Tema 1:

PARADIGMAS DE COMPUTACIÓN

Un **sistema distribuido** es un conjunto de elementos de computación independientes, interconectaos, que comunican y coordinan sus acciones a través de una red de comunicaciones.
Ejemplos SD: internet, intranets privadas, computación ubicua.

La **computación distribuida** es aquella que se desarrolla en un sistema distribuido: servicios y aplicaciones de red.

Características de un sistema distribuido:
La **heterogeneidad** es un rasgo característico de los paradigmas de computación:

- Estandarización
- Representación de datos
- Representación de código
- Representación de objetos
- Protocolos

La **extensibilidad.**
La **escalabilidad**.
La **seguridad**:

- Entornos que puedan ser atacados
- Confidencialidad
- Integridad
- Disponibilidad
- Firewalls, SSL, HTTPS, Radius, Kerberos.

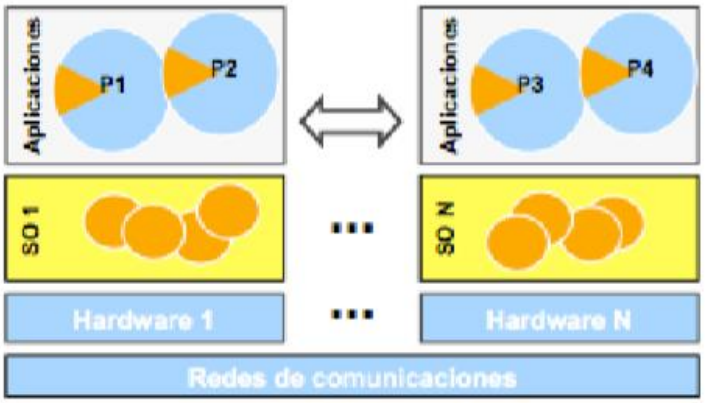
La **Concurrencia y sincronización**:

- De acceso
- De ubicación
- De movilidad
- De escalabilidad
- Frente a fallos

La tolerancia a fallos.
La transparencia.

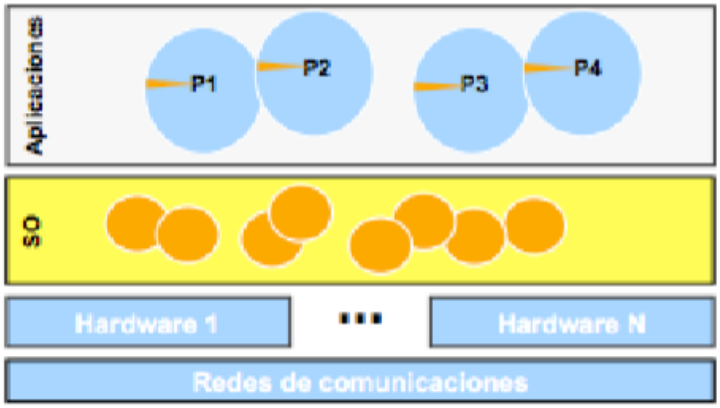
SISTEMAS OPERATIVOS EN RED

Tener implementado un SO en red implica flexibilidad y técnicas maduras. Tiene la desventaja de falta de transparencia y mayor esfuerzo de integración. Ejemplos: Linux, Windows, Novell NetWare.



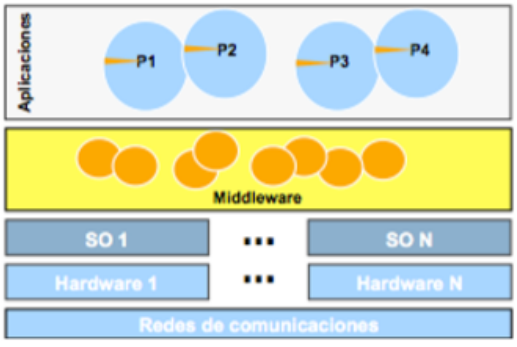
SISTEMAS OPERATIVOS DISTRIBUIDOS

Tener implementado un SO distribuido implica las ventajas de transparencia, escalabilidad y facilidad de integración (características de un sistema distribuido). Las desventajas es que las técnicas son complejas, se necesitan comunicaciones de alta velocidad y hay mucha competencia de mercado. Ejemplos: Mach, Amoeba.



