Existen protocolos individuales que puede ser especificada por el proveedor y actúa de manera exclusiva., es decir éste define el protocolo y cómo funciona, pudiendo ser utilizada por otras organizaciones con permiso del propietario.

### 1.6.1 Estándares y protocolos

Un estándar es un conjunto de reglas muy rígido que ha sido avalado por la industria de networking y ratificado por una organización de estándares, como el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, Institute of Electrical and Electronics Engineers) o el Grupo de trabajo de ingeniería de Internet (IETF), con la finalidad de asegurar el éxito en un determinado proceso. Su uso asegura que productos de diferentes fabricantes puedan funcionar conjuntamente para lograr comunicaciones eficientes. El protocolo es también un conjunto de reglas, pero más flexibles, admiten diferentes modos y prestaciones que pueden ser usados o no según las necesidades y decisiones de los usuarios o administradores de red.

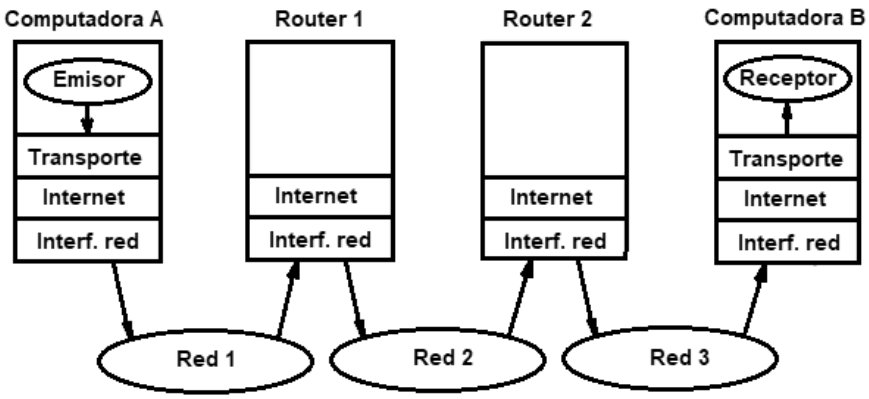
Una red para funcionar apropiadamente, utiliza todo un conjunto de reglas y protocolos denominadas comúnmente suite de protocolos de red. Un ejemplo del uso de una suite de protocolos de red es la comunicación entre un servidor Web y un explorador Web, esta interacción utiliza una cantidad de protocolos y estándares en el proceso de intercambio de información entre ellos. Los distintos protocolos trabajan en conjunto para asegurar que ambas partes reciben y entienden los mensajes, tenemos:

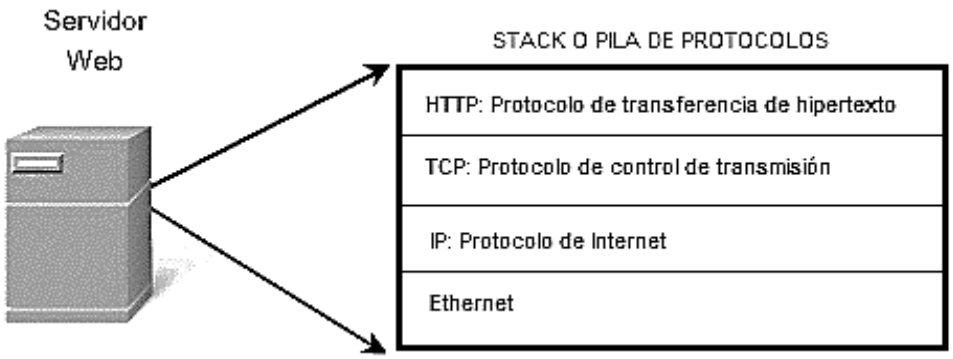
- Protocolo de aplicación: el Protocolo de transferencia de hipertexto (HTTP) rige la forma en que interactúan un servidor Web y un cliente Web. HTTP define el contenido y el formato de las solicitudes y respuestas intercambiadas entre cliente y servidor. Tanto el cliente como el software del servidor Web implementan el HTTP como parte de la aplicación. HTTP se basa en otros protocolos para regular la forma en que se transportan los mensajes entre el cliente y el servidor.

- Protocolo de transporte: el Protocolo de control de transmisión (TCP) administra las conversaciones individuales entre servidores Web y clientes Web. TCP divide los mensajes HTTP en pequeñas partes, denominadas segmentos, para enviarlas al cliente de destino. También es responsable de controlar el tamaño y los intervalos a los que se intercambian los mensajes entre el servidor y el cliente.

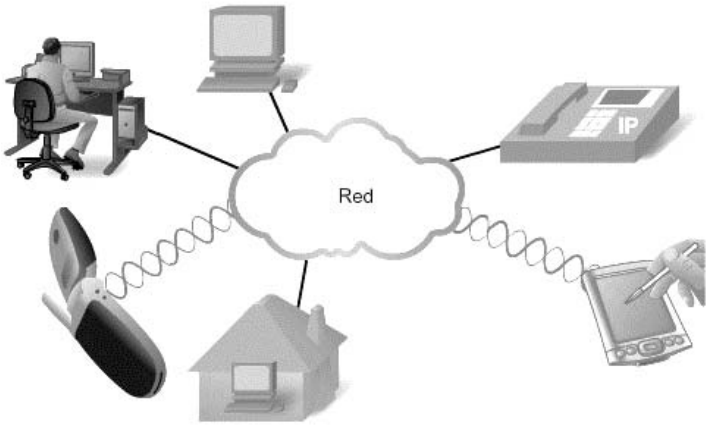
- Protocolo de internetwork: el Protocolo de Internet (IP). El IP es responsable de tomar los segmentos formateados del TCP, encapsularlos en paquetes, asignar las direcciones apropiadas y seleccionar la mejor ruta al host de destino.

- Protocolos de acceso a la red: dos funciones, enlace de datos y transmisión física. Los protocolos de administración de enlace de datos toman paquetes IP y los formatean para transmitirlos por los medios. Los estándares y protocolos de los medios físicos rigen de qué manera se envían las señales por los medios y cómo las interpretan los clientes que las reciben. Las tarjetas de interfaz de red implementan los estándares apropiados para los medios que se utilizan.



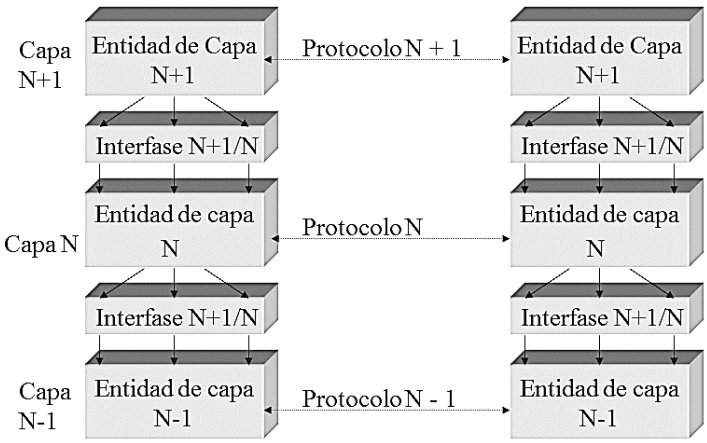



Los protocolos de red describen las funciones que se producen durante las comunicaciones de red, pero no describen cómo lograr una función en particular, lo que asegura que sea independiente de la tecnología. Así por ejemplo no se especifica que software se utiliza para desarrollar las aplicaciones o el sistema operativo usado, esta característica asegura que dispositivos distintos puedan acceder a las mismas aplicaciones sin restricción alguna.




