



# INICIO GRABACIÓN



**SANJOSÉ**  
FUNDACIÓN DE EDUCACIÓN SUPERIOR



## INDICE

1

INTRODUCCION A LAS REDES

2

COMPONENTES DE UNA RED

3

RED CONVERGENTE

4

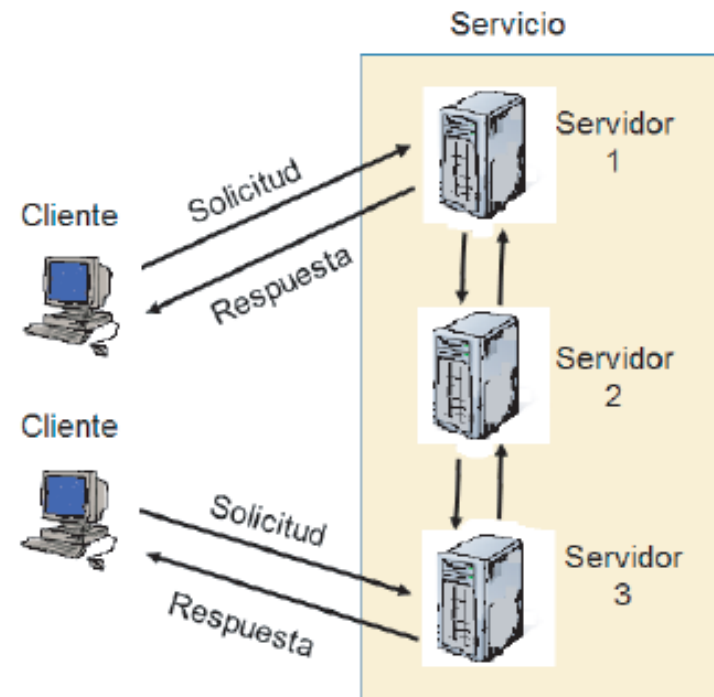
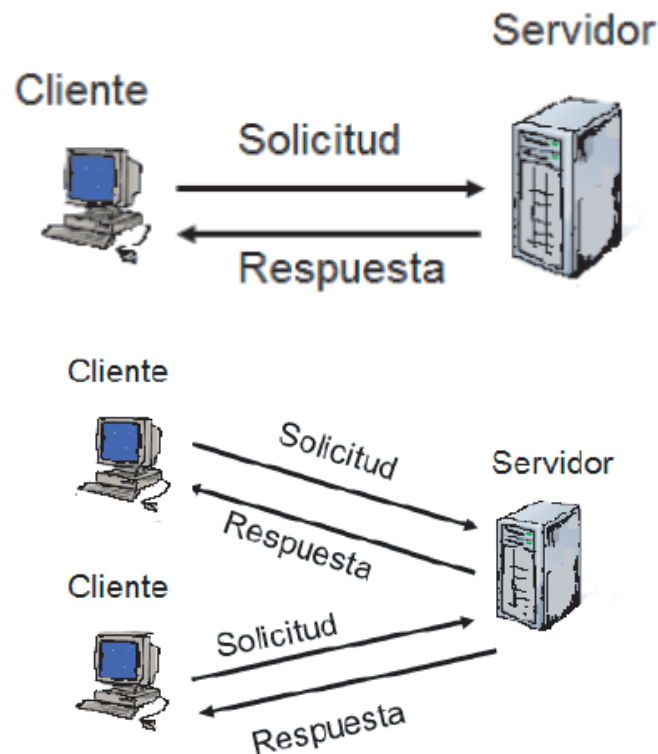
CONCLUSIONES

The background features a photograph of several hands stacked in a circle, symbolizing teamwork. This image is partially covered by a large dark blue circle on the right and a smaller light blue circle on the left. The text is centered within the light blue circle.

# Modelo cliente - servidor

El modelo cliente-servidor es la arquitectura más citada cuando se discuten los sistemas distribuidos. Es el modelo más importante y sigue siendo el más ampliamente utilizado.

En particular, los procesos de cliente interactúan con los procesos de servidor individuales en equipos anfitriones (host) potencialmente separados, con el fin de acceder a los recursos compartidos que administran.





