MODELO CLIENTE SERVIDOR DE 3 CAPAS

El sistema es escalable, porque es relativamente fácil añadir nuevos servidores web, a medida que el número de clientes crece.

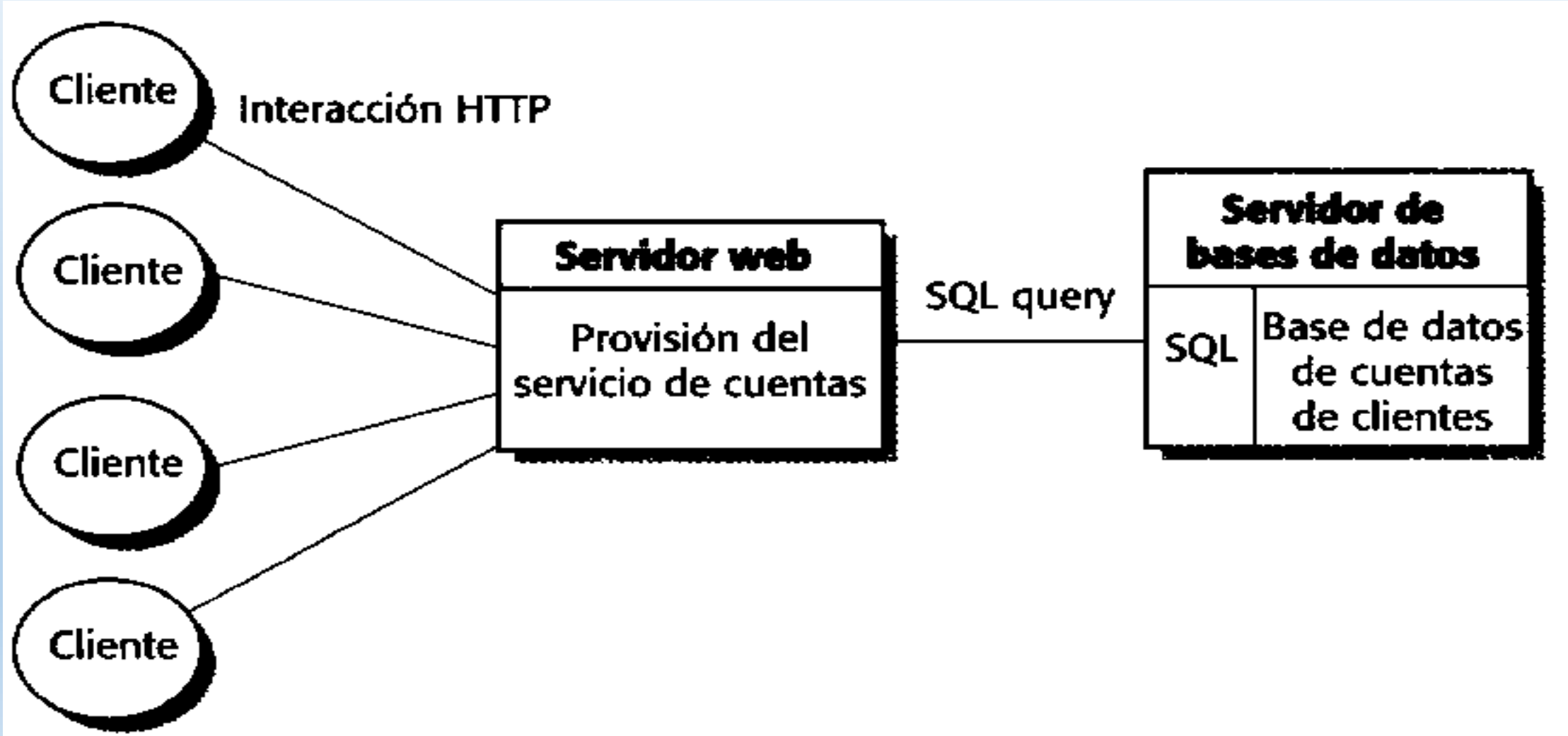
* El uso de una arquitectura de tres capas permite optimizar la transferencia de información entre el servidor web y el servidor de la base de datos.