### 1.6.2 Modelo en capas

Este modelo de funcionamiento de las redes, parte del hecho de que existen varios protocolos trabajando en un stack, describe el funcionamiento de los protocolos que se produce en cada capa y la interacción con las capas que se encuentran por encima y por debajo de ellas. Esto facilita el diseño de protocolos, ya que los protocolos que operan en una capa específica saben a priori sobre qué información actúan, y una interfaz definida para las capas superiores e inferiores, esto asegura la interoperabilidad, la competitividad y el trabajo colaborativo entre los diferentes desarrolladores

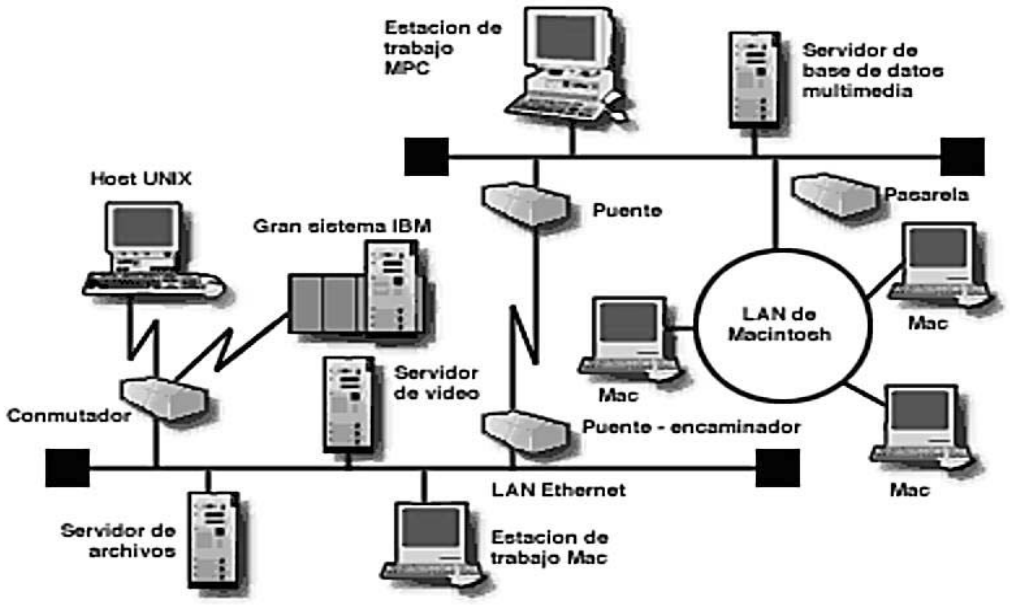



Modelo en capas

Existen dos tipos básicos de modelos de networking: modelos de protocolo y modelos de referencia, un modelo de protocolo proporciona un modelo que coincide exactamente con la estructura de una suite de protocolo en particular, mientras que un modelo de referencia proporciona una referencia común para mantener la consistencia dentro de todos los tipos de protocolos y servicios de red. Un modelo de referencia no está pensado para ser una especificación de implementación ni para proporcionar un nivel de detalle suficiente para definir de forma precisa los servicios de la arquitectura de red.

## 1.7 SISTEMAS MULTIMEDIA EN REDES

Las redes multimedia, como cualquier otra red, también presentan las características de permitir buscar rápidamente, localizar y obtener información de importancia de las bases de datos internas y los servidores de medios de comunicación, de depósitos mundiales o los foros electrónicos que ya existen sobre cualquier tema. Los usuarios deberán poder acceder a un almacén de información mundial. La diferencia fundamental está en el tipo de datos que se manejan y las exigencias de los usuarios en la calidad de servicio y la velocidad del mismo.




Actualmente se está desarrollando e iniciando el despliegue de un conjunto de protocolos y tecnologías de red con las que se pretende sentar las bases necesarias para que la comunicación multimedia en tiempo real a través de Internet sea tan accesible como la comunicación de texto y datos. Además, se pretende que estos nuevos sistemas puedan interoperar con el sistema de conferencia en tiempo real más extendido: la red telefónica.

Para lograr un sistema de telecomunicación universal en el que tengan cabida contenidos tan diversos como correo-e, vídeo, voz o fax, entre otros muchos, son necesarios avances en diversos campos. Nos centraremos en la arquitectura y los protocolos propuestos por el IETF, dejando al margen tecnologías como las relacionadas con los formatos multimedia o las tecnologías e infraestructuras de red, middleware de seguridad y autenticación, así como aspectos organizativos: dotación y gestión de entornos y salas de videoconferencia, normativas y procedimientos de administración de los servicios, etc.

Un Sistema Multimedia hace uso de múltiples formatos para presentar la información. Texto, sonido, imágenes estáticas o en movimiento y videos son los formatos más comunes. Combinan estos formatos junto con elementos interactivos. Factores como su bajo coste o la amplia gama de posibilidades que nos ofrecen los Sistemas Multimedia Interactivos han incrementado su uso. Cabe destacar que los sensores son un componente importante dentro de un Sistema Multimedia Interactivo porque éstos captan eventos que pueden convertirse en respuestas sensoriales con la ayuda de una unidad computacional.

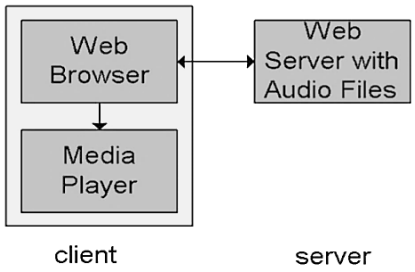
Las aplicaciones multimedia sobre redes tienen tres tipos de servicios requeridos

*a) Streaming de audio/video almacenado (Stored audio/video)*

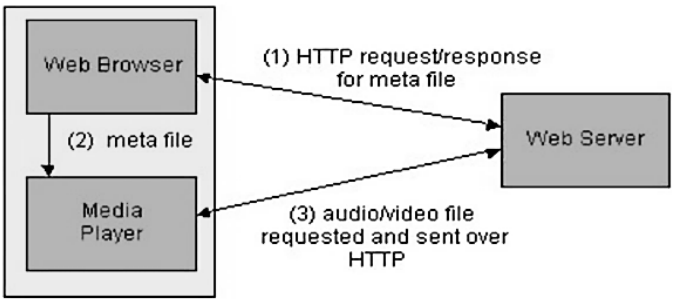
El contenido está almacenado en un servidor, el usuario solicita el contenido, el audio/video es descargado generando una latencia y hay aplicaciones en internet para su reproducción (play/pause/ffw/rew), por ejemplo, mbone VCR, estas aplicaciones deben encargarse de funciones como descompresión, corrección de errores y debe disponer de una interface gráfica de usuario como controles para la interavtividad. Algunos plug-ins podrían ser usados para alojar el reproductor de medios en la ventana del navegador

*b) Streaming de audio/video desde un servidor Web*

Los archivos de audio y video están almacenados en servidores Web, entonces el navegador requiere el archivo con un mensaje HTTP (Protocolo de Transferencia de Hipertexto) de requerimiento. El servidor Web envía el archivo en un mensaje de respuesta HTTP, la cabecera contiene el tipo de codificación usada para la información. El navegador o browser ejecuta el reproductor de medios trasladándole el archivo para su reproducción

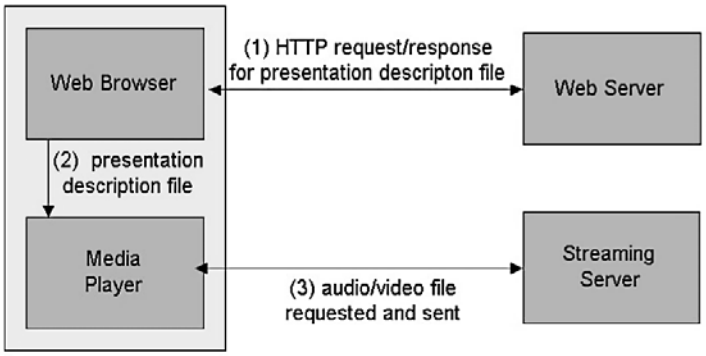



Alternativamente, se podría establecer una conexión entre servidor y reproductor de medios, en este caso el navegador requiere y recibe un meta-file (un file describiendo el objeto) y no recibe el archivo en sí mismo, el indicador de tipo en la cabecera hace alusión al tipo de reproductor o aplicación de medios requerido el browser lanza el reproductor y le traslada el meta-file, el reproductor establece una conexión TCP (protocolo de control de transferencia) y envía un requerimiento HTTP, sin embargo en este caso no se dispone de un control de la reproducción esta es ininterrumpida.