# Proxi



Es un servidor que se emplea como intermediario entre las peticiones de recursos que realiza un cliente a otro servidor. Por ejemplo, si una computadora *A* solicita un recurso a una computadora *C*, lo hará mediante una petición a la computadora *B* que, a su vez, trasladará la petición a la computadora *C*. De esta manera, la computadora *C* no sabrá que la petición procedió originalmente de la computadora *A*. Esta situación estratégica de punto intermedio suele ser aprovechada para soportar una serie de funcionalidades, como:

- Proporcionar caché.
- Control de acceso.
- Registro del tráfico.
- Prohibir cierto tipo de tráfico.
- Mejorar el rendimiento.
- Mantener el anonimato.

El proxy más conocido es el servidor proxy web, su función principal es interceptar la navegación de los clientes por páginas web por motivos de seguridad, rendimiento, anonimato, entre otros.

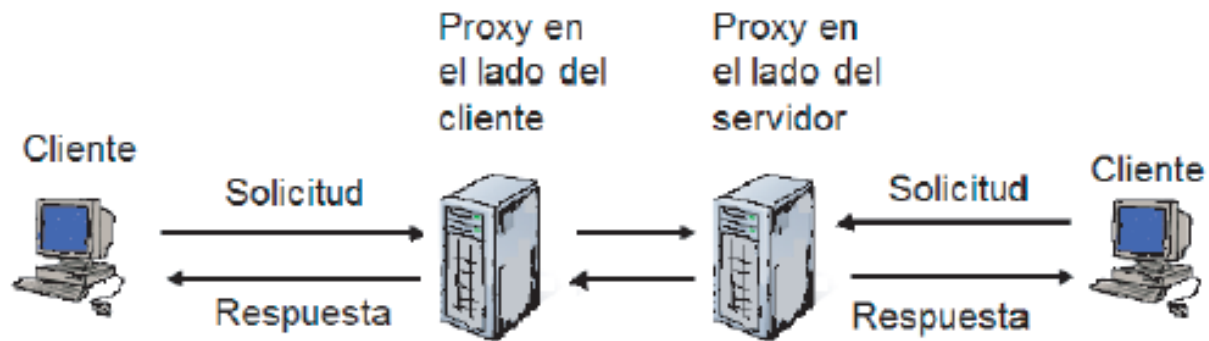
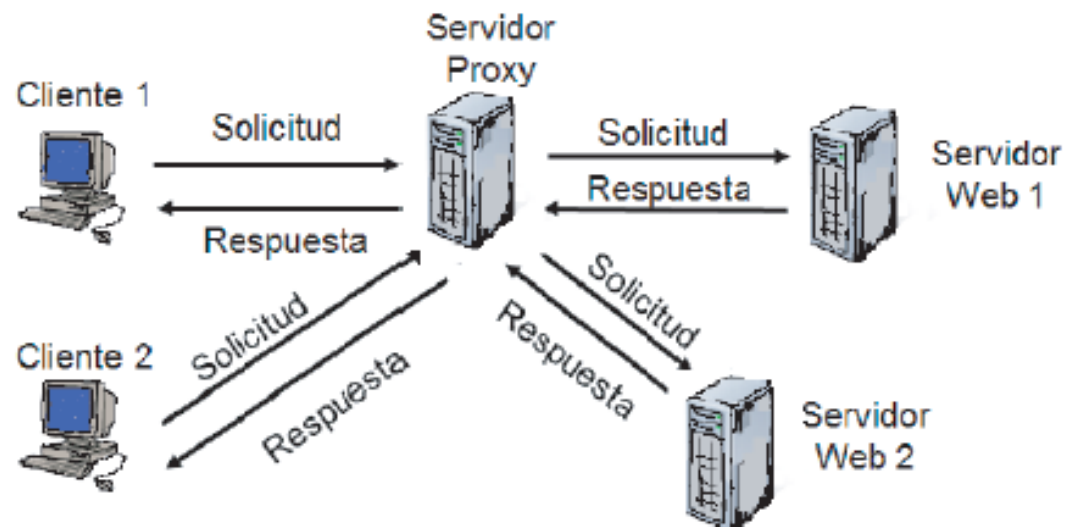


Figura 3.5. Acceso a servidores web vía un proxy





# Clúster



En informática, el término clúster (“grupo” o “racimo”) hace referencia a conjuntos o conglomerados de computadoras contruidos mediante el uso de hardware común y que se comportan como si fueran una única computadora.

