ejemplo, en la compra de un artículo en una tienda en línea, el usuario a menudo llena una *formulario web* para proporcionar datos personales o para especificar exactamente lo que desea comprar. Un formulario web es una web

la página que contiene instrucciones para los widgets de usuario y de entrada, tales como campos de texto y casillas de verificación. Cuando el usuario envía el formulario (normalmente pulsando un botón o la tecla 'retorno'), el navegador envía una petición HTTP a un servidor web, que contiene los valores que el usuario ha introducido.

Puesto que el resultado de la solicitud depende de la entrada del usuario, el servidor tiene que *proceso* la entrada del usuario. Por lo tanto la dirección URL o su componente inicial designa una *programa* en el servidor, no un archivo. Si la entrada del usuario es un razonablemente pequeño conjunto de parámetros a menudo es enviado como el *consulta* componente de la URL, utilizando el método GET; Alternativamente, se envía como datos adicionales en la solicitud utilizando el método POST. Por ejemplo, una solicitud que contenga la siguiente URL invoca un programa que se llama 'búsqueda' en www.google.com y especifica una cadena de consulta de 'Obama':

http://www.google.com/search?q=obama.

Ese 'buscar' programa produce el texto HTML como su salida, y el usuario verá una lista de páginas que contienen la palabra 'Obama'. (El lector puede entrar en una consulta en su motor de búsqueda favorito y observe la dirección URL que las pantallas del navegador cuando se devuelve el resultado.) El servidor devuelve el texto HTML que genera el programa como si se hubiera recuperado desde un archivo. En otras palabras, la diferencia entre el contenido estático obtienen de un archivo y el contenido que se genera dinámicamente es transparente para el navegador.

Un programa que ejecutan los servidores web para generar contenido para sus clientes que se conoce como un programa Common Gateway Interface (CGI). Un programa CGI puede tener cualquier funcionalidad específica de la aplicación, siempre y cuando se puede analizar los argumentos de que el cliente proporciona a la misma y producir contenidos del tipo requerido (generalmente texto HTML). El programa a menudo consultar o actualizar una base de datos en el procesamiento de la solicitud.

código descargado: Un programa CGI se ejecuta en el servidor. A veces los diseñadores de servicios web requieren algún código relacionado con el servicio a ejecutar dentro del navegador, en el ordenador del usuario. En particular, el código escrito en Javascript [www.netscape.com] Menudo se descarga con una página web que contiene un formulario, con el fin de proporcionar una interacción de mejor calidad con el usuario de lo que admite widgets estándar de HTML. Una página Javascriptenhanced puede dar la retroalimentación inmediata de usuario en las entradas no válidas, en lugar de obligar al usuario a comprobar los valores en el servidor, lo que llevaría mucho más tiempo.

Javascript también se puede utilizar para actualizar partes del contenido de una página web sin necesidad de ir a buscar una versión completamente nueva de la página y volver a representar la misma. se producen estos cambios dinámicos ya sea debido a una acción del usuario (por ejemplo, al hacer clic en un enlace o un botón de radio), o cuando el navegador adquiere nuevos datos del servidor que suministra la página web. En este último caso, ya que el momento de la llegada de los datos es ajeno a cualquier acción del usuario en el propio navegador, se denomina *asincrónico*. Una técnica conocida como *AJAX (* JavaScript asíncrono y XML) se utiliza en estos casos. AJAX se describe más completamente en la Sección

2.3.2.

Una alternativa a un programa Javascript es una *subprograma:* una aplicación escrita en lenguaje Java [Flanagan 2002], que el navegador descarga y ejecuta automáticamente cuando se obtiene una página web correspondiente. Los applets pueden tener acceso a la red y proporcionar interfaces de usuario personalizadas. Por ejemplo, 'chatear' aplicaciones a veces se implementan como applets que se ejecutan en los navegadores de los usuarios, junto con un programa de servidor. Los applets de enviar mensajes de texto de los usuarios al servidor, que a su vez lo distribuye a todos los applets para su presentación al usuario. Se discuten los applets en más detalle en la Sección 2.3.1.

Servicios web • Hasta ahora hemos hablado de la Web en gran medida desde el punto de vista de un usuario que opera un navegador. Pero los programas de otros navegadores que pueden ser clientes de la Web, también; de hecho, el acceso mediante programación a los recursos web es un lugar común.

