Introducción a los Sistemas Distribuidos

Desde el *inicio de la era de la computadora moderna* (1945), hasta cerca de 1985, sólo se conocía la *computación centralizada*

A partir de la mitad de la década de los ochentas aparecen dos avances tecnológicos fundamentales:

- Desarrollo de **microprocesadores** poderosos y económicos con arquitecturas de 8, 16, 32 y 64 bits.
- Desarrollo de **redes de área local** *(LAN)* de alta velocidad, con posibilidad de conectar cientos de máquinas a velocidades de transferencia de millones de bits por segundo (mb/seg).

Introducción a los Sistemas Distribuidos

Surgen **Sistemas Distribuidos**, en contraste con los **Sistemas *Centralizados.***

Estos nuevos Sistemas necesitan un software distinto

Los usuarios pueden acceder a una *gran variedad de recursos computacionales:*

- De hardware y de software.
- Distribuidos entre un gran número de sistemas computacionales conectados.

Un importante antecedente de las redes de computadoras lo constituye Arpanet, iniciada en 1968 en los EE. UU.

Introducción a los Sistemas Distribuidos

Un Sistema Distribuido es un conjunto de procesadores que no comparten memoria, ni reloj, en vez de esto, cada procesador cuenta con su propia memoria local y los procesadores se comunican entre sí a través de diversas líneas de comunicación, ofreciendo al usuario acceso a los diversos recursos que mantiene el sistema.

El acceso a recursos compartidos permite acelerar los cálculos y mejorar la disponibilidad y confiabilidad de los datos.

Un sistema distribuido es aquel que se ejecuta en una colección de máquinas sin memoria compartida, pero que aparece ante sus usuarios como una sola computadora

Características de un Sistemas Distribuidos

Comparte recursos: Si varias instalaciones están conectadas entre sí, entonces el usuario de una instalación puede utilizar los recursos disponibles en otra.

Acelera cálculos: Si un cálculo puede dividirse en varios subcálculos que pueden ejecutarse concurrentemente, entonces la disponibilidad de un sistema distribuido nos permite distribuir los cálculos entre varias instalaciones y ejecutarlos en forma concurrente.

Características de un Sistemas Distribuidos

Confiabilidad: el funcionamiento de todo el sistema no debe estar ligado a ciertas máquinas de la red, sino que cualquier equipo pueda suplir a otro en caso de que uno falle. La forma más evidente de lograr la fiabilidad del sistema es uso de redundancia. La información no debe estar almacenada en un solo servidor de archivos. Otro tipo de redundancia más compleja son los procesos. Las tareas críticas podrían enviarse a varios procesadores independientes.

Características de un Sistemas Distribuidos

Comunicación: La comunicación entre procesos en sistemas con un único procesador se lleva a cabo mediante el uso de memoria compartida entre los procesos. En los sistemas distribuidos, al no haber conexión física entre las distintas memorias de los equipos, la comunicación se realiza mediante la transferencia de mensajes.

S.O.Distribuido <> S.O.Red

En un S.O. de Red las computadoras están interconectadas por medios de comunicación: software y hardware. En este tipo los usuarios saben donde están ejecutando su trabajo y guardando su información. Mientras en un S. O. Distribuidos existe un software que distribuye las tareas de los usuarios sobre una red de computadoras y para los usuarios es transparente donde realizan sus tareas y guardan su información.

Sistemas Distribuidos

Existen dos esquemas básicos de éstos sistemas. Un sistema fuertemente acoplado es aquel que comparte la memoria y un reloj global, cuyos tiempos de acceso son similares para todos los procesadores. En un sistema débilmente acoplado los procesadores no comparten ni memoria ni reloj, ya que cada uno cuenta con su memoria local.

Sistemas Distribuidos

Los S. Distribuidos se inician a partir del año 1985

Un Sist. Dist. Es un conjunto de procesadores conectados por diversas líneas, que no comparten ni memoria, ni reloj.

Características:

Comparte recursos,
Acelera los cálculos,
Comunicación,
Confiabilidad

Sistema Fuertemente Acoplado – SI Comparte Memoria y reloj
Sistema Débilmente Acoplado – NO Comparte Memoria y reloj

Ventajas de los Sistemas Distribuidos

Los sistemas distribuidos generalmente tienen en potencia una proporción precio / desempeño mucho mejor que la de un único sistema centralizado

Aceleración de Cálculos, la carga de trabajo se puede difundir entre las máquinas.

Desventajas de los Sistemas Distribuidos

El principal problema es el software, ya que el diseño, implantación y uso del software distribuido presenta numerosos inconvenientes.

El hecho de que sea fácil compartir los datos es una ventaja pero se puede convertir en un gran problema (pérdidas de mensajes, saturación en el tráfico, etc.) por lo que la seguridad debe organizarse adecuadamente.