El uso de los clústeres varía desde las aplicaciones de supercómputo, servidores web y comercio electrónico hasta el software de misiones críticas y bases de datos de alto rendimiento. El cómputo con clústeres es el resultado de la convergencia de varias tendencias tecnológicas actuales, entre las que se pueden destacar:

- Microprocesadores de alto rendimiento.
- Redes de alta velocidad.
- Software para cómputo distribuido de alto rendimiento.
- Crecientes necesidades de potencia computacional.

Los servicios esperados de un clúster principalmente son:

- Alto rendimiento.
- Alta disponibilidad.
- Escalabilidad.
- Balanceo de carga.



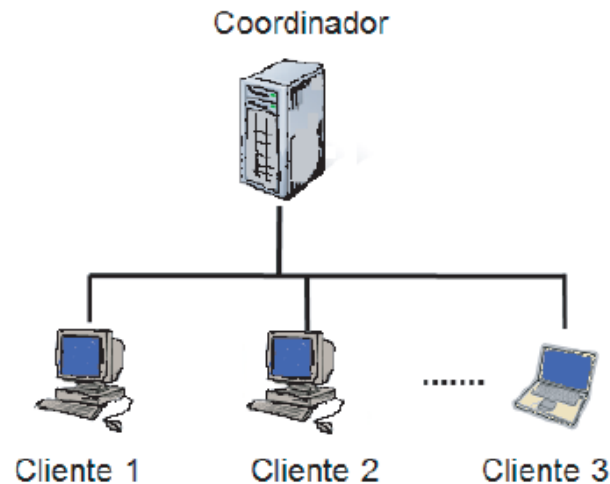


Típicamente respecto a la rapidez y disponibilidad, se espera que un clúster sea más económico que el uso de computadoras individuales.

Un clúster puede ser:

- Homogéneo.
- Semihomogéneo.
- Heterogéneo.

Un clúster es homogéneo cuando todas las computadoras tienen la misma configuración en hardware y sistema operativo. Es semihomogéneo cuando las computadoras tienen diferente rendimiento pero guardan una similitud con respecto a su arquitectura y sistema operativo. Finalmente, un clúster es heterogéneo cuando las computadoras tienen diferente hardware y sistema operativo.