* Las comunicaciones entre estos sistemas pueden usar protocolos de comunicación de bajo nivel muy rápidos.

* Para recuperar información de la base de datos se utiliza un middleware eficiente que soporte consultas a la base de datos en SQL (Structured Query Language).

MODELO CLIENTE SERVIDOR DE 3 CAPAS

Figura de Modelo de Arquitectura Cliente – Servidor de 3 Capas (Sistema Bancario en Internet)



Modelo Cliente - Servidor

Resumen del uso de diferentes arquitecturas Cliente-Servidor

Arquitectura C/S de dos capas con clientes ligeros	Aplicaciones de sistemas heredados en donde no es práctico separar el procesamiento de l aplicación y la gestión de los datos. Aplicaciones que requieren cálculos intensivos tales como compiladores con poca o ninguna gestión de los datos. Aplicaciones que requieran manejar una gran cantidad de datos (navegar y consultar) con poco o ningún procesamiento de la aplicación.
Arquitectura C/S de dos capas con clientes ricos	Aplicaciones en donde el procesamiento de la aplicación se proporciona por software comercial (por ejemplo, Microsoft Excel) sobre el cliente. Aplicaciones que requieren un procesamiento de datos computacionalmente intensivo (por ejemplo, visualización de datos). Aplicaciones con una funcionalidad para el usuario final relativamente estable usada en un entorno de gestión del sistema bien establecido.
Arquitectura C/S de tres capas o multicapa	Aplicaciones a gran escala con cientos o miles de clientes. Aplicaciones en donde tanto los datos como la aplicación son volátiles. Aplicaciones en donde se integran datos de múltiples fuentes.

MODELO CLIENTE - SERVIDOR

A veces sirve extender el modelo cliente-servidor de tres capas a una variante multicapa en la que se **añaden al sistema servidores adicionales**.

* Los sistemas multicapa pueden usarse cuando las aplicaciones necesitan acceder y usar datos de diferentes bases de datos.

* Entonces, un servidor de integración se ubica entre el servidor de aplicaciones y los servidores de la base de datos.

* El servidor de integración recoge los datos distribuidos y los presenta a la aplicación como si se obtuviesen desde una única base de datos.

MODELO CLIENTE - SERVIDOR

Las arquitecturas cliente-servidor de tres capas y las variantes multicapa, que distribuyen el procesamiento de la aplicación entre varios servidores, son más escalables que las arquitecturas de dos capas.

* El tráfico de la red se reduce, al contrario que con las arquitecturas de dos capas de cliente ligero.

* El procesamiento de la aplicación es la parte más volátil del sistema, y puede ser fácilmente actualizada, porque está localizada centralmente.

* El procesamiento, en algunos casos, puede distribuirse entre: la lógica de la aplicación y los servidores de gestión de datos, lo que permite una respuesta más rápida a las peticiones de los clientes.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

En el modelo cliente-servidor de un sistema distribuido, los clientes y los servidores son diferentes.

* Los clientes reciben servicios de los servidores y no de otros clientes; los servidores pueden actuar como clientes recibiendo servicios de otros servidores, pero sin solicitar servicios de clientes.

* Los clientes deben conocer los servicios que ofrece cada uno de los servidores y deben conocer cómo contactar con cada uno de ellos.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

El modelo **Cliente – Servidor** funciona bien para muchos tipos de aplicaciones.

* Sin embargo, limita la flexibilidad del diseñador, que debe decidir dónde se proporciona cada servicio.

* También debe planificar la escalabilidad y proporcionar algún medio para distribuir la carga sobre los servidores, cuando más clientes se añadan al sistema.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

Una opción superadora es eliminar la distinción entre cliente y servidor y diseñar una **arquitectura de objetos distribuidos**.

* Aquí, los componentes del sistema son **objetos que proporcionan y requieren un conjunto de servicios**.

* Otros objetos realizan llamadas a estos servicios sin hacer ninguna distinción lógica entre un **cliente** (el **receptor de un servicio**) y un **servidor** (el **proveedor de un servicio**).