Conceptos

Interoperabilidad: es la habilidad del sistema de facilitar intercambio de información entre los componentes heterogéneos en el sistema.

Transparencia: este concepto es muy parecido al de máquina virtual en los sistemas operativos tradicionales, la transparencia en los sistemas operativos distribuidos, esta es la propiedad que permite a los usuarios ver al conjunto de máquinas en las que esta trabajando como una sola máquina.

Autonomía: es la independencia de los sistemas operativos con respecto al hardware, lo que permite que el sistema trabaje con unidades autónomas.

Conceptos de Hardware

Todos los sistemas distribuidos presentan varias CPU, organizadas de distintas formas, como puede ser la manera de *interconectarlas* y los *esquemas de comunicación* utilizados.

Existen diversos esquemas de clasificación para los sistemas de cómputos con varias CPU , uno de los mas conocidos es la “Taxonomía de Flynn” la misma considera como características esenciales el número de flujo de instrucciones y el número de flujos de datos.

Conceptos de Hardware

SISD (Single Instruction Single Data)

Un flujo de instrucciones y un flujo de datos

Poseen un único procesador.

SIMD (Single Instruction Multiple Data)

Un flujo de instrucciones y varios flujos de datos

Se refiere a ordenar procesadores con una unidad de instrucción que:

- Busca una instrucción.
- Instruye a varias unidades para que la lleven a cabo en paralelo, cada una con sus propios datos.

Son útiles para los cómputos que repiten los mismos cálculos en varios conjuntos de datos.

Conceptos de Hardware

MISD (Multiple Instruction Single Data)

Un flujo de varias instrucciones y un solo flujo de datos. No se presenta en la práctica

MIMD (Multiple Instruction Multiple Data)

Un grupo de computadoras independientes,

cada una con su propio contador del programa, programa y datos

Todos los sistemas distribuidos son de este tipo.

Conceptos de Hardware

División de las computadoras **MIMD**

Multiprocesador: poseen memoria compartida (Los distintos procesadores comparten el mismo espacio de direcciones virtuales).

Multicomputadoras: no poseen memoria compartida (grupo de PC conectadas mediante una red).

Cada una de las categorías indicadas se puede clasificar según la arquitectura de la red de interconexión

Tipo de Conexiones

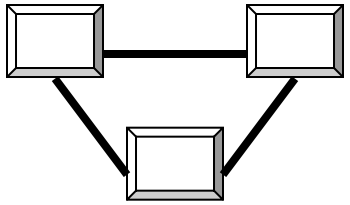
Objetivo básico es **compartir recursos**, que estén disponibles para cualquiera de la red que lo solicite, sin importar la localización del recurso y del usuario.

Un segundo objetivo es proporcionar una **alta fiabilidad**, al contar con fuentes alternativas de suministro. Si una de ellas deja de funcionar, las otras pueden ser capaces de encargarse de su trabajo, aunque se tenga un rendimiento menor.

Otro objetivo es el **ahorro económico**.

Una red de computadoras puede proporcionar un poderoso medio de comunicación entre personas que se encuentran muy alejadas entre sí.

Conexión Total

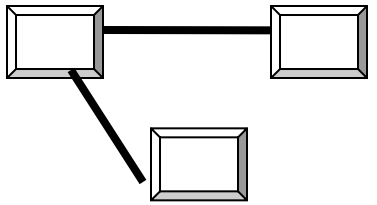


En una red de conexión total cada instalación está enlazada directamente con todas las demás instalaciones del sistema.

En este entorno los mensajes pueden enviarse con ***gran rapidez***, dado que un mensaje sólo requiere viajar por un único enlace, estos sistemas son muy ***confiables*** ya que deben averiarse muchos enlaces para particionar el sistema.

Un sistema ha sido particionado si se divide en dos o más subsistemas que carecen de conexión entre sí.

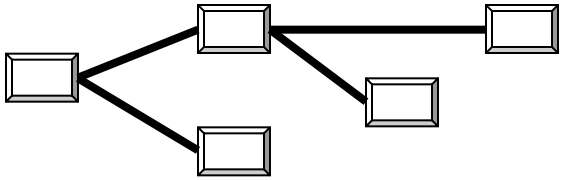
Conexión Parcial



En una red de conexión parcial hay un enlace directo entre algunos, pero no todos, los pares de instalaciones, por consiguiente, el ***costo*** básico de esta configuración es ***menor*** al de una red de conexión total.

Es posible que un mensaje enviado de una instalación a otra tengas que pasar por varias instalaciones intermedias, lo que hace ***más lenta*** la comunicación; además ***no es tan confiable*** como una red de conexión total, puesto que la falla de un enlace puede particionar la red.