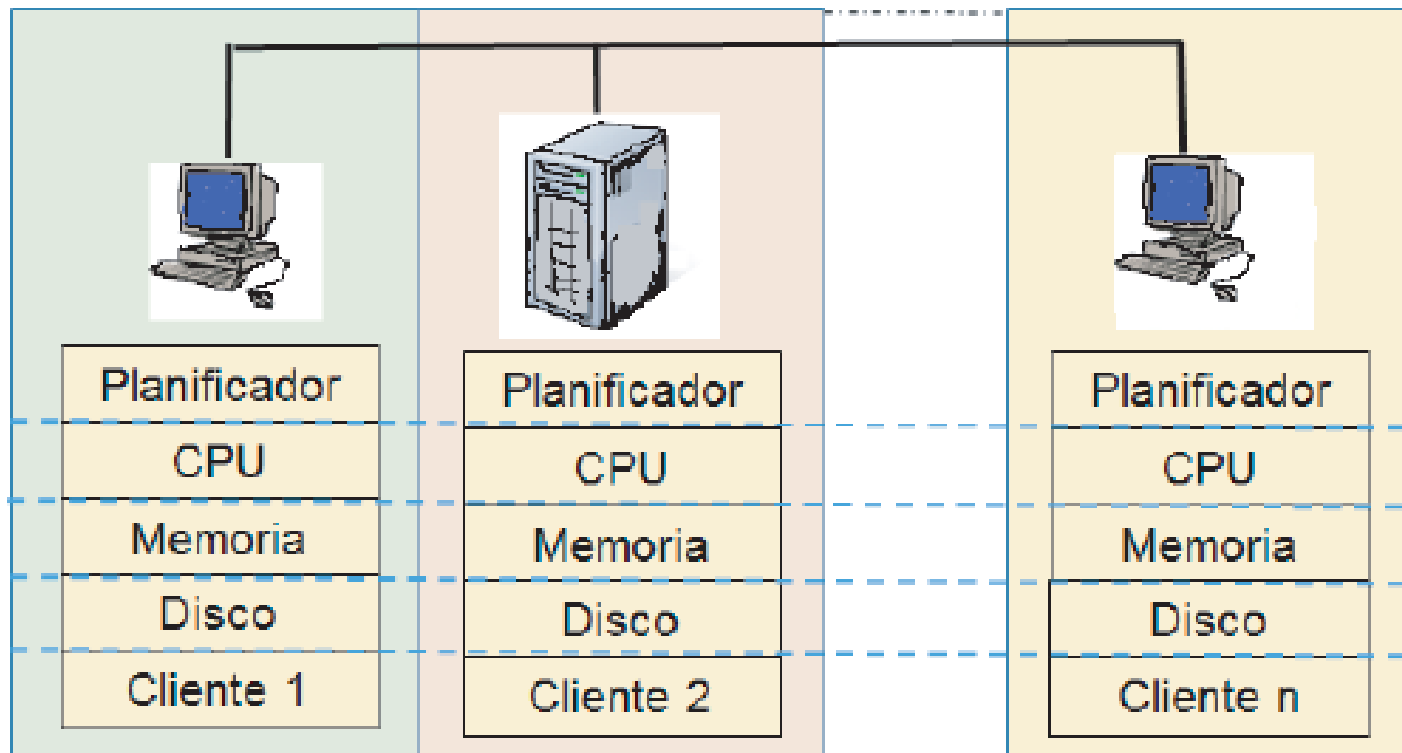
# Grid



El cómputo grid es un paradigma del cómputo distribuido, frecuentemente usado para indicar una infraestructura de gestión de recursos distribuidos que se centra en el acceso coordinado a los recursos informáticos remotos. Estos recursos de cómputo son colectados desde múltiples localizaciones para alcanzar una meta común. A diferencia del cómputo de cluster (en grupo o racimo), el cómputo grid tiende a ser más heterogéneo y disperso geográficamente. Generalmente las grids son usadas para una variedad de propósitos pero puede haber grids especializadas para fines específicos.

Beneficios del cómputo grid :

- Explotación de recursos infrautilizados.
- Capacidad de CPU paralelos.
- Recursos virtuales y organizaciones virtuales para la colaboración.
- Acceso a recursos adicionales.
- Balanceo de recursos.
- Fiabilidad.
- Mejor gestión de infraestructuras de TI más grandes y distribuidos.





# Middleware

En la primera generación de los sistemas distribuidos todos los servicios proporcionados por los servidores debían de programarse a la medida. Así, servicios de acceso a bases de datos, de impresión y transferencias de archivos tenían que ser desarrollados por las propias aplicaciones. Queda en evidencia la necesidad de crear servicios de uso más común por las aplicaciones, de tal manera que pueda incluirse en todas las aplicaciones como software prefabricado. En este escenario surge la idea del middleware, representado por estándares tales como ODBC, OLE, DCOM y CORBA.

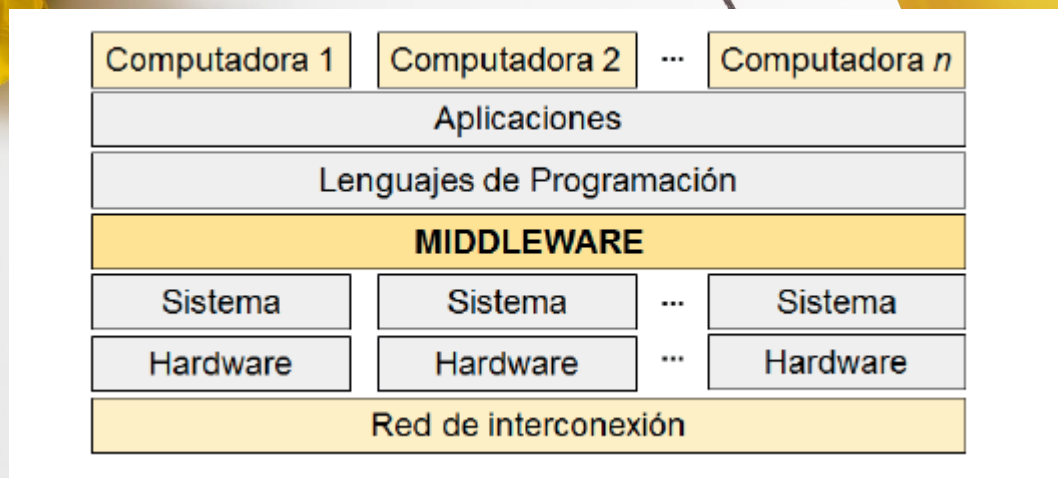
Middleware es un conjunto de servicios que permite distribuir datos y procesos a través de un sistema multitarea, una red local, una red remota o Internet. Los servicios del Middleware pueden ser clasificados en dos grandes grupos: Servicios de desarrollo, Servicios de administración.