Sin embargo, el HTML es inadecuada para la interoperación programática. Existe una creciente necesidad de intercambiar muchos tipos de datos estructurados en la Web, pero HTML está limitado en que no es extensible para aplicaciones más allá de la navegación información. HTML tiene un conjunto estático de estructuras tales como párrafos, y que están vinculados con la forma en que los datos van a ser presentados a los usuarios. El Extensible Markup Language (XML) (ver Sección 4.3.3) se ha diseñado como una forma de representar datos en estándar, formas estructuradas, específicos de la aplicación. En principio, los datos expresados en XML es portátil entre las aplicaciones, ya que es *auto-descripción:* Contiene los nombres, tipos y estructura de los elementos de datos dentro de ella. Por ejemplo, XML puede ser usado para describir los productos o información sobre los usuarios, para muchos servicios o aplicaciones diferentes. En el protocolo HTTP, los datos XML pueden ser transmitidos por el POST y GET operaciones. En AJAX puede ser utilizado para proporcionar datos a los programas de JavaScript en los navegadores.

recursos Web proporcionan operaciones específicas del servicio. Por ejemplo, en la tienda en amazon.com, operaciones de servicios web incluyen uno para pedir un libro y otro para comprobar el estado actual de un pedido. Como ya hemos mencionado, HTTP proporciona un pequeño conjunto de operaciones que son aplicables a cualquier recurso. Estos incluyen principalmente los métodos GET y POST sobre los recursos existentes y el PUT y DELETE operaciones, respectivamente. para la creación y eliminación de recursos web. Cualquier operación en un recurso puede ser invocada mediante uno de los métodos GET o POST, con contenido estructurado utiliza para especificar el de parámetros, resultados y respuestas de error operación. El RESTO llamada (Transferencia de estado representacional), la arquitectura de servicios web [2000] Fielding adopta este enfoque sobre la base de su extensibilidad: todos los recursos en la Web tiene una URL y responde a la misma serie de operaciones, aunque el procesamiento de las operaciones puede variar mucho de un recurso a otro. La otra cara de la extensibilidad de que puede haber una falta de robustez en el funcionamiento del software. Capítulo 9 describe, además, REST y da una mirada en profundidad en el marco de servicios web, que permite a los diseñadores de servicios web para describir a los programadores más específicamente lo específico del servicio de operaciones están disponibles y cómo los clientes deben tener acceso a ellos.

La discusión de la Web • el fenomenal éxito de la Web se basa en la facilidad relativa con la que muchas fuentes individuales y organizacionales pueden publicar recursos, la idoneidad de la estructura de hipertexto para organizar muchos tipos de información, y la apertura de su arquitectura de sistema. Las normas en que se basa su arquitectura son simples y que fueron publicadas ampliamente en una etapa temprana. Ellos han permitido a muchos nuevos tipos de recursos y servicios que deben integrarse.

El éxito de la Web desmiente algunos problemas de diseño. En primer lugar, su modelo de hipertexto es deficiente en algunos aspectos. Si se elimina o mueve un recurso, los llamados 'colgando' enlaces a ese recurso es posible que queden, provocando frustración para los usuarios. Y está el conocido problema de los usuarios que consiguen 'perdido en el hiperespacio'. Los usuarios a menudo se encuentran confundidos, tras muchos enlaces dispares, haciendo referencia a las páginas de una colección dispar de las fuentes y de dudosa fiabilidad en algunos casos.

Los motores de búsqueda son una alternativa muy popular para los siguientes enlaces como un medio para encontrar información en la Web, pero éstos son imperfectos en la producción de lo que el usuario tiene la intención específica. Un enfoque a este problema, ejemplificado en el recurso

Description Framework [www.w3.org V], Es producir un vocabulario estándar, sintaxis y la semántica para expresar metadatos acerca de las cosas en nuestro mundo, y para encapsular que los metadatos correspondientes en los recursos web para el acceso mediante programación. En lugar de buscar palabras que se producen en las páginas web, los programas pueden entonces, en principio, realizar búsquedas en contra de los metadatos para compilar listas de enlaces relacionados que se basan en la coincidencia semántica. En conjunto, la red de recursos de metadatos vinculados es lo que se entiende por el

web semántica.

Como la arquitectura del sistema de la Web se enfrenta a problemas de escala. servidores web más populares pueden experimentar muchos éxitos " por segundo, y como resultado de la respuesta de los usuarios pueden ser lentos. El capítulo 2 describe el uso de almacenamiento en caché de los navegadores y servidores proxy para aumentar la capacidad de respuesta, y la división de la carga del servidor a través de grupos de ordenadores.