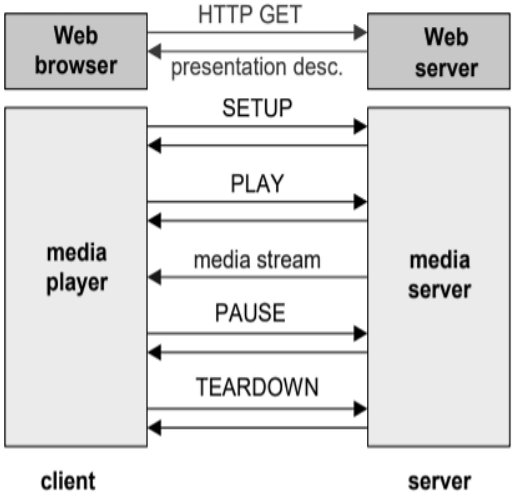



*c) Streaming desde un Servidor de streaming*

Esta arquitectura no requiere el uso del protocolo HTTP entre el servidor y el reproductor de medios, pudiendo simplemente trasladarse porciones de datos de los archivos de audio/video.




En este caso y sirviendo a este tipo de modelo se ha desarrollado un protocolo específico, RTSP (Real Time Streaming Protocol) para manejar HTML, imágenes, applets, etc. Está descrito en la norma RFC 2326, para la arquitectura cliente/servidor, soporta funciones de control de reproducción, pausa, retroceso rápido, reinicialización, reposicionamiento, etc. Está desarrollado en el nivel más alto de los protocolos y no se ocupa del manejo de las unidades de datos, su transporte, formateo o almacenamiento.
Mostraremos como se inicializa RTSP y se establecen los controles:




1. El Cliente obtiene una descripción de la presentación multimedia, la cual podría contener muchos streams
2. El browser invoca al reproductor de medios (helper application) basado en el tipo de contenido indicado en la descripción de la presentación
3. La descripción de la presentación incluye referencias a los streams usando el método URL rtsp://
4. El reproductor envía el requerimiento RTSP SETUP; el servidor envía la respuesta RTSP SETUP.
5. El reproductor envía el requerimiento RTSP PLAY; el servidor envía la respuesta RTSP PLAY.
6. El servidor de medios lanza el stream.
7. El reproductor envía el requerimiento RTSP PAUSE; el servidor envía la respuesta RTSP PAUSE.
8. El reproductor envía el requerimiento RTSP TEARDOWN; el servidor envía la respuesta RTSP TEARDOWN.

## 1.8 COMPUTACION DISTRIBUIDA EN REDES

La computación grid es una tecnología para utilizar de forma compartida y coordinada los diferentes recursos (procesamiento, almacenamiento y aplicaciones) sin estar sujetos a un control centralizado. En esta forma de computación distribuida, con recursos heterogéneos, estos se hallan conectados mediante redes de área extensa (WAN), como Internet. Desarrollado como Utility Computing, se usa desde la década de los 90s. Grid es la infraestructura hecha para la integración de computadoras de alto rendimiento, redes y bases de datos que son propiedad de diferentes organizaciones, que se asocian y establecen un software de administración para esta integración, se construye entonces un "supercomputador virtual" compuesto por una serie de computadores agrupados.

El objetivo es compartir recursos en la red de manera uniforme, segura, transparente, eficiente y fiable, ofreciendo un único punto de acceso a un conjunto de recursos distribuidos geográficamente en diferentes dominios de administración, esto se basa en la nueva generación del protocolo IP. El protocolo de Internet IPv6 permite una Internet más rápida y accesible, con este se superan las limitaciones de Internet IPv4 con nuevos niveles de servicio, para ello está en desarrollo un estándar para definir los Grid Services frente a los actuales Web Services.

Proyectos Grid son Edonkey, Emule o Limewire, que comparten datos a nivel mundial entre diferentes máquinas. Un proyecto importante es SETI@home, que tiene miles de computadoras conectadas por Internet que ceden tiempo de sus procesadores, ciclos de proceso desocupados, para analizar señales buscando patrones inteligentes extraterrestres, física de partículas, astrofísica y biología. En Europa está el CERN (Centro Europeo Investigación Nuclear) y EDG (grid de datos europea).

En cuanto a las empresas e instituciones, tenemos a Microsoft, Sun Microsystems e IBM, que han desarrollado soluciones comerciales, por ejemplo:

- Enterprise Grid Alliance (EGA) (California, 2004) por un grupo de empresas líderes en tecnología para desarrollar soluciones comerciales-empresariales de informática distribuida. Es un consorcio abierto enfocado en el desarrollo y promoción de soluciones de mallas empresariales.
- Sun Microsystems con el software Grid Engine de Sun para agregar máquinas al grid y automáticamente toma ventaja de la energía incrementada, ahorra tiempo y recursos para un grid rápido, eficiente y confiable para el manejo y despliegue.
- Andago: tecnología grid aplicada a entornos industriales y de negocio.
- JPPF: software que habilita aplicaciones con altos requerimientos de procesamiento para ser ejecutadas en varias computadoras con diferentes características (heterogéneas), basada en Java, de manera escalable y dinámica, distribuye la aplicación en tareas (jobs).

Las ventajas que ofrece la computación grid para construir la e-ciencia son:

- Capacidad de balanceo de sistemas: no se debe calcular la capacidad de los sistemas en función de los picos de trabajo, sino reasignar los recursos a donde se necesite.
- Alta disponibilidad. con la nueva funcionalidad, si un servidor falla, se reasignan los servicios a los servidores restantes.
- Reducción de costos: los servicios son gestionados por "granjas de recursos". Ya no se necesitan superservidores, estos son reemplazados por conjuntos de componentes de bajo coste. Cada sistema puede ser configurado siguiendo el mismo patrón.

En cuanto a las desventajas:

- Debe poder manejar recursos heterogéneos eficientemente
- Descubrimiento, selección, reserva, asignación, gestión y monitorización de recursos controlados externamente.
- Necesidad de desarrollar aplicaciones para manejar el grid.
- Comunicación lenta y no uniforme.
- Requiere dominios de administración, modelos de explotación y costes, políticas de seguridad
- Es costoso

### 1.8.1 Planificación

La planificación (scheduling) de tareas en un sistema distribuido de carga compartida involucra decidir no solamente cunando ejecutar un proceso, sino también dónde ejecutarlo. Para ello la planificación se lleva a cabo mediante dos componentes: el distribuidor (políticas de distribución) y el planificador (políticas de planificación).

El distribuidor decide dónde se ejecutará una tarea y el planificador dictamina cunando una tarea recibe su porción de CPU. típicamente cada nodo de un sistema distribuido tiene su propio planificador responsable de organizar los procesos en el(los) procesador(es) local(es), mientras que las decisiones de alto nivel de asignar un proceso a un nodo son tomadas por el distribuidor.

Aunque existen pequeñas variaciones, este esquema es el que surge naturalmente en un sistema distribuido. Esto se debe a dos motivos, el primero es que cada nodo tiene su sistema operativo propio encargado de planificar procesos, el segundo es el modularidad: los diseñadores pueden concentrarse mejor en los temas

relativamente complejos de la distribución de la carga sin necesidad de involucrarse con los detalles de planificación. Las políticas de distribución tratan de repartir la carga total del sistema a sus nodos individuales transfiriendo procesos entre los nodos.

Las políticas de planificación simplemente eligen un proceso del conjunto de procesos listos a ejecutarse en un nodo de forma tal de maximizar el throughput. La política de planificación de procesos de cualquier sistema operativo tradicional puede funcionar como una política de planificación del sistema distribuido. Un clúster que cumpla con las características antes mencionadas no solo proveerá una mayor capacidad de cómputo (al aprovechar CPUs ociosos) sino que además será capaz de lograr un mejor balance en la utilización de recursos, suavizando los picos de actividad. Otras posibilidades interesantes de la migración de procesos en un clúster se relacionan con un mejor aprovechamiento del espacio de almacenamiento, tolerancia a fallas, localidad de acceso a los datos, mejor administración del sistema, y computación móvil.

### 1.8.2 Arquitectura

La arquitectura grid consta de cuatro **niveles** o capas:

- Aplicación: donde se ubican todas las aplicaciones de los usuarios, plataformas y herramientas de desarrollo de esas aplicaciones. Es la capa que ve el usuario ya que proporciona el serviceware, que recoge las funciones generales de gestión como contabilidad del uso del grid, manejo de seguridad, administración general, etc.