MIDDLEWARE

Tiene un enfoque mixto. El modelo conceptual está en SOD y las infraestructuras en SOR. El middleware es la capa por encima de SO. Es homogéneo. Las principales ventajas son la flexibilidad, transparencia, integración, madurez y escalabilidad. Las desventajas son las plataformas heterogéneas y la necesidad de estandarización.



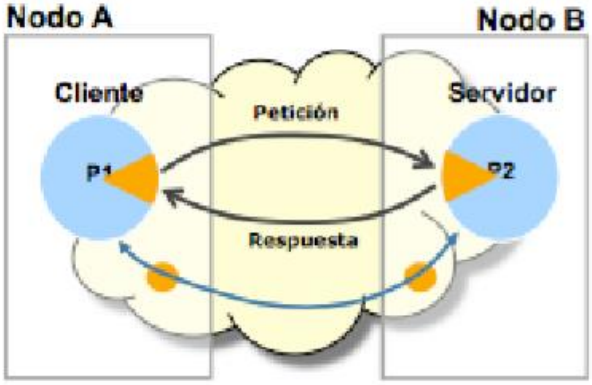
MODELOS ARQUITECTÓNICOS

CLIENTE/SERVIDOR

Este paradigma asigna roles diferentes a dos procesos que colaboran.
Se realizará la abstracción del acceso de recursos. Los roles serán:

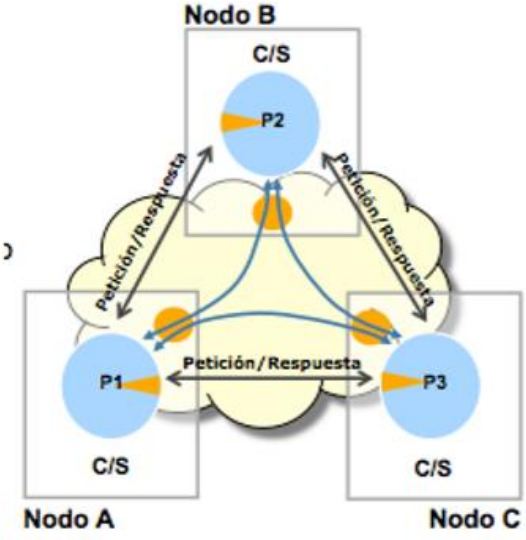
- Proceso servidor. Interpreta el papel de proveedor de servicio, esperando de forma pasiva la llegada de peticiones. Se encargará de escuchar y aceptar peticiones.
- Proceso cliente. Invoca determinadas peticiones al servidor y aguarda respuestas. Se encargará de solicitar peticiones y aceptar respuestas.

Este modelo proporciona una abstracción eficiente para facilitar los servicios en red. La sincronización se simplifica: el servidor espera peticiones y el cliente espera respuestas.



PEER-TO-PEER

Los procesos participantes interpretan los mismos papeles, con idénticas capacidades y responsabilidades.
Cada participante puede solicitar una petición a cualquier otro participante y recibir respuesta.
Este paradigma resulta apropiado para aplicaciones como mensajería instantánea, compartición de archivos, video conferencia y trabajo colaborativo. También es posible que una aplicación se base en ambos modelos: cliente/servidor y peer-to-peer.



MOM: MIDDLEWARE ORIENTADO A MENSAJES

- El modelo MOM (Middleware Orientado a Mensajes) es una evolución del sistema de paso de mensajes convencional. Este permite que la comunicación entre el emisor y el receptor se produzca de una manera completamente desacoplada. Se utiliza en sistemas asíncronos, es decir, donde el emisor no necesita una respuesta inmediata del receptor tras enviar el mensaje y puede continuar con su funcionamiento. Existe un intermediario, conocido como proceso intermediario, un emisor, proceso emisor y un receptor, proceso consumidor.