1.7 Resumen

Los sistemas distribuidos están en todas partes. Internet permite a los usuarios en todo el mundo para acceder a sus servicios dondequiera que se encuentren. Cada organización gestiona una intranet, que proporciona servicios locales y servicios de Internet para los usuarios locales y en general proporciona servicios a otros usuarios de Internet. sistemas distribuidos pequeñas pueden ser construidos a partir de equipos móviles y otros pequeños dispositivos computacionales que están conectados a una red inalámbrica.

el intercambio de recursos es el principal factor de motivación para la construcción de sistemas distribuidos. Los recursos tales como impresoras, archivos, páginas web o registros de la base son gestionados por los servidores del tipo apropiado. Por ejemplo, los servidores web gestionan páginas web y otros recursos web. Los recursos son accesibles a los clientes - por ejemplo, los clientes de los servidores web son generalmente llamados navegadores.

La construcción de sistemas distribuidos produce muchos retos:

Heterogeneidad: Deben construirse a partir de una variedad de diferentes redes, sistemas operativos, hardware y lenguajes de programación. Los protocolos de comunicación de Internet enmascaran la diferencia en las redes, middleware y pueden hacer frente a las otras diferencias.

Franqueza: Los sistemas distribuidos deben ser extensibles - el primer paso es la publicación de las interfaces de los componentes, pero la integración de los componentes escritos por diferentes programadores es un verdadero reto.

Seguridad: El cifrado se puede utilizar para proporcionar una protección adecuada de los recursos compartidos y para mantener en secreto la información sensible cuando se transmite en mensajes a través de una red. Ataques de denegación de servicio siguen siendo un problema.

escalabilidad: Un sistema distribuido es escalable si el costo de añadir un usuario es una cantidad constante en términos de los recursos que deben ser añadidos. Los algoritmos utilizados para el acceso compartido de datos deben evitar los cuellos de botella de rendimiento y los datos deben ser estructurados jerárquicamente para obtener los mejores tiempos de acceso. Frecuentemente los datos de acceso pueden ser replicados.

el control de fallos: Cualquier proceso, el ordenador o la red pueden fallar independientemente de los otros. Por lo tanto, cada componente tiene que ser consciente de las posibles formas en que

los componentes que depende puede fallar y ser diseñado para hacer frente a cada uno de estos fracasos adecuadamente.

concurrencia: La presencia de múltiples usuarios en un sistema distribuido es una fuente de solicitudes simultáneas a sus recursos. Cada recurso debe ser diseñado para ser seguro en un entorno concurrente.

Transparencia: El objetivo es hacer que ciertos aspectos de la distribución invisible para el programador de la aplicación para que sólo se necesitan preocuparse con el diseño de su aplicación particular. Por ejemplo, no necesita preocuparse por su ubicación o los detalles de cómo sus operaciones se accede por otros componentes, o si va a ser replicado o migrar. Incluso los fracasos de las redes y los procesos pueden ser presentados a los programadores de aplicaciones en forma de excepciones - pero deben ser manejados.

Calidad de servicio. No es suficiente para proporcionar acceso a los servicios en los sistemas distribuidos. En particular, también es importante proporcionar garantías en cuanto a las cualidades asociadas con dicho acceso de servicio. Ejemplos de tales cualidades incluyen parámetros relacionados con el rendimiento, la seguridad y la fiabilidad.

CEREMONIAS

1.1 Dar cinco tipos de recursos de hardware y cinco tipos de datos o recursos de software que pueden útilmente ser compartida. Dar ejemplos de su uso compartido como ocurre en la práctica en los sistemas distribuidos. *páginas 2, 14*

1.2 ¿Cómo podrían ser los relojes de dos equipos que están conectados por una red local sincronizado sin referencia a una fuente horaria externa? ¿Qué factores limitan la exactitud del procedimiento que usted ha descrito? ¿Cómo puede ser sincronizado los relojes en un gran número de ordenadores conectados a través de Internet? Discutir la exactitud de este procedimiento. *página 2*

1.3 Tenga en cuenta las estrategias de implementación para juegos multijugador masivos en línea como se discute en la Sección 1.2.2. En particular, ¿qué ventajas ves en la adopción de un enfoque de servidor único para representar el estado del juego multijugador? ¿Qué problemas se puede identificar y cómo se pueden resolver? *página 5*