El objetivo principal del middleware es conseguir la transparencia en los sistemas distribuidos, por medio de:

- Ofrecer la capacidad, así como solicitar y recibir de manera transparente al sistema.
- Liberar a los diseñadores y administradores del sistema de problemas derivados de la complejidad del sistema operativo.



En la práctica, middleware es representado por procesos u objetos en un conjunto de equipos que interactúan entre sí para implementar la comunicación y el intercambio de recursos de soporte para las aplicaciones distribuidas [Coulouris *et al.*, 2012]. El middleware está relacionado con el suministro de materiales de construcción útiles para la construcción de componentes de software que pueden trabajar con otros en un sistema distribuido. Las abstracciones del middleware apoyan a diversas actividades de comunicación, como la invocación de método remoto, la comunicación entre un grupo de procesos, notificación de eventos, el particionamiento, la colocación y recuperación de objetos de datos compartidos entre los equipos cooperantes, la replicación de objetos de datos compartidos y la transmisión de datos multimedia en tiempo real.





CORBA

CORBA (Component Request Broker Architecture) elaborado y promovido por el Object Management Group



# CORBA



CORBA es una arquitectura de comunicaciones que soporta la construcción e integración de tecnologías de diferente fabricante independientemente del tiempo de creación, así como pueden intercambiar información personas que dominan diferente idioma, sin importar que no sea usado actualmente.

El paradigma orientado a objetos juega un importante rol en el desarrollo de software y cuenta con gran popularidad desde su introducción. La orientación a objetos se comenzó a utilizar para el desarrollo de sistemas distribuidos en la década de 1980. El Grupo de Gestión de Objetos (OMG-Object Management Group) se creó en 1989 como una asociación de las empresas líderes de la tecnología software para definir especificaciones que puedan ser implementadas por ellas y que faciliten la interoperabilidad de sus productos.



Los middlewares basados en objetos distribuidos están diseñados para proporcionar un modelo de programación basado en principios orientados a objetos y, por tanto, para llevar los beneficios del enfoque a objetos para la programación distribuida. Los principales ejemplos de middleware basados en objetos distribuidos incluyen Java RMI y CORBA. Es usado en Distintos sistemas operativos (Unix, Windows, MacOS, OS/2), Distintos protocolos de comunicación (TCP/IP, IPX), Distintos lenguajes de programación (Java, C, C++), Distinto hardware.



# CORBA



Para construir componentes que utilicen el entorno CORBA se deben seguir los siguientes pasos:

1. **Definir la interfaz remota.** Se define, en primer lugar, la interfaz del objeto remoto en IDL. Dicha interfaz permitirá generar, de manera automática, el código fuente del stub y el skeleton así como todo el código necesario para comunicarse con el ORB. Si sólo se implementa el cliente porque el servidor ya existe, se tendría que proporcionar el fichero IDL correspondiente a la interfaz que expone el servidor.
  2. **Compilar la interfaz remota.** El compilador genera todo el código fuente mencionado en el paso anterior.
  3. **Implementar el servidor.** A partir de los esqueletos que genera el compilador idl es sencillo implementar el servidor. Además de los métodos que implementan la interfaz remota, el código del servidor crea un mecanismo para arrancar el ORB y esperar por la invocación de un cliente.
- 
- 





# CORBA



**4. Implementar el cliente.** De una manera similar al servidor, el cliente hace uso de los stubs generados en el paso 2. El cliente se basa en el stub para arrancar su ORB, encontrar el servidor utilizando el servicio de nombrado, obtener una referencia al objeto remoto e invocar sus métodos.

**5. Arrancar los programas.** Una vez está todo implementado, se arranca el servicio de nombrado, el servidor y finalmente, el cliente.






# Características fundamentales de la Arquitectura CORBA



## 1 Objetos CORBA

Las implementaciones de los objetos reciben las invocaciones como llamadas hacia arriba (up-call), desde el ORB hacia la Implementación de la interfaz. La implementación de la interfaz puede elegir un adaptador de objetos entre un conjunto de ellos, una decisión que estará basada en la clase de servicios que pueda requerir dicha implementación.