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

Una opción superadora es eliminar la distinción entre cliente y servidor y diseñar una **arquitectura de objetos distribuidos**.

* Aquí, los componentes del sistema son **objetos que proporcionan y requieren un conjunto de servicios**.

* Otros objetos realizan llamadas a estos servicios sin hacer ninguna distinción lógica entre un **cliente** (el **receptor de un servicio**) y un **servidor** (el **proveedor de un servicio**).

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

Los objetos pueden distribuirse a través de varias computadoras en una red y comunicarse a través de middleware.

* A este middleware se lo denomina intermediario de peticiones de objetos.

* Su misión es proporcionar una interfaz transparente entre los objetos.

* Proporciona un conjunto de servicios que permiten la comunicación entre los objetos y que éstos sean añadidos y eliminados del sistema.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

Ventajas del modelo de objetos distribuido:

1) Permite al diseñador retrasar decisiones sobre dónde y cómo deberían proporcionarse los servicios.

* Los objetos que proporcionan servicios pueden ejecutarse sobre cualquier nodo de la red.

* Por lo tanto, la distinción entre los modelos de cliente rico y ligero es irrelevante, ya que no hay necesidad de decidir con antelación dónde ubicamos la lógica de aplicación de los objetos.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

Es una arquitectura abierta: permite añadir nuevos recursos si es necesario.

*Se han desarrollado estándares de comunicación de objetos, que permiten escribir objetos, en diferentes lenguajes de programación para comunicarse y proporcionarse servicios entre ellos.

3) El sistema es flexible y escalable.

*Pueden añadirse nuevos objetos, a medida que la carga del sistema se incrementa, sin afectar al resto de los objetos del sistema.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

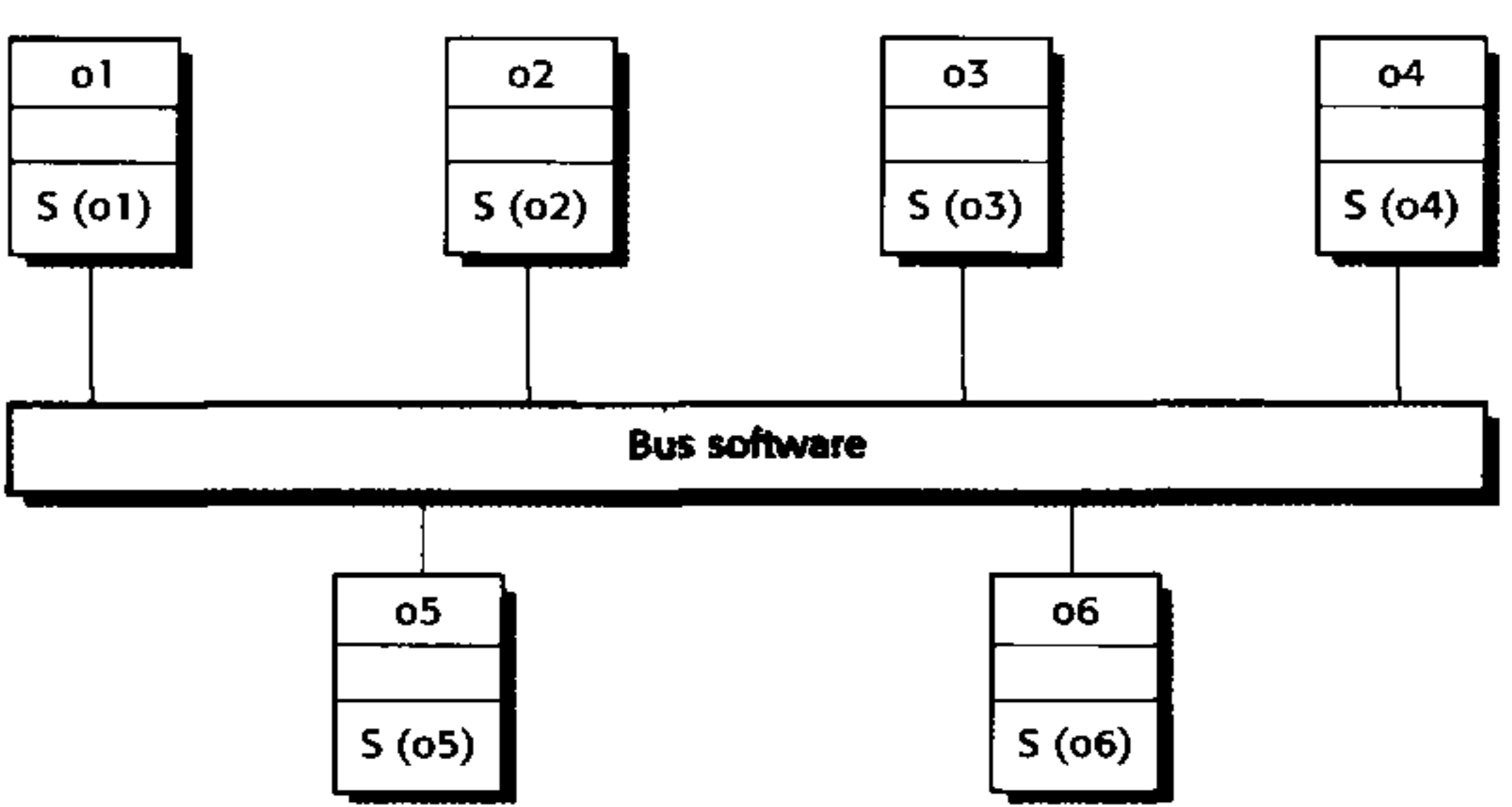
4) Si es necesario, se puede reconfigurar el sistema, de forma dinámica, mediante la migración de objetos a través de la red.