El proceso intermediario es el encargado de almacenar los mensajes del emisor, generalmente este es un middleware que incluye ciertos servicios para el tratamiento de los mensajes. Puede poseer las siguientes características: gestión de prioridades de mensajes, temporizadores para la gestión de mensajes, gestión de formatos de mensajes, gestión de seguridad, gestión de persistencia de los mensajes.

Es un modelo adecuado para aplicaciones y sistemas donde se quiere establecer un único punto de entrada a una comunicación asíncrona de los mismos.

Existen dos variantes de este modelo:

- Punto a punto (1:1) → cada mensaje enviado por el emisor únicamente sera procesado por un proceso consumidor. El agente intermediario obtiene el mensaje, lo procesa y el mensaje es eliminado del intermediario.
- Publicación/Subscripción (1:M) → un mensaje publicado por un emisor será procesado por todos los agentes consumidores que se hayan subscrito a dicho proceso intermediario.

- Las principales diferencias entre SOA y MOM son:

- MOM es usado en comunicación asíncrona, mientras que SOA es petición/respuesta (síncrona).
- MOM consigue desacoplar el escenario completo gracias al proceso intermediario, en SOA únicamente se desacopla la localización de los servicios.
- MOM puede ser usado para la implementación de la arquitectura SOA.

- Debido a que MOM se apoya en un proceso intermediario para la gestión de los mensajes, una comunicación adecuada sería ORB. ORB dispone de un agente intermediario para la gestión de objetos ofreciéndonos la abstracción del acceso a objetos heterogéneos. Por otro lado, la arquitectura de servicios web también nos podría dar un enfoque MOM con una mayor interoperabilidad a más alto nivel.

Explica el modelo arquitectónico denominado middleware orientado a mensajes (MOM). Enumera y describe sus elementos. ¿Cuáles son las principales similitudes y diferencias entre MOM y el modelo cliente-servidor convencional?

- Mientras que el modelo C/S actúa sobre el paradigma de paso de mensajes, la arquitectura MOM es una elaboración más extensa del mismo. En el modelo MOM nos encontramos con un proceso intermediario, mientras que en la arquitectura C/S solo existen dos procesos (cliente y servidor). Ambos nos permiten la abstracción del acceso a recursos de red.

ARQUITECTURA ORIENTADA A SERVICIOS (SOA)

- La arquitectura SOA (Service Oriented Architecture) es un paradigma de arquitectura para desarrollar sistemas distribuidos. Permite la abstracción de acceso a actividades de negocio, conocidas como servicios.

Sus principales características son:

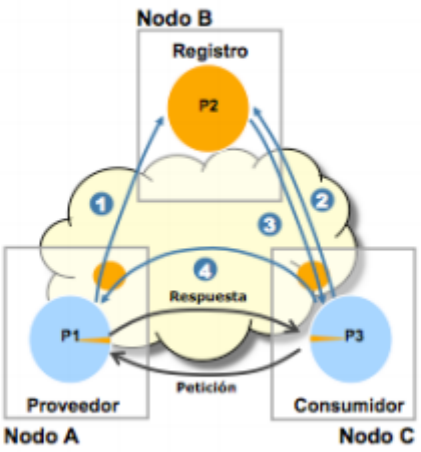
- Localización, descubrimiento y publicación.
- Interoperabilidad.
- Composición.
- Autnomía y autocontenidos.
- Reusabilidad y desacoplamiento.
- Contrato bien definido.
- Sin estado

El funcionamiento SOA se basa en que un proceso proveedor publica en un proceso registro los datos necesarios para que un proceso consumidor pueda recurrir a consumir. Hecho esto, el proceso consumidor realizará una búsqueda del proceso proveedor. El registro enviará la información necesaria al servicio de consumidor para realizar la conexión y ya podrá consumir el recurso del proveedor directamente.