- Middleware: proporciona herramientas para que los recursos sean usados de manera coordinada y segura dentro del grid. Son elementos típicos los agentes, brokers, metadatos, Aquí se implementan las siguientes funciones:
  - Encontrar el elemento donde se debe ejecutar la tarea solicitada por el usuario.
  - Optimizar el uso de recursos.
  - Organizar el acceso eficiente a los datos.
  - Autenticar los elementos del grid.
  - Ejecutar las tareas.
  - Monitorizar los trabajos en ejecución.
  - Gestionar la recuperación frente a fallos.
  - Informar la finalización de una tarea.


- Recursos: que son parte del grid como computadores, servidores, sistemas de almacenamiento, catálogos electrónicos de datos, bases de datos, sensores, etc.

- Red: que proporciona la conectividad dentro del grid.

Existen tres **tipos** de arquitecturas fundamentales

*a) Grid de información*

Berners-Lee y Robert Calliau con la Web crearon una infraestructura de red pública y accesible, un Grid que entrega información de cualquier tipo a cualquier lugar en el mundo, que se puede obtener al conectar cualquier ordenador a una red telefónica pública vía modem.

Servicios de compartición de archivos como Napster, Gnutella Network, E-Donkey forman parte del Grid de Información actual. A diferencia de la Web, los datos compartidos no se encuentran respaldados por una organización o dueño de algún sitio Web, sino que el servicio para compartir archivos es dispuesto por personas que desean intercambiar archivos de música, películas, videos o software. El servicio de intercambio se mantiene gracias a los participantes, no hay un repartidor central involucrado. Es un ambiente distribuido, dinámico y altamente flexible. Una de las razones de su éxito es el concepto de hipervínculo, una referencia hacia otras webs que es muy fácil de usar. El seguir los hipervínculos es comúnmente la manera más rápida para encontrar información sin tener que teclear la información. Debido a esto, la Web dominó rápidamente a ftp y a redes anteriores.

*b) Grid de Recursos*

Provee mecanismos para el uso coordinado y compartido de recursos como computadoras, bases de datos, servicios, almacenamiento y aplicaciones científicas como laboratorios, campus y otros, independientemente de su localización. La diferencia fundamental con el Grid de Información, es que usuarios anónimos no pueden acceder a estos recursos sin una credencial otorgadas por del Grid de Recursos. Solo usuarios autorizados y previamente registrados pueden ser usuarios.

Son un poco más difíciles de implementar ya que los recursos son costosos y requieren aportes económicos para mantenimiento y expansión. Los más comunes son los Grids Computacionales para acceder a superordenadores distribuidos, con tareas que consumen mucho tiempo y recursos, la mayoría trabajando bajo la aplicación Globus Toolkit. Existen prototipos de Grids en áreas específicas de investigación farmacéutica, química, astrofísica, tratamiento y representación de video, post producción, simulación del clima, geología, etc.

Además de los Grids Computacionales, están los Grids de Datos los que proveen mecanismos para almacenamiento seguro y redundante en sitios esparcidos geográficamente, manejan volúmenes de Petabytes de datos en diferentes localizaciones y deben considerar problemas como la replicación, obtención, catalogación y la coordinación.

El tercer grupo son los Grids de Aplicaciones Paralelas como el proyecto SETI@home, fightcancer@home o distributed.net, los cuales son ejecutables en ordenadores distribuidos. No necesitan Middleware adicional, porque la propia aplicación se encarga de la ejecución de los trabajos remotos y la recolección de los resultados. El aspecto más importante de estas aplicaciones es la confianza implícita en ambas partes (el dueño de la computadora confía en la integridad del software sin verificar la autenticación y la autorización, y el distribuidor del software Grid confía que los resultados no han sido falsificados por el dueño de la computadora).

Finalmente tenemos los Grids de Acceso, que son la base técnica para la colaboración remota al proveer video conferencias interactivas y facilidades Blackboard.

*c) Grid de Servicios*

Provee servicios y aplicaciones sin importar la ubicación geográfica, implementación o plataforma de hardware. Los servicios son montados en los recursos concretos disponibles en el Grid de Recursos. Una de las mayores diferencias entre estos dos tipos de Grid se encuentra en que el Grid de Servicios provee servicios abstractos sin importar su localización, mientras que el Grid de Recursos facilita accesos a recursos concretos ofrecidos en un sitio en particular. Abarca servicios como máquinas de búsqueda, portales, páginas de servidor activas y de contenido dinámico. Son gratuitos porque permiten publicidad. Servicios de email y autorización como Passport, GMX y Hotmail recaen en esta categoría.

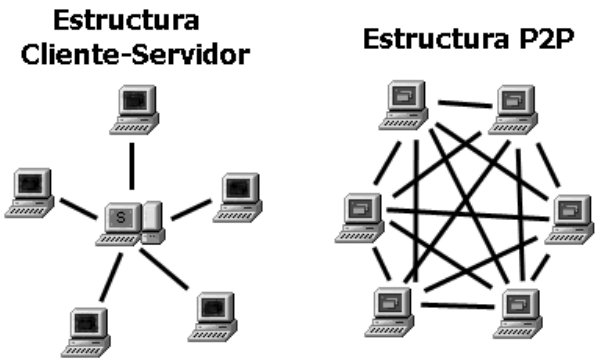
Con los webs services y el Open Grid Service Architecture OGSA están diseñados para proveer interoperabilidad entre los servicios sin importar la implementación, localización geográfica o plataforma de ejecución.

### 1.8.3 Soporte

Las instituciones trabajando sobre los estándares Grid son:

- "Globus Project" (www.globus.org) formado por múltiples instituciones de investigación y desarrollo, trabaja las tecnologías centrales para Grids computacionales. Su producto primordial es el Globus Toolkit, de arquitectura abierta, un sistema de protocolos de código libre, servicios y herramientas que permiten una computación en Grid segura y distribuida.
- Global Grid Forum (www.golbalGridforum.com) comunidad en foro de investigadores individuales y usuarios que promueven el desarrollo de las tecnologías Grid y sus aplicaciones, por consenso. Otros grupos como el New Productivity Initiative y el Peer-to- Peer Working Group han unido fuerzas con este forum para crear el mayor grupo global de estándares de cómputo en Grid.
- Data Grid Project: (Unión Europea) provee tecnología Grid necesaria en la investigación científica de próxima generación que requiere enormes cantidades de poder de procesamiento, análisis de datos y el tratamiento de millones de Gigabytes, a lo largo de diversas comunidades científicas dispersas geográficamente.

## 1.9 PARADIGMA CLIENTE/SERVIDOR.

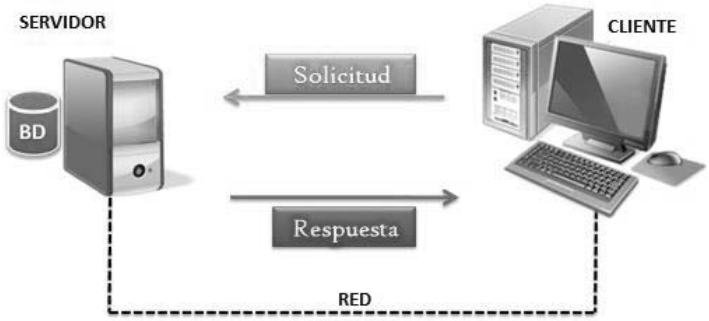



Los dos paradigmas de comunicación

La **metodología Cliente/Servidor** es un paradigma de organización de elementos de una aplicación distribuida, para que colaborando conjuntamente implementen la funcionalidad de la aplicación, proporciona un marco de referencia sencillo, flexible y abierto para distribuir la ejecución de una aplicación en múltiples nodos de una plataforma. Permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

Los *clientes* son los elementos activos que dirigen las actividades que deben ejecutarse para implementar la tarea requerida por la aplicación, solicitan a los servidores que ejecuten algunas actividades (requerimientos de servicios). Los *servidores* son los elementos pasivos especializados en realizar ciertas tareas bajo requerimientos de los clientes (respuesta al requerimiento), representan elementos que son compartidos por múltiples clientes, por una o varias aplicaciones.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor (hace una petición), y este envía uno o varios mensajes con la respuesta (provee el servicio). En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.