*Esto importa cuando haya fluctuación en los patrones de demanda de servicios.

*Un objeto que proporciona servicios puede migrar al mismo procesador que los objetos que demandan los servicios, lo que mejora el rendimiento del sistema.

ARQUITECTURAS DE OBJETOS DISTRIBUIDOS

Arquitectura de Objetos Distribuidos



ARQUITECTURA DE OBJETOS DISTRIBUIDOS

Una arquitectura de objetos distribuidos puede ser usada como un modelo lógico que permita estructurar el sistema.

* Entonces, debemos pensar cómo proporcionar las funcionalidades de la aplicación únicamente en términos de servicios y combinaciones de servicios.

* Después, debemos identificar cómo proporcionar estos servicios utilizando varios objetos distribuidos.

ARQUITECTURA DE OBJETOS DISTRIBUIDOS

La principal desventaja de las arquitecturas de objetos distribuidos es que son mucho más complejas de diseñar que los sistemas cliente-servidor.

* Los sistemas cliente-servidor parecen ser la forma más natural de concebir a los sistemas.

* Estos reflejan muchas transacciones humanas, en las que la gente solicita y recibe servicios de otra gente especializada en proporcionárselos.

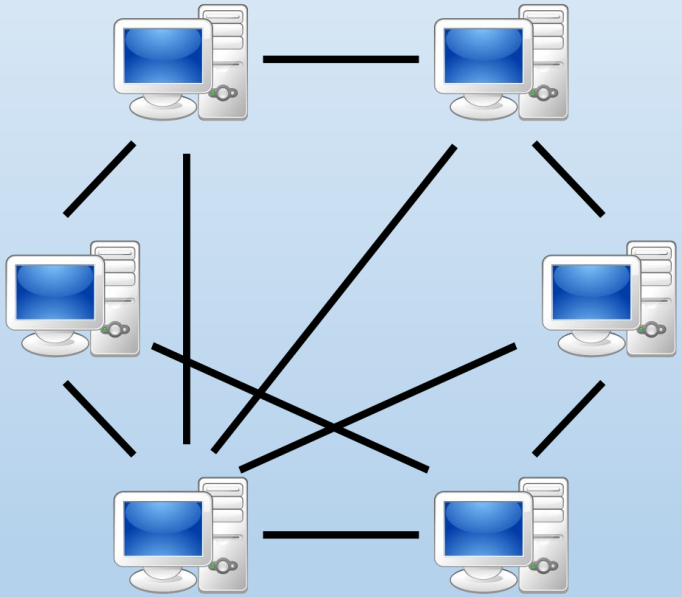
* Es más difícil pensar en una provisión de servicios generales.