1.4 Un usuario llega a una estación de ferrocarril que nunca ha visitado antes, llevando una PDA que es capaz de redes inalámbricas. Sugiere cómo el usuario podría estar provisto de información sobre los servicios locales y comodidades en esa estación, sin introducir el nombre o los atributos de la estación. ¿Qué desafíos técnicos deben superar? *página 13*

1.5 Compare y contraste de computación nube con la computación cliente-servidor más tradicional? Lo novedoso de la computación en nube como un concepto? *páginas 13, 14*

1.6 Utilizar la World Wide Web como un ejemplo para ilustrar el concepto de compartir recursos, cliente y el servidor. ¿Cuáles son las ventajas y desventajas de HTML, URL y HTTP como tecnologías básicas para navegar por la información? ¿Alguna de estas tecnologías adecuadas como base para la computación cliente-servidor en general? *páginas 14, 26*

1.7 Un programa de servidor escrito en un idioma (por ejemplo, C ++) proporciona la implementación de un objeto BLOB que está destinado a ser visitada por los clientes que pueden estar escritas en un lenguaje diferente (por ejemplo, Java). Los equipos cliente y servidor pueden tener un hardware diferente, pero todos ellos están unidos a un internet. Describir los problemas debidos a cada uno de los cinco aspectos de la heterogeneidad que deben ser resueltos para que sea posible que un objeto de cliente para invocar un método en el objeto de servidor.

página 16

1.8 Un sistema distribuido abierto permite nuevos servicios para compartir recursos tales como el BLOB objeto en el ejercicio 1.7 que se añade y se accede por una variedad de programas de cliente. Discutir en el contexto de este ejemplo, en qué medida las necesidades de apertura difieren de los de la heterogeneidad.

página 17

1.9 Supongamos que las operaciones del objeto BLOB se dividen en dos categorías - las operaciones públicas que están disponibles para todos los usuarios y las operaciones protegidas que están disponibles sólo para ciertos usuarios con nombre. Estado todos los problemas involucrados en asegurar que sólo los usuarios con nombre pueden utilizar una operación protegida. Suponiendo que el acceso a una operación protegida proporciona información que no debe ser revelada a todos los usuarios, lo que más problemas surgen?

página 18

1.10 El servicio INFO gestiona un potencial muy grande de recursos, cada uno de los cuales puede ser visitada por los usuarios a través de Internet por medio de una llave (un nombre de cadena). Discutir un enfoque para el diseño de los nombres de los recursos que logra la mínima pérdida de rendimiento que el número de recursos en los aumentos de servicios. Sugiere cómo el servicio de información puede ser implementado con el fin de evitar cuellos de botella de rendimiento cuando el número de usuarios se hace muy grande.

página 19

1.11 Enumerar los tres principales componentes de software que pueden fallar cuando un cliente invoca un proceso de método en un objeto de servidor, dando un ejemplo de un fracaso en cada caso. Sugieren cómo se pueden hacer los componentes de tolerar fallos uno del otro.

página 21

1.12 Un proceso de servidor mantiene un objeto de información compartida, como el objeto de BLOB Ejercicio 1.7. Dar argumentos a favor y en contra de lo que el cliente solicita que se ejecuten simultáneamente por el servidor. En el caso de que se ejecutan simultáneamente, dar un ejemplo de posible 'interferencia' que puede producirse entre las operaciones de los diferentes clientes. Sugiere cómo se puede evitar tal interferencia.

página 22

1.13 Un servicio es implementado por varios servidores. Explicar por qué podrían ser recursos transferidos entre ellos. ¿Sería satisfactorio para los clientes de multidifusión todas las peticiones al grupo de servidores como una manera de lograr la transparencia de movilidad para los clientes? *página 23*

1.14 Recursos en los demás servicios World Wide Web y son nombrados por URL. Qué las iniciales URL denotan? Dar ejemplos de tres diferentes tipos de recursos web que pueden ser nombradas por direcciones URL.

página 26

1.15 Dé un ejemplo de una dirección URL HTTP. Enumerar los principales componentes de un URL HTTP, indicando cómo sus límites se indican y que ilustran cada uno de su ejemplo. ¿En qué medida es además una base transparente URL HTTP?

página 26

Esta página se ha dejado intencionadamente en blanco