Los objetos CORBA se diferencian de los objetos de los lenguajes habituales de programación en que:

- Pueden estar localizados en cualquier lugar de la red.
  - Pueden ejecutarse en cualquier plataforma de hardware y de sistema operativo.
  - Pueden estar escritos en cualquier lenguaje.
  - Pueden tener la capacidad de detectar el entorno, procesar información y además tienen la capacidad de comunicación.
- 



## 2. ORB object request bróker



Componente que permite que clientes y objetos puedan comunicarse en un ambiente distribuido. Y que contempla cada una de las interfaces que el ORB manipula. El bus de objetos es el intermediario entre clientes y servidores que transmite las peticiones cliente- servidor y las respuestas servidor-cliente. Se necesita un ORB en cada máquina. El ORB soporta cuatro tipos de interfaces de objetos:

Object Services: Son interfaces para servicios generales. Son usadas en cualquier programa basado en objetos distribuidos.

Common Facilities: Son interfaces orientadas al usuario final y que se programan por la aplicación específica.

Domain Interfaces: Son interfaces de dominio específico para las aplicaciones.

Application Interfaces: Este tipo de interfaz acepta interfaces que no sean estandarizadas y se utilizan en aplicaciones específicas.





#### **4. IDL (Interface Definition Language)**

Para poder especificar los servicios que ofrecen los objetos que forman parte de un sistema abierto y distribuido, se necesita contar con algún lenguaje preciso, bien definido, e independiente de cualquier posible representación de los datos o estructuras que él define, así como la futura implementación de los objetos que especifica. La norma ISO/IEC 14750 (ITUT X.920) define dicho lenguaje, al que se conoce como lenguaje de definición de interfaces de ODP, o ODP IDL por su acrónimo en inglés. Su principal objetivo es describir la signatura de los objetos que especifica, en términos de las estructuras de datos que se manejan y el perfil de las operaciones que definen sus servicios. De esta forma se consigue la ocultación necesaria para el desarrollo de aplicaciones abiertas<sup>10</sup>.





## 4. IDL (Interface Definition Language)

En IDL, una interfaz es una descripción de un conjunto de posibles operaciones que un cliente puede solicitar de un objeto. El objeto satisface una interfaz si este puede satisfacer una solicitud de otro objeto. La interfaz provee mecanismos compuestos que le permiten a tal objeto soportar múltiples interfaces.

Las operaciones que se realizan denotan servicios que pueden ser atendidos y ejecutados para cambiar de valor y adquirir un valor. Una operación es reconocida por un identificador de operación. Una operación no es un valor.

Los tipos de datos que manipula CORBA en IDL son:

Tipos básicos : long, short, ushort, ulong, float, double, char, boolean, enum, string, octet, any

Tipos compuestos: struct, union, array

Tipos derivados: sequence <tipo>

Tipos de objeto: interface, referencia a objetos



## 4. IDL (Interface Definition Language)

Un tipo es una entidad con predicados asociados y definidos con valores en un objeto. Un valor satisface un tipo si el predicado es verdadero para la variable.

Los tipos son usados para restringir los posibles valores, parámetros, o para identificar un posible resultado.

La Interfaz IDL se compone del repositorio de interfaces y la interoperabilidad de la Interfaz de invocación dinámica:

- El repositorio de interfaces


El repositorio de interfaces (IR) es un servicio que ofrece objetos persistentes que representan la información IDL de las interfaces disponibles en CORBA, de una forma accesible en tiempo de ejecución (runtime). Esta información puede ser utilizada por el ORB para realizar peticiones. Y además, el programador de aplicaciones puede utilizar esta información para acceder a objetos cuya interfaz no se conoce en tiempo de compilación, o para determinar que operaciones son válidas en un objeto.



## 4. IDL (Interface Definition Language)

- La interfaz de invocación dinámica

El DII (Dynamic Invocation Interface) es una interfaz que nos permite la construcción dinámica de invocaciones para un determinado objeto. Ello garantiza que el cliente pueda especificar el objeto, la invocación y los parámetros que se pasan al servidor. La invocación es idéntica a la que llega a través de la interfaz estática pero que ya dentro del cliente, logra una flexibilidad fundamental en arquitecturas complejas y dinámicas. Una invocación dinámica se compone, de una referencia al objeto, una operación y una lista de parámetros. Todos estos datos se obtienen del Repositorio de Interfaces (IR).