ARQUITECTURA PEER - TO - PEER

Los sistemas peer-to-peer (p2p) son sistemas descentralizados, en los que los cálculos pueden llevarse a cabo en cualquier nodo de la red y, al menos **en principio**, **no se hacen distinciones entre clientes y servidores**.

* En las aplicaciones peer-to-peer, el sistema en su totalidad se diseña para aprovechar la ventaja de la potencia computacional y disponibilidad de almacenamiento a través de una red de computadoras potencialmente enorme.

ARQUITECTURA PEER - TO - PEER



ARQUITECTURA PEER - TO - PEER

Los estándares y protocolos que posibilitan las comunicaciones, a través de los nodos, están embebidos en la propia aplicación.

* Cada nodo debe ejecutar una copia de dicha aplicación.

* Las tecnologías peer-to-peer han sido mayormente usadas para sistemas personales.

* Sin embargo, se está utilizando de forma creciente en empresas con potentes redes de PCs.

ARQUITECTURA PEER - TO - PEER

Intel y Boeing han implementado sistemas p2p para aplicaciones que requieren computaciones intensivas.

* Para aplicaciones cooperativas que soportan trabajo distribuido, es la tecnología más efectiva.

* Hay dos tipos principales de arquitecturas lógicas de red que se pueden usar: arquitecturas descentralizadas y arquitecturas semicentralizadas.

ARQUITECTURA PEER - TO - PEER

En teoría, en los sistemas peer-to-peer, cada nodo podría conocer cualquier otro nodo, conectarse con él, e intercambiar datos.

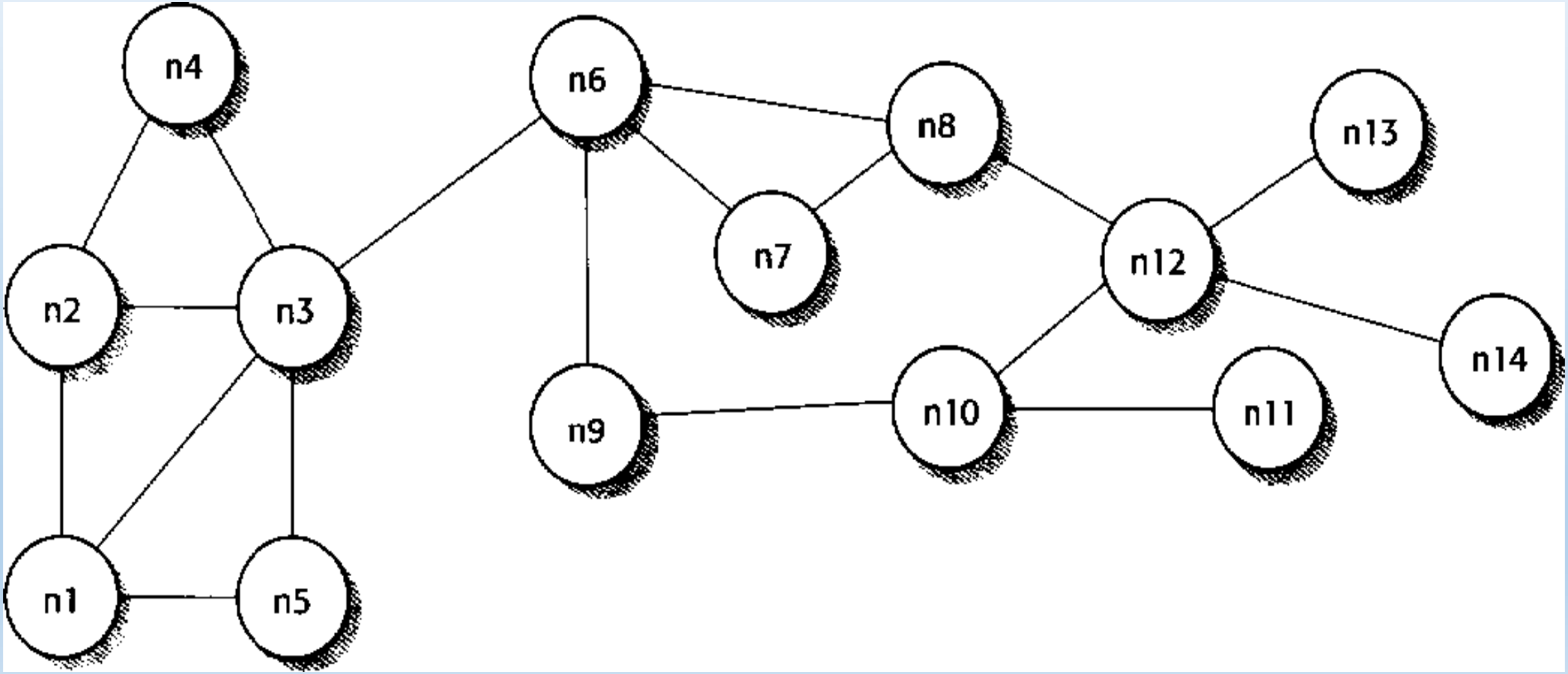
* En la práctica, esto es imposible, ya que los nodos se organizan dentro de «localidades» con algunos nodos que actúan como puentes a otras localidades de nodos.