Comunicación Cliente/Servidor

La arquitectura cliente/servidor tiene las siguientes **características**:

- Servicios: Facilita la colaboración de procesos que se ejecutan en diferentes máquinas, a través de intercambios de servicios. Los procesos servidores proveen los servicios, los clientes los consumen.

- Recursos compartidos: Los servidores pueden ser invocados concurrentemente por los clientes, y una de sus principales funciones es arbitrar el acceso a recursos compartidos que son gestionados por el propio servidor.

- Protocolos asimétricos: Un servidor puede atender a múltiples clientes. El cliente conoce el servidor que invoca. El servidor no necesita conocer el cliente que atiende.

- Independencia de la ubicación: La ubicación de los servidores es irrelevante. Se utilizan servicios de localización definidos a nivel de plataforma para que los clientes encuentren a los de servidores.

- Compatibilidad de clientes y servidores: Los mecanismos de interacción entre clientes y servidores son independientes de las plataformas. Un middleware independiza la aplicación de la plataforma.

- Comunicación basada en intercambio de mensajes: Los clientes y servidores son elementos acoplados de forma muy libre. Interaccionan a través de intercambios de mensajes, con los se implementan las invocaciones de los servicios y las respuestas de los servicios.

- Encapsulación de los servicios: Los servicios son elementos especializados, que tienen declarados públicamente los servicios que puede servir. Sin embargo, la forma que implementa el servicio es sólo propia de él, y no puede afectar a los clientes que los requieren.

- Escalabilidad: Las aplicaciones basadas en clientes/servidores son fácilmente escalables. Hay dos tipos de escalado:
    - Escalado vertical: Los sistemas pueden crecer por un incremento del número de clientes y servidores.
    - Escalado horizontal: Los servidores pueden descomponerse en grupos de servidores que ofrezcan servicios desacoplados más específicos.

- Integridad: La información es administrada por el servidor de forma unificada, dando lugar un mantenimiento más sencillo y seguro. El middleware de distribución garantiza la seguridad en los accesos a los servicios y en la integridad de los datos.

- Clientes pesados / Servidores ligeros: La mayor parte de la funcionalidad de la aplicación se implementa en el cliente, por ejemplo, Servidores de bases de datos o servidores de ficheros.
    - Los servidores son mecanismos de acceso a recursos compartidos.
    - Mayor flexibilidad para aplicaciones que implementan nuevas funcionalidades.

- Clientes ligeros / Servidores pesados: La mayor parte de la funcionalidad se implementa en los servidores, por ejemplo, Servidores de transacciones y servidores web.
    - Incrementar la reusabilidad del código.
    - Son más fáciles de desplegar y administrar.
    - Se basan en servidores más abstractos que reducen el flujo por la red.
    - En vez de proporcionar datos, exportan procedimientos.

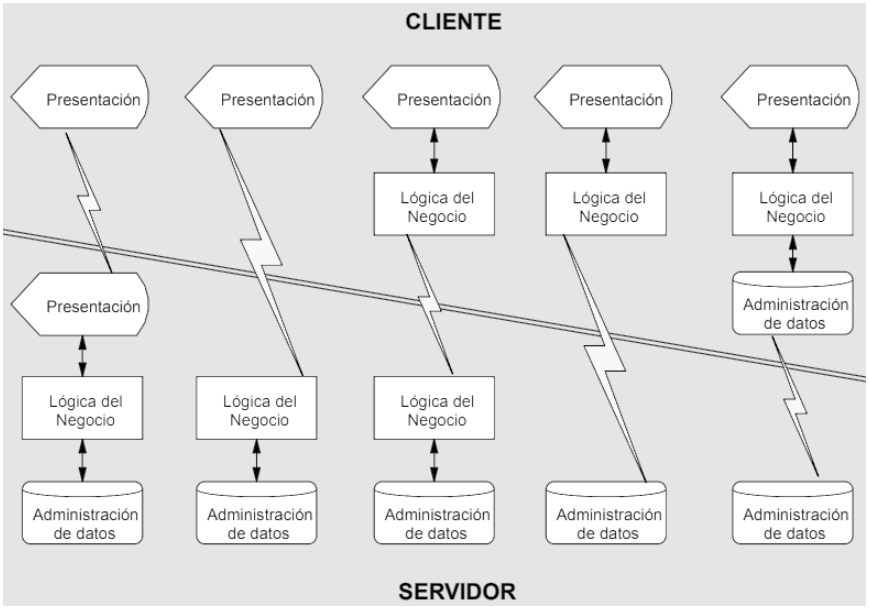
- Ambos modelos coexisten y se complementan dentro de una misma aplicación.

La arquitectura Cliente/Servidor se implementa para asumir las siguientes **tareas distribuidas**

- Presentación: software que permiten presentar en forma adecuada los resultados de una aplicación, p. ej. ventanas en Windows, páginas web en un navegador.

- Aplicación: software que entrega un resultado útil para el usuario (lógica del negocio), p. ej. Consultas, reportes, etc.

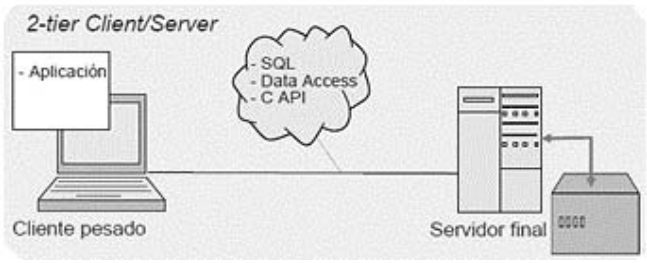
- Administración de datos: manejo de los datos (en una BD) que sirven a las aplicaciones de la lógica del negocio, p ej. datos de los productos, inventarios, etc.



Arquitecturas Cliente/Servidor

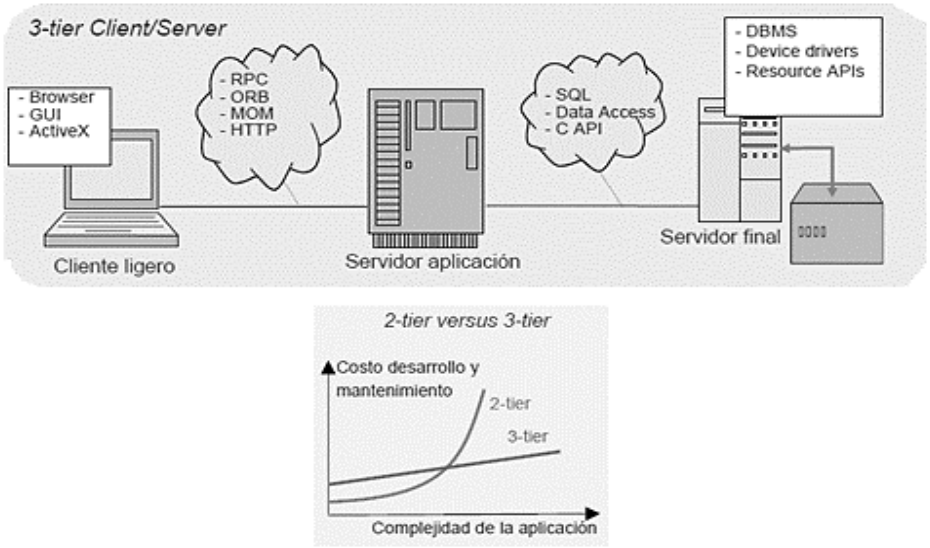
La arquitectura a implementar, depende del número de niveles o capas que se implementen

- La arquitectura en 2 niveles se utiliza para describir los sistemas cliente/servidor en donde el cliente solicita recursos y el servidor responde directamente a la solicitud, con sus propios recursos. Esto significa que el servidor no requiere otra aplicación para proporcionar parte del servicio. Al hacerse las computadoras más poderosas, la capa de presentación se mueve al cliente. Esto tiene varias ventajas:
    - Los clientes son independientes entre sí: se puede tener varias capas de presentación dependiendo de lo que cada cliente quiere hacer.
    - Se puede aprovechar la potencia de cálculo en la máquina cliente teniendo capas de presentación más sofisticadas. Esto también ahorra recursos de la máquina servidor.
    - Se introduce el concepto de API (Application Program Interface). Una interfaz para invocar el sistema desde el exterior.
    - El administrador de recursos sólo tiene un cliente: la lógica de la aplicación. Lo que ayuda con el rendimiento ya que no hay conexiones y sesiones para mantener