## 5. Stub

Es el intermediario entre el cliente y el ORB. El Stub recoge del cliente llamadas a métodos y las transmite al ORB. Se requiere una clase de stub por cada clase remota.

Además, es un componente que actúa como servidor, puede estar ejecutándose en cualquier máquina conectada a la red que recibe peticiones por parte de clientes que pueden ser locales o remotos. Indistintamente de ello, el cliente siempre tendrá la ilusión de que la llamada se ejecuta localmente. En otras palabras el stub logra que el programador no se ocupe de las instrucciones de programación remotas ya que son objetos que residen en el cliente y que representan objetos remotos instalados en un servidor. En él se identifica: Host, puerto e identificador del objeto.





## 6. Esqueleto

Es el intermediario entre ORB y los objetos del servidor. Recibe llamadas del ORB y ejecuta los métodos correspondientes en el servidor sobre el objeto que corresponda. Cuando el cliente establece un objeto local (con servicio remoto), la petición se realiza por intermedio del protocolo de comunicaciones IIOP a través del ORB. El servidor recibe la petición, busca el objeto definido (compara el esqueleto del método en el módulo esqueleto) lo ejecuta y retorna la respuesta al cliente.



The background of the slide features a photograph of several hands of different skin tones clasped together in a supportive grip. Overlaid on this is a large, semi-transparent dark blue circle on the right side and a smaller, semi-transparent light blue circle on the left side, which contains the text. A small, solid dark blue sphere is positioned at the bottom of the light blue circle.

# Ventajas al utilizar CORBA



# Comunicación transitoria orientada a mensajes



## Heterogeneidad

Un sistema heterogéneo consiste en conjuntos de elementos interconectados de hardware y software de diferente fabricante y que puede integrar aplicaciones de diferente tecnología.

La infraestructura de sistemas de información antiguos que poseen las compañías no son fácilmente reemplazable, debido al costo de desarrollo y al tiempo de implantación, una de las mejores alternativas es integrar antiguas tecnologías con nuevas para así obtener un completo beneficio.



## Movilidad

La migración de procesos en sistemas distribuidos tradicionales es muy útil para mejorar el reparto de carga de los diferentes computadores. Tiene como fin garantizar el rendimiento global y ciertas restricciones de administración o seguridad.

## Eficiencia

La red lleva menos mensajes, el servidor realiza más trabajo, se evita la latencia/inestabilidad de la red en los procesos.

## Adaptación al cliente

- El cliente puede extender la funcionalidad del servidor.
  - Fácil instalación para el usuario.
  - No se requiere instalación de servidor.
  - No se acuerdan los procedimientos entre los clientes y los servidores.
  - Instalación dinámica de los procedimientos del cliente en el servidor.
- 
- 



# Comunicación transitoria orientada a mensajes



## **Tiempo de desempeño**

Además, la ejecución asíncrona permite que los procesos controlen la gestión y terminación de tarea y que el cliente pueda finalizar o continuar haciendo otras cosa en su sistema, por otro lado se reduce el tráfico en la red y la capacidad de cómputo del cliente

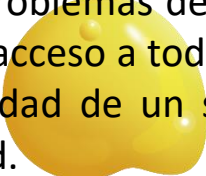
## **Movilidad**

Reducción de la dependencia de la disponibilidad de la red y del cliente/servidor.

Los procesos migrados al sistema servidor no se ven afectados por los fallos del cliente o de la red. Los procesos se ejecutan realizando tareas específicas en lugares diferentes.

Automatización de las tareas distribuidas.

El problema fundamental de los sistemas de integración es el software. Aún no existe mucha experiencia en el diseño, implantación y uso de software como CORBA. Precisamente, éste es un campo de investigación actual. Las redes son indispensables para la comunicación entre máquinas; sin embargo, pueden plantear problemas de saturación, embotellamiento, interrupción o pérdidas de mensajes. El posible acceso a todo el sistema por parte de los usuarios plantea el inconveniente de la necesidad de un sistema de seguridad adecuado y estándar, aunque CORBA maneja la seguridad.





FUNDACIÓN DE EDUCACIÓN SUPERIOR

**SAN JOSÉ**

INSTITUCIÓN TECNOLÓGICA

FIN DE  
GRABACIÓN