Jerarquía

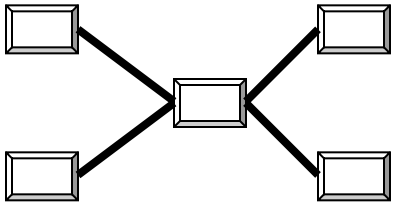


Es una red jerárquica las instalaciones se organizan como un árbol. Cada instalación tiene un solo padre y varios hijos.

El ***costo*** básico de esta configuración generalmente es ***menor*** al del esquema de conexión parcial. En este entorno, un padre y un hijo se comunica directamente y los hermanos sólo pueden hacerlo a través de su padre en común. Lo mismo con los primos.

Si falla una instalación padre, entonces sus hijos no pueden comunicarse entre sí ni con otros procesadores. Esto provoca la partición de la red en varios subárboles disjuntos.

Estrella

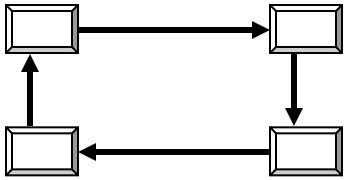


En una red estrella una de las instalaciones del sistema está conectada con todas las demás, ninguna de las otras instalaciones está conectada con otra.

Este esquema proporciona ***bajo costo*** de comunicación ya que un mensaje a lo sumo realiza dos transferencia, no obstante, no asegura rapidez puesto que la instalación central se puede convertir en un cuello de botella.

Si falla la instalación central, la red sé particiona por completo.

Anillo

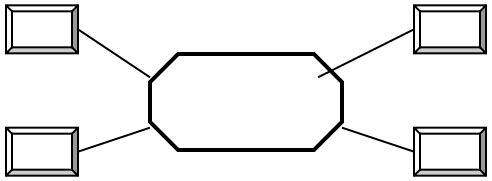


En una red anillo cada instalación está físicamente conectado a exactamente otra dos.

El anillo puede ser unidireccional o bidireccional.

El costo de comunicación puede ser elevado ya que un mensaje de una instalación a otra viaja por el anillo hasta que llega a su destino.

Canal Multiacceso



En una red de canal multiacceso existe un solo enlace compartido (el canal). Todas las instalaciones del sistema se conectan directamente a ese enlace que se puede organizar como una línea recta o como un anillo.

Las instalaciones pueden comunicarse directamente por medio de este canal. El ***costo básico*** de la red ***es la línea*** con él numero de instalaciones y el costo de comunicación es bastante bajo, a menos que el enlace se convierta en un cuello de botella.

La falla de una instalación no afecta a la comunicación entre las demás, sin embargo si falla el enlace la red queda completamente particionada.

Modelos de Sistemas Distribuidos

Multiprocesadores con Base en Buses: Consta de cierto número de μ_p conectados a un bus común, junto con un módulo de memoria. Un bus típico posee al menos: 32 líneas de direcciones, 32 líneas de datos y 30 líneas de control. Todos operan en paralelo. Para leer una palabra de memoria, una cpu:

•Coloca la dirección de la palabra deseada en las líneas de direcciones del bus.

•Coloca una señal en las líneas de control adecuadas para indicar que desea leer.

•La memoria responde y coloca el valor de la palabra en las líneas de datos para permitir la lectura de esta por parte de la cpu solicitante.

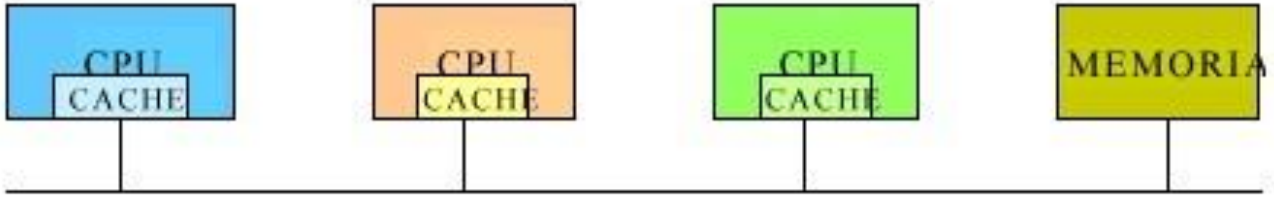
Para grabar el procedimiento es similar.

Modelos de Sistemas Distribuidos

Solo existe una memoria, la cual le da coherencia

Las modificaciones hechas por una cpu se reflejan de inmediato en las subsiguientes lecturas de la misma o de otra cpu.

Problema: el bus tiende a sobrecargarse y el rendimiento a disminuir drásticamente; la solución es añadir una memoria caché de alta velocidad entre la cpu y el bus.



Multiprocesadores con base en un bus.