Arquitectura de 2 niveles o capas

- En la arquitectura en 3 niveles, existe un nivel intermediario. en la arquitectura encontramos:
  - Un cliente: equipo que solicita los recursos, equipado con una interfaz de usuario (generalmente un navegador Web) para la presentación.
  - El servidor de aplicaciones (también denominado software intermedio), cuya tarea es proporcionar los recursos solicitados, pero que requiere de otro servidor para hacerlo. Algunos interpretan los middlewares como esta capa intermedia
  - El servidor de datos, que proporciona al servidor de aplicaciones los datos que requiere



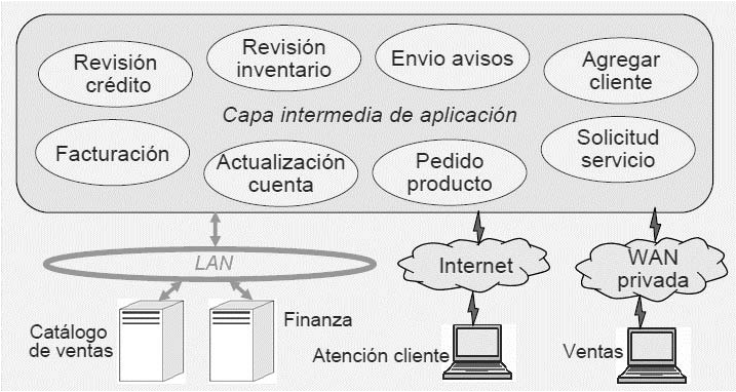

Arquitectura de 3 niveles o capas

La existencia del servidor de aplicaciones, proporciona las siguientes ventajas:
  - simplifica el diseño de los clientes mediante la reducción del número de interfaces
  - proporciona un acceso transparente a los sistemas subyacentes
  - actúa como plataforma para la funcionalidad inter-sistema y la lógica de aplicación de alto nivel
  - se encarga de localizar los recursos, el acceso a ellos, y la recolección de resultados.

- En la arquitectura en n niveles, los diferentes procesos están distribuidos en diferentes capas no sólo lógicas, sino también físicas. los procesos se ejecutan en diferentes equipos, que pueden incluso residir en plataformas o sistemas operativos completamente distintos. cada equipo posee una configuración distinta y está optimizado para realizar el papel que le ha sido asignado dentro de la estructura de la aplicación, de modo que tanto los recursos como la eficiencia global del sistema se optimicen. El surgimiento de la tecnología de componentes distribuidos es la clave de las arquitecturas de n-capas. estos sistemas de computación utilizan un número variable de componentes individuales que se comunican entre ellos utilizando estándares predefinidos y frameworks de comunicación como:

  - corba: (common object request broker architecture) del object management group (omg).
  - dna : (distributed network applications) de microsoft (incluye com/dcom y com+ además de mts, msmq, etc.
  - ejb : (enterprise java beans) de sun microsystems

- xml : (extensible markup language) del world wide web consortium (w3
- .net: de microsoft que incluye nuevos lenguajes como visual basic.net, c#.




Arquitectura de n capas

El aumentar el número de niveles permite:

- Los proyectos grandes se pueden desarrollar como conjunto de pequeños proyectos.
- Se pueden reutilizar componentes entre aplicaciones. Las aplicaciones se construyen agrupando componentes de forma específica a la funcionalidad que requiere.
- Permite elevar el nivel de abstracción. Los clientes pueden requerir los servicios por sus nombres y no necesitan conocer de qué base de datos proceden, ni que elementos participan para implementarlos.
- Permiten incorporar fácilmente elementos legados.
- Los entornos de componentes no envejecen sino mejoran: nuevos clientes pueden generarse añadiendo algún objeto servidor, los objetos servidores pueden modificarse o sustituirse sin que afecten a los clientes.

### 1.9.1 Clientes

Constituyen el núcleo de una aplicación, aunque puede ser sólo una parte pequeña de ella, los clientes tienen en común que elaboran su funcionalidad solicitando servicio a los servidores. Los clientes se diferencian en con qué desencadenan las invocaciones:

- Clientes que no necesitan GUI ni concurrencia: cajeros automáticos, puntos de venta, lectores de código de barras, teléfonos móviles, faxes, etc.
- Clientes que no necesitan GUI pero si concurrencia: robots, máquinas herramientas, demonios, etc.
- Clientes que necesitan GUI: la evolución del programa y los servicios que requiere son resultado de la interacción del operador con la interfaz gráfica que se le ofrece. Utilizan el modelo objeto/acción.
- Clientes con OOUI (Object Oriented user Interface): la evolución del programa y los servicios que utiliza son resultado de la interacción del operador con iconos mostrados en la consola y que una vez abiertos pueden ser manipulados concurrentemente y en cualquier orden.

### 1.9.2 Servidor

Su función es satisfacer los requerimientos de servicio que son realizados por múltiples clientes que quieren usar de la funcionalidad que ofrece o acceder a los recursos protegidos que gestiona. Su funcionalidad suele ser compleja y pesada, pero su actividad es monótona y poco creativa:

- Espera pasivamente los clientes inicien requerimientos de servicios.
- Ejecuta concurrentemente los requerimientos recibidos de los clientes.

- Prioriza las respuestas a los clientes de acuerdo con su importancia o urgencia.
- Arranca y ejecuta demonios y actividades de segundo plano (background)
- Se mantienen continuamente en ejecución, y si cae por un fallo se recupera.
- En función de la actividad que en cada momento desarrolla puede acaparar grandes cantidades de capacidad de procesamiento, recursos y anchura de banda de comunicaciones.

El sistema operativo que corre en el servidor deberá manejar los siguientes *servicios básicos*:

- Gestionar la concurrencia basada en threads
- Establecer políticas flexibles de planificación de los threads.
- Establecer mecanismos de sincronización que garantice el acceso seguro a los recursos compartidos.
- Establecer mecanismos eficientes de comunicación entre procesos.
- Administrar de manera flexible la memoria.
- Manejar un sistema de ficheros multiusuarios

El sistema operativo que corre en el servidor deberá manejar los siguientes *servicios extendidos*:

- Facilidades para acceder a través de la red a sistemas con diferentes protocolos.
- Facilidades a información compartida.
- Acceso a diferentes middlewares.
- Streaming y mecanismos de transferencia de grandes bloques de información.
- Directorios globales y servicios de páginas amarillas, que permitan localizar los servidores y los servicios que ofrecen.
- Servicios de identificación, autentificación y encriptación.
- Servicio de tiempos y hora global.
- Servicios de bases de datos
- Servicios de transacciones.
- Brocker para la gestión de objetos y componentes.

## 1.10 PARADIGMA PEER-TO-PEER

Una red peer-to-peer es una red entre pares o red punto a punto, todos o algunos aspectos funcionan sin clientes ni servidores fijos, sino una serie de nodos que se comportan como iguales entre sí, eso quiere decir que actúan simultáneamente como clientes y servidores. Estas redes permiten el intercambio directo de información, en cualquier formato, entre los ordenadores interconectados, por lo que estas redes administran y optimizan el uso del ancho de banda de los demás usuarios de la red por medio de la conectividad entre los mismos. Un par entonces, es una entidad capaz de desarrollar algún trabajo útil y de comunicar los resultados de ese trabajo a otra entidad de la red, ya sea directa o indirectamente.