El esquema a seguir de la explicación anterior es el siguiente:

1. **Publicación**: proveedor se registra en el registro
2. **Búsqueda**: consumidor busca en registro a proveedor
3. **Descubrimiento**: registro proporciona a consumidor datos proveedor
4. **Consumo**: el consumidor realiza la petición a proveedor

Algunas herramientas utilizadas para SOA son: servicios web, JINI, UPNP.
Cuando un cliente localiza un servicio jini a través de un servicio de búsqueda, el propio servidor de búsqueda actúa como elemento que concede periodos de alquiler para el servicio.

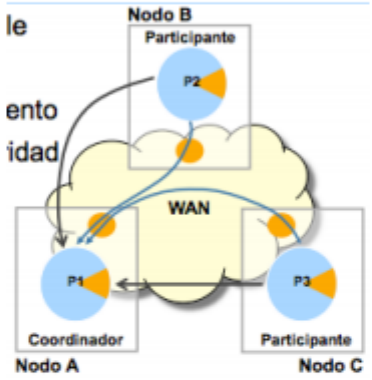


CLUSTER Y GRID

Infraestructuras hardware y software para ofrecer mayor capacidad de procesamiento y almacenamiento. Se basan en la unión de un conjunto de computadores que se unen para dar forma a un supercomputador.

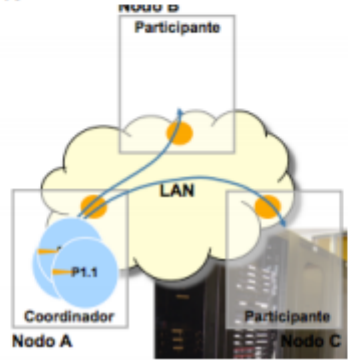
Grid

Heterogeneidad, por lo que es más flexible. Se dan en internet. El coordinador coordina los procesos que se ejecutan en los participantes, sin decidir cuál se ejecuta en cada participante.



Clúster

Homogeneidad. Se dan en una red local de alta velocidad. El coordinador decide qué procesos se ejecutan en cada participante(gestor de recursos centralizado). Algunas aplicaciones son: servidores web y de aplicaciones, sistemas de información o supercómputo.



MECANISMOS DE COMUNICACIÓN

El mecanismo básico en los sistemas distribuidos es la comunicación IPC. Los modelos básicos son Unidifusión (unicast) y multidifusión (multicast).
Las operaciones pueden ser bloqueantes o síncronas o no bloqueantes o asíncronas.

El intercambio de datos se realiza en el nivel de red mediante el flujo binario. Es importante la existencia de heterogeneidad, en la que todos los elementos de la comunicación se adapten a la representación de su receptor o emisor y que haya una representación externa común. Para resolver el problema de la representación común hay tres tipos de soluciones más o menos factibles:

- El proceso emisor adapta antes de enviar los datos al receptor. No es muy fiable puesto que siempre se debe conocer el formato de representación del receptor.
- El proceso receptor adapta los datos que le envía el emisor. Junto a la información enviada se debería incluir información del formato de representación del emisor.
- Uso de una representación externa que negocien emisor y receptor. De esta forma el emisor enviará la información ya transformada al formato acordado. El emisor se encargará de transformar la información a su sistema de representación interno. La mayoría de plataformas middleware utilizan esta solución.

Ejemplos de lenguajes externos de representación estándar son:

- XDR (eXternal Data Representation). RPC.
- ASN.1 (Abstract Syntax Notation). LDAP O DNS.
- XML (eXtensible Markup Language). Uno de los más usados.
-

El marshalling o serialización es un proceso de codificación de un objeto en un medio de almacenamiento, con el fin de transmitirlo a través de una conexión de red. Dos etapas:

- la primera es serialización. Los datos son transformados para transmitirse a través de la red.
- La codificación, que los adaptará a una representación externa antes de enviar.