* En una arquitectura descentralizada, los nodos en la red no son simplemente elementos funcionales, sino que también son interruptores de comunicaciones que pueden encaminar los datos y señales de control de un nodo a otro.

* En la figura siguiente, si **n1** debe buscar un archivo que está almacenado en **n10**, esta búsqueda se encamina a través de los nodos **n3**, **n6** y **n9** hasta llegar a **n10**.

ARQUITECTURA PEER - TO - PEER

Arquitectura P2P descentralizada



ARQUITECTURA PEER - TO - PEER

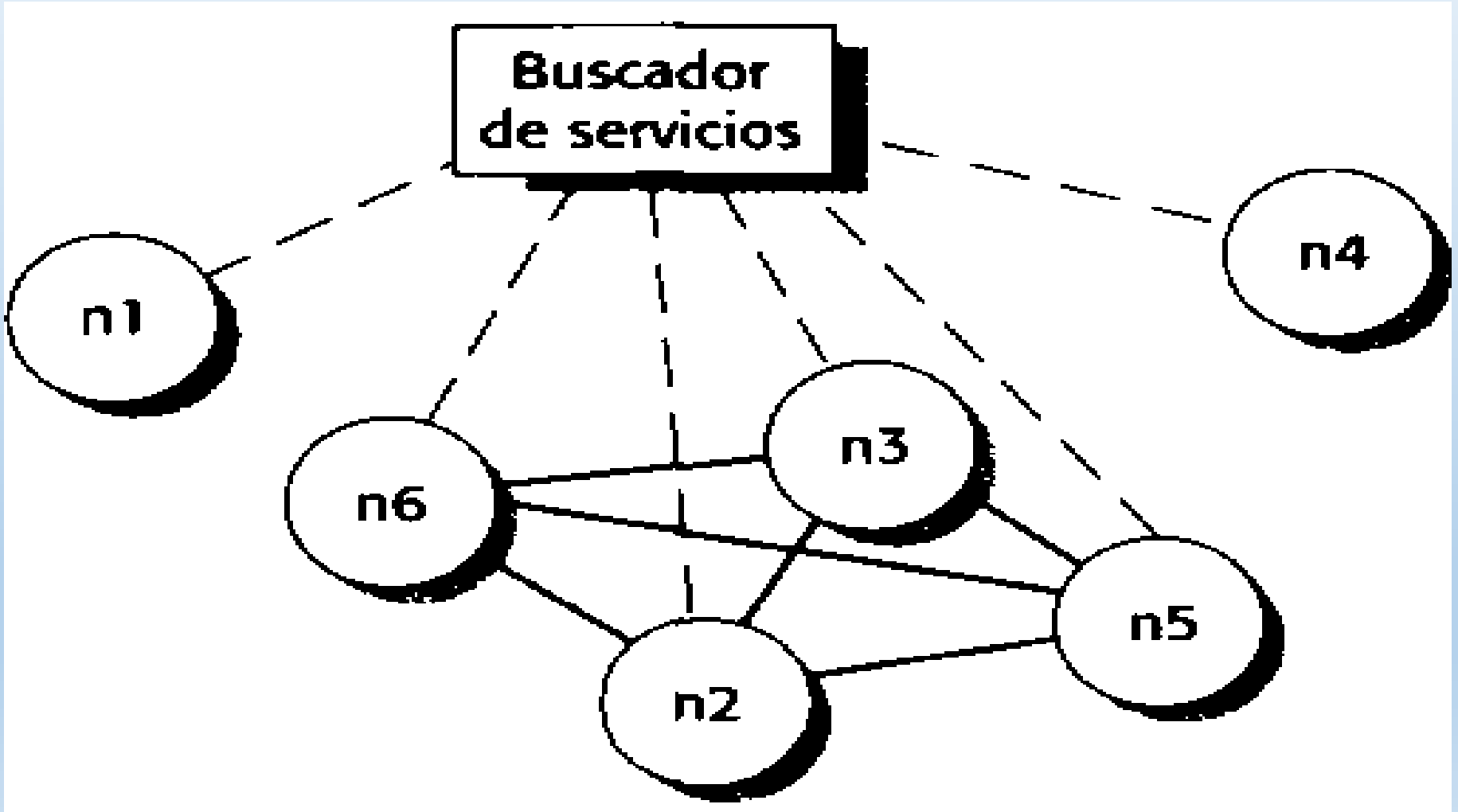
Esta arquitectura descentralizada tiene ventajas: es altamente redundante, y tolerante a defectos y a nodos desconectados de la red.

* Sin embargo, existen sobrecargas obvias en el sistema ya que la misma búsqueda puede ser procesada por muchos nodos diferentes y hay una sobrecarga significativa en comunicaciones.

* Un **modelo de arquitectura p2p alternativo** que parte de una **arquitectura p2p pura** es una **arquitectura semi-centralizada** en la que, **dentro de la red, uno o más nodos actúan como servidores para facilitar las comunicaciones entre los nodos**.

ARQUITECTURA PEER - TO - PEER

Arquitectura P2P Semi-centralizada



ARQUITECTURA PEER - TO - PEER

En una **arquitectura semi-centralizada**, el papel del servidor es **ayudar a establecer contacto entre iguales en la red o para coordinar los resultados de un cálculo**.