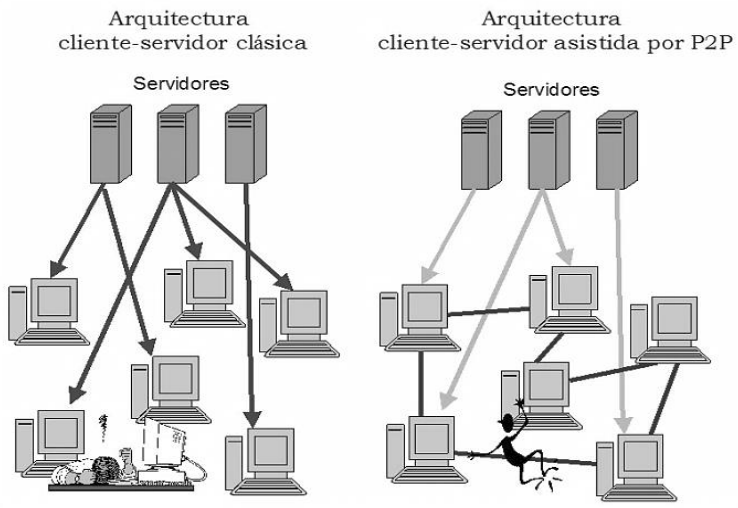
Su funcionalidad hace que se utilicen para compartir información entre dos o más usuarios, por lo que se usan para intercambiar archivos bajo leyes de copyright, compartir archivos de diferentes tipos (audio, vídeo o software), telefonía VoIP, telefonía sobre internet (Skype).

Las redes P2P, tienen las siguientes **características**:

- Escalabilidad: cuantos más nodos estén conectados a una red P2P, mejor será su funcionamiento. Así, cuando los nodos llegan y comparten sus propios recursos, los recursos totales del sistema aumentan.
- Robustez: si hay fallos en la réplica excesiva de los datos hacia múltiples destinos, permitiendo a los peers encontrar la información sin hacer peticiones a ningún servidor centralizado de indexado.

- Descentralización: todos los nodos son iguales, no tienen funciones especiales y ninguno es imprescindible (aunque no todas cumplen la característica, como Napster, eDonkey o BitTorrent).
- Distribución de costes entre los usuarios: se donan recursos a cambio de recursos (archivos, ancho de banda, ciclos de proceso o almacenamiento de disco).
- Anonimato: del autor de un contenido, el editor, el lector, el servidor que lo alberga y la petición para encontrarlo.
- Seguridad: hay que identificar y evitar nodos maliciosos, contenido infectado, espionaje de las comunicaciones, creación de grupos seguros de nodos dentro de la red y protección de los recursos de la red.

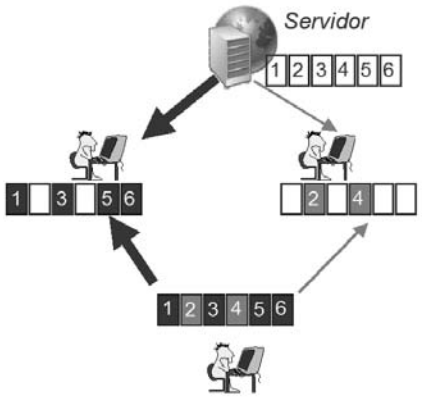
La arquitectura para la distribución de contenidos empleada por los primeros programas P2P era una arquitectura cliente-servidor, pero en la que también colaboran los clientes o pares, se ayudan entre sí, convirtiéndose en servidores de otros clientes. Estas redes pueden crecer indefinidamente sin incrementar el tiempo de las búsquedas y sin necesidad de costosos recursos centralizados; con relación al ancho de banda (bits/seg), capacidad de almacenamiento y de procesamiento, el número de dichos recursos siempre aumenta en proporción directa al crecimiento de la red. Debido a que los ordenadores individuales no tienen una IP fija, sino una asignación dinámica al conectarse a la red, no pueden comunicarse entre sí porque no saben las direcciones que deben usar, entonces se conectan a un servidor con dirección IP fija, que mantiene la relación de direcciones IP de los clientes y servidores. Con ello, los clientes disponen ya de información sobre el resto de la red y pueden intercambiar información o recursos entre sí, sin intervención de los servidores.




Arquitecturas de Redes

Para descargar un archivo usando redes P2P, estos ficheros deben ser segmentados en pequeñas partes para su distribución, los clientes van solicitando diferentes segmentos al servidor o a otros clientes e inician la descarga de forma concurrente para conseguir una mayor velocidad. Luego esos mismos clientes se convierten en servidores de los segmentos que están descargando o que acaban de descargar. Como el mismo archivo se descarga desde múltiples fuentes al mismo tiempo, la velocidad es mucho mayor cuando crece el número de usuarios que lo comparten. Dado que el ancho de banda y el número de conexiones establecidas es limitado en cada servidor, es necesario establecer un sistema de colas que permita que todos los clientes descarguen distintos segmentos de forma equitativa. Una vez que el cliente disponga de todos los segmentos descargados, reconstruirá el archivo total. En ese momento, el usuario podrá reproducir el vídeo o la canción descargada. Este sistema se utiliza mucho en archivos de audio y video.

El procedimiento empleado cuando se comparten recursos computacionales a través de redes P2P es el mismo, ahora serán las tareas las que se segmentan teniendo en cuenta sus posibilidades de paralelización. Una vez divididas en partes independientes, estas tareas se envían a distintos pares para su solución.



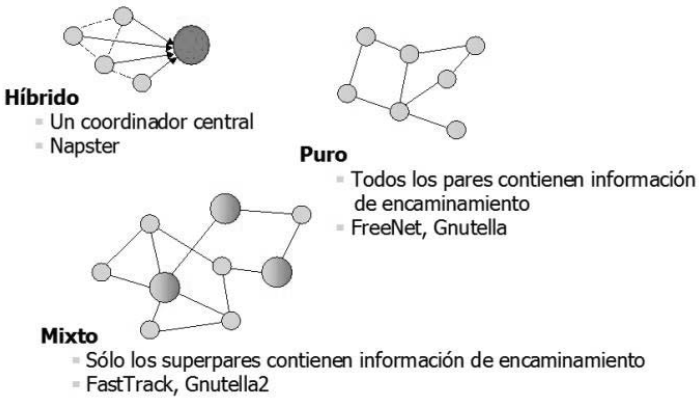

Funcionamiento de una red P2P

### 1.10.1 Arquitecturas P2P

Los sistemas computacionales pueden clasificarse en centralizados o distribuidos. Los sistemas distribuidos pueden, a su vez, trabajar según el modelo cliente-servidor o el modelo P2P.

Como hemos visto las arquitecturas cliente/servidor están implementados en un modelo de capas, que vá desde el monocapa o plano, cuando varios clientes dispersos se comunican con un único servidor, que podría estar replicado para mejorar la robustez y fiabilidad; hasta el multicapa o jerárquico, para mejorar la escalabilidad y el rendimiento, en este caso, los servidores de un nivel actúan como clientes de los servidores de un nivel superior. Como ejemplo de modelos planos pueden citarse las soluciones middleware convencionales como CORBA (*Common Object Request Broker Architecture*), DNS (*Domain Name Service*) y Usenet.

La arquitectura P2P puede ser pura o híbrida, en un modelo puro no existe ningún servidor centralizado (por ejemplo, FreeNet o Gnutella), en un modelo híbrido, antes de que un cliente contacte con otros clientes de la red se accede a un servidor para obtener cierta información, como puede ser la identidad de otro par en el que se encuentra cierta información, o para verificar las credenciales de seguridad (por ejemplo, Napster, Groove o iMesh). Existe también un modelo mixto, como el empleado por Kazaa o BitTorrent, donde se manejan superpares que pueden contener información de la que carecen otros pares normales, de esta forma, cuando los pares no encuentran cierto tipo de información a partir de otros, contactan con los superpares.