PROTOCOLOS

Basados en texto o binario:

- HTTP, SMTP, POP3
- LDAP, DNS

Tipo de patrón de mensaje:

- Petición- respuestas
- Solicitud-respuestas
- Unidireccional
- Notificación

Técnicas de comunicación:

- Pool
- Push

Orientados o no a la conexión:

- HTTP, FTP
- DNS, DHCP

Con o sin estado:

- FTP, http

PASO DE MENSAJES

Intercambio de información entre procesos mediante mensajes. La interfaz mínima requerida debería permitir el envío y el recibimiento de mensajes. La interfaz es dependiente de la conexión, dependiendo de si está conectado o no.

Sockets

Un socket es un punto de comunicación entre 2 agentes, por el cual se puede emitir o recibir información.
Mecanismo que permite a los procesos comunicarse y sincronizarse entre si, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la red subyacente. Esta interfaz de comunicaciones es una de las distribuciones de Berkeley al sistema UNIX, implementándose las utilidades de interconectividad de este sistema operativo.
La comunicación puede ser tanto en local como en remota, siendo la familia INET la más utilizada. La conexión puede ser:

- flujo de datos, confiable y ordenado.
- Datagrama, no confiable y sin orden.

Un ejemplo de la comunicación por sockets es RPC.

RPC (llamadas a procedimientos remotos)

RPC es un protocolo que permite a un programa ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.
Las llamadas a procedimientos remotos que pueden invocarse desde una máquina cliente pertenecen normalmente e equipos servidores.

Estas llamadas son muy utilizadas dentro de cliente-servidor, siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de la operación al cliente.

Invocación de métodos remotos (RMI)

Es un mecanismo ofrecido por Java para invocar un método de manera remota. Proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java. SI SE REQUIERE COMUNICACIÓN ENTRE DISTINTAS TECNOLOGÍAS DEBEMOS USAR CORBA O SOAP EN LUGAR DE RMI.

Se caracteriza por la facilidad de su uso en la programación Java. A través de RMI se puede exportar un objeto, con lo que pasará a estar accesible a través de la red y el programa permanece en espera de peticiones en puerto tcp. A partir de ese momento, el cliente puede invocar los métodos remotos.
Los componentes de RMI son:

- Interfaz remota: definición de métodos remotos
- Objeto remoto: implementación de métodos remotos
- Cliente
- Stub y skeleton: los skeletons reciben las peticiones de los clientes, realiza la invocación del método y devuelve resultados.
- Servicio de nombres
- Servicio de registro: registro del objeto remoto

La arquitectura RMI podría resumirse en:

1. Registro del objeto remoto.
2. El cliente solicita una referencia al objeto remoto.
3. El servidor devuelve la referencia al objeto remoto.
4. El cliente instancia un método remoto a través de stubs.
5. El proceso servidor devuelve el resultado.

El marshalling se produce en el objeto cliente/stub.

Agente intermediario para la gestión de objetos.

Es la base de la arquitectura CORBA. Los elementos básicos de esta comunicación de alto nivel son:

- ORB (agente intermediario para la gestión de objetos).
- IDL y CDR (interfaces y representación externa).
- GIOP (protocolos de comunicación).

Es una capa de software tipo middleware que recibe peticiones de un proceso que está en una máquina y le permite realizar una llamada a un método de un objeto que se encuentra en una máquina remota.

Los pasos a seguir para utilizar este paradigma de computación distribuida son:

1. El cliente, que tiene una referencia del objeto remoto, un nombre de la operación que desea realizar y una serie de parámetros, solicita un servicio de la implementación del objeto.
2. El ORB transporta la petición que invoca al método usando los adaptadores del objeto.
3. La petición es manejada por el ORB, el cual debe localizar el objeto buscado y mandársela.
4. El ORB es responsable de devolver los resultados al cliente de una forma adecuada.

Decisiones de diseño

Abstracción frente a sobrecarga.

Escalabilidad. Un sistema es escalable si conserva su efectividad cuando ocurre un incremento significativo en el número de recursos y el número de usuarios.