* Por ejemplo, si la Figura anterior representa un **sistema de mensajería instantánea**, entonces los **nodos de la red** se comunican con el **servidor** (indicado por **líneas punteadas**), para encontrar **qué otros nodos están disponibles**.

* Una vez que éstos son encontrados, **se pueden establecer comunicaciones directas** y **la conexión con el servidor es innecesaria**.

* Por lo tanto, **los nodos n2, n3, n5 y n6 están en comunicación directa**.

ARQUITECTURA PEER - TO - PEER

En un sistema p2p, donde un cálculo (que requiere un uso intensivo del procesador) **se distribuye a través de un gran número de nodos**, es normal que se distingan **algunos nodos cuyo papel es distribuir el trabajo a otros nodos y reunir y comprobar los resultados del cálculo**.

* Si bien hay **sobrecargas obvias** en los sistemas peer – to - peer, **éstos son una aproximación mucho más eficiente para la computación inter-organizacional que la aproximación basada en servicios** que veremos seguidamente.

ARQUITECTURA PEER - TO - PEER

Todavía hay problemas en el uso de las **arquitecturas p2p**, ya que cuestiones tales como la **protección** y la **autenticidad no están del todo resueltas**.

* Esto significa que los sistemas p2p se usan más en sistemas de información no críticos.