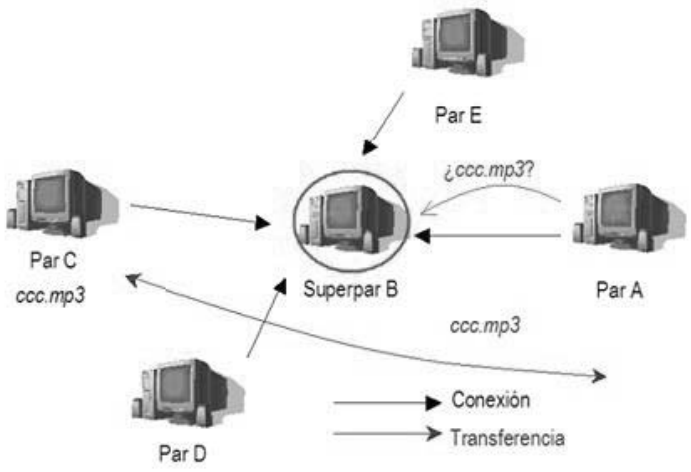

Arquitecturas P2P

- *Modelo P2P híbrido*
  La primera generación de P2P (Napster) empleaba una **estructura de red cliente-servidor**. El servidor central mantiene una base de datos con información de los ficheros almacenados por cada par. Cada vez que un cliente se conecta o desconecta de la red, se actualiza la base de datos. En este modelo, todos los mensajes de búsqueda y control son enviados al servidor centralizado. El servidor centralizado compara la solicitud de sus clientes con el contenido de su base de datos y envía las correspondencias al cliente en cuestión. Una vez informado de las correspondencias, el cliente contacta con el par directamente y accede

al recurso solicitado. Los contenidos nunca se almacenan en el servidor central. La arquitectura P2P centralizada proporciona un rendimiento muy elevado cuando se trata de localizar recursos. Todos los pares de la red deben registrarse, lo cual asegura que todas las búsquedas van a ser cursadas y ejecutadas rápida y eficientemente siempre y cuando el servidor esté bien dimensionado. Sin embargo, es un sistema muy costoso.

- *Modelo P2P puro*
  La segunda generación de P2P (FreeNet o Gnutella) usa un modelo distribuido donde **no existe ningún servidor central y todos los nodos son iguales**. Cada nodo actúa como servidor y como cliente en la red. Cada par dentro de esta arquitectura trata de mantener un cierto número de conexiones con otros pares todo el tiempo. Este conjunto de pares conectados transporta el tráfico de red, conformado esencialmente por peticiones y respuestas a esas peticiones y mensajes de control que facilitan el descubrimiento de otros nodos. El modelo P2P puro no reside en un servidor centralizado y es mucho más robusto y económico que el P2P híbrido. La principal desventaja reside en el elevado tiempo y la sobrecarga de ancho de banda que suponen las búsquedas de información en la red. Una solicitud puede tener que viajar a través de varios centenares de usuarios antes de que se consiga el resultado.

- *Modelo P2P mixto*
  En la tercera generación de P2P (FastTrack, Kazaa, Grokster, o Groove; y Gnutella2, empleado por las aplicaciones: BearShare, LimeWire, Gnucleus, Shareaza o Morpheus) se emplea una arquitectura P2P mixta. Dentro de este modelo, ciertos pares de la red se seleccionan como superpares y ayudan a gestionar el tráfico dirigido hacia otros pares. Los superpares cambian dinámicamente a medida que lo hacen el ancho de banda de la red y la topología. Cada nodo cliente mantiene sólo un pequeño número de conexiones abiertas y cada una de esas conexiones es un superpar. Esta nueva topología virtual tiene el efecto de hacer la red escalable, mediante la reducción del número de nodos involucrados en el encaminamiento y manejo de los mensajes de búsqueda y control, así como la disminución del volumen de tráfico entre ellos. Debido a los superpares, que también actúan como concentradores de búsquedas, la velocidad de respuesta a las solicitudes dentro de este entorno P2P descentralizado es comparable a la de un entorno P2P centralizado, aunque más escalable y tolerante a fallos.




Modelo P2P mixto asistido por superpares

### 1.10.2 Propiedades de P2P

Son propiedades de las arquitecturas P2P:

- *Descentralización*
  En el modelo cliente-servidor tradicional, la información está en servidores y a ella acceden los usuarios mediante programas clientes, que básicamente se comportan como interfaces de usuario. Estos sistemas centralizados son ideales para algunas aplicaciones. Por ejemplo, el control de derechos de acceso y las búsquedas de recursos se gestionan mucho más fácilmente en sistemas centralizados. Sin embargo, la

topología de los sistemas centralizados lleva a ineficiencias, cuellos de botella y recursos desperdiciados, sin contar lo costoso que resulta establecer los registros centralizados y mantenerlos con información relevante y actualizada.

- *Escalabilidad*
  La escalabilidad está limitada por factores tales como la cantidad de operaciones centralizadas (sincronización y coordinación) que deben ejecutarse, la cantidad de estados que han de mantenerse, el paralelismo inherente a una aplicación y el modelo de programación que se emplea en el desarrollo de la aplicación. La escalabilidad siempre debe estar en equilibrio con otras características deseables como el determinismo y el rendimiento.

- *Anonimato*
  El anonimato permite a los usuarios emplear un sistema sin preocuparse de ataques a su privacidad. Para ello, es necesario proteger el anonimato tanto de la identidad del emisor y receptor de los mensajes como del autor, distribuidor y lector de los contenidos y del lugar donde se almacenan. Existen varias técnicas para alcanzar el anonimato en las redes P2P, como la creación de grupos de multicasting para que el receptor de un mensaje o contenido no pueda ser identificado; ocultación de la dirección IP y la identidad del emisor; comunicación mediante nodos intermedios a pesar de que sea factible contactar directamente con el destinatario, o ubicación involuntaria, encriptada y fragmentada de los contenidos.

- *Propiedad compartida*
  La propiedad compartida reduce el coste de la posesión de los sistemas y contenidos, así como el coste de su mantenimiento. El coste del sistema global se ve limitado, porque en P2P se reducen las capacidades de cálculo, almacenamiento y ancho de banda ociosas.

- *Conectividad ad hoc*
  Las aplicaciones están preparadas para nodos que no están disponibles durante todo el tiempo o que lo hacen de forma intermitente, para ello se toman medidas como, por ejemplo, proveedores de contenidos redundantes con técnicas de replicación espontáneas o nodos centralizados que se encargan de mantener la información o mensajes destinados a pares temporalmente desconectados de la red.

- *Rendimiento*
  Los sistemas P2P buscan agregar anchos de banda, capacidad de almacenamiento y ciclos de computación de los dispositivos diseminados por una red. Los sistemas descentralizados consiguen un mayor rendimiento, salvo para ciertas funciones, típicamente la búsqueda de recursos en la red. Por ello, las aplicaciones P2P actuales suelen tener una arquitectura mixta, incorporando el concepto de superpares, o pares en los que otros pares delegan las búsquedas de recursos.

- *Seguridad*
  Los sistemas P2P comparten la mayoría de los requisitos de seguridad con los sistemas distribuidos tradicionales, como el establecimiento de relaciones de confianza entre los nodos y objetos distribuidos y de esquemas de intercambio de claves de sesión. No obstante, en los sistemas P2P aparecen nuevos requisitos: encriptación en las comunicaciones y almacenamiento de datos, sandboxing, gestión de derechos digitales, reputación e interoperabilidad con cortafuegos y NAT.

- *Tolerancia a fallos*
  Los sistemas Cliente/Servidor suponen una pérdida total del sistema cuando el servidor falla, lo que no ocurre en P2P. Es decir, un objetivo de diseño de una red P2P es que ésta no pierda su funcionalidad por fallos asociados a desconexiones de nodos o a nodos no alcanzables, caídas de la red y fallos de nodos. Por ello, es siempre deseable encontrar mecanismos que permitan manejar este tipo de fallos, bastante frecuentes sobre todo en las redes inalámbricas, por ello se considera la existencia de que varios nodos sean capaces de ofrecer los mismos recursos y servicios, para lo cual, habitualmente, se replican espontáneamente.

- *Interoperabilidad*
  A pesar de que existen ya muchos sistemas P2P, la mayoría de ellos no son interoperables, es decir, aunque comparten un mismo propósito no son capaces de comunicarse y entenderse entre sí. La razón de esta falta de interoperabilidad ha sido la inexistencia de una entidad de estandarización mundial dedicada al mundo del P2P. No obstante, recientemente se han obtenido mejoras significativas a este respecto, con la formación del Grupo de Trabajo P2P de Internet2 y la aparición de JXTA, que son un intento de reunir los esfuerzos de los desarrolladores de aplicaciones P2P y de establecer un grupo común para escribir reportes y especificaciones que hagan posible el entendimiento común entre todas ellas. El Proyecto JXTA es un importante esfuerzo para conseguir la interoperabilidad ofreciendo una infraestructura de código abierto para el desarrollo de aplicaciones P2P, con el objetivo de convertir esta arquitectura básica o middleware en un estándar.