Modelos de Sistemas Distribuidos

El esquema de multiprocesadores con base en buses es apropiado hasta aproximadamente 64 procesadores, para superar esta cifra es necesario un método distinto de conexión entre procesadores (cpu) y memoria. Una posibilidad es dividir la memoria en módulos y conectarlos a las cpu con un “conmutador de cruceta” (**cross-bar switch**)

Cada cpu y cada memoria tiene una conexión que sale de él. En cada intersección está un “conmutador del punto de cruce” (crosspoint switch) electrónico que el HW puede abrir y cerrar.

Cuando una cpu desea tener acceso a una memoria particular, el conmutador del punto de cruce que los conecta se cierra momentáneamente.

Modelos de Sistemas Distribuidos

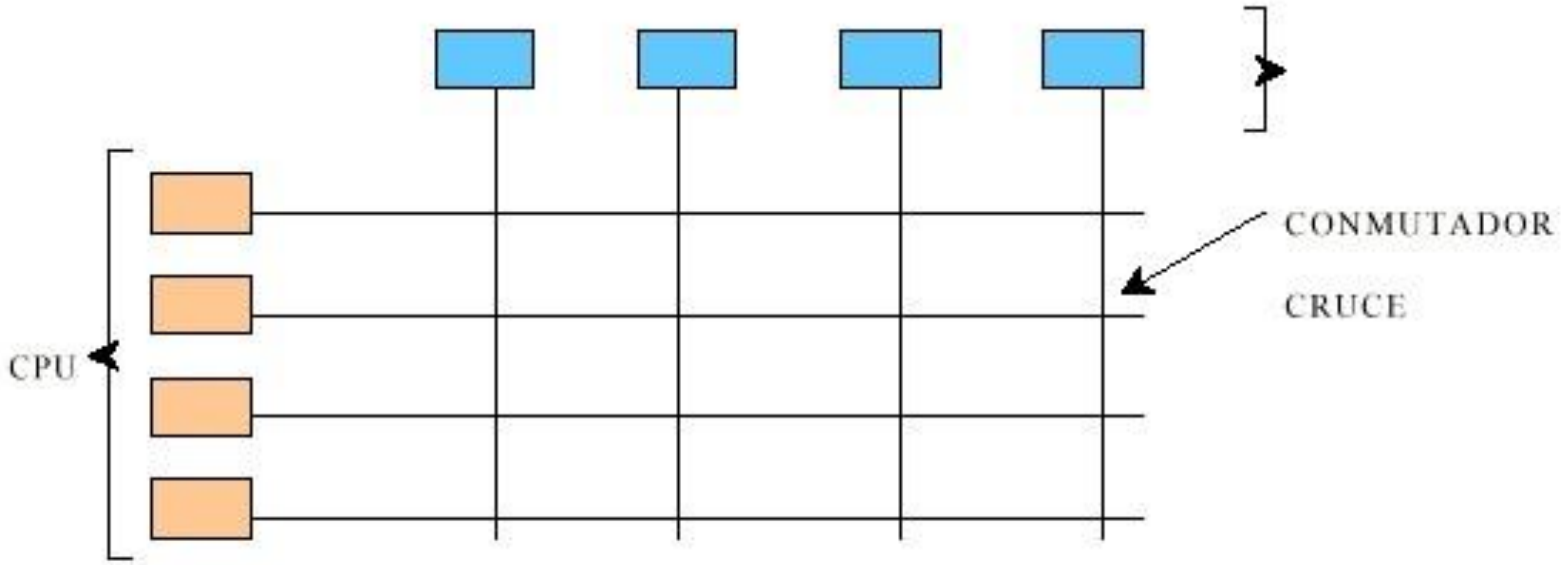
Multiprocesadores con conmutador

Si el caché es lo bastante grande la “tasa de encuentros” será alta y la cantidad de tráfico en el bus por cada cpu disminuirá notablemente. Permite incrementar el número de cpu. El problema es la “**incoherencia de la memoria**”.

Una solución es diseñar las caché de tal forma que cuando una palabra sea escrita al caché, también sea escrita a la memoria. A esto se denomina “**caché de escritura**”.

También los cachés pueden monitorear constante el bus, cada vez que un caché observa una escritura a una dirección de memoria presente en él, puede eliminar ese dato o actualizarlo en el caché con el nuevo valor, se denominan “**cachés monitores**”.

Modelos de Sistemas Distribuidos



Conmutador de cruceta.

Modelos de Sistemas Distribuidos

El diseño con cachés monitores y de escritura es coherente e invisible en su funcionamiento.

La virtud del conmutador de cruceta es que muchas cpu pueden tener acceso a la memoria al mismo tiempo; aunque no a la misma memoria simultáneamente.

Lo negativo de este esquema es el alto número de conmutadores: para “n” cpu y “n” memorias se necesitan “n” x “n” conmutadores.

El número de conmutadores del esquema puede resultar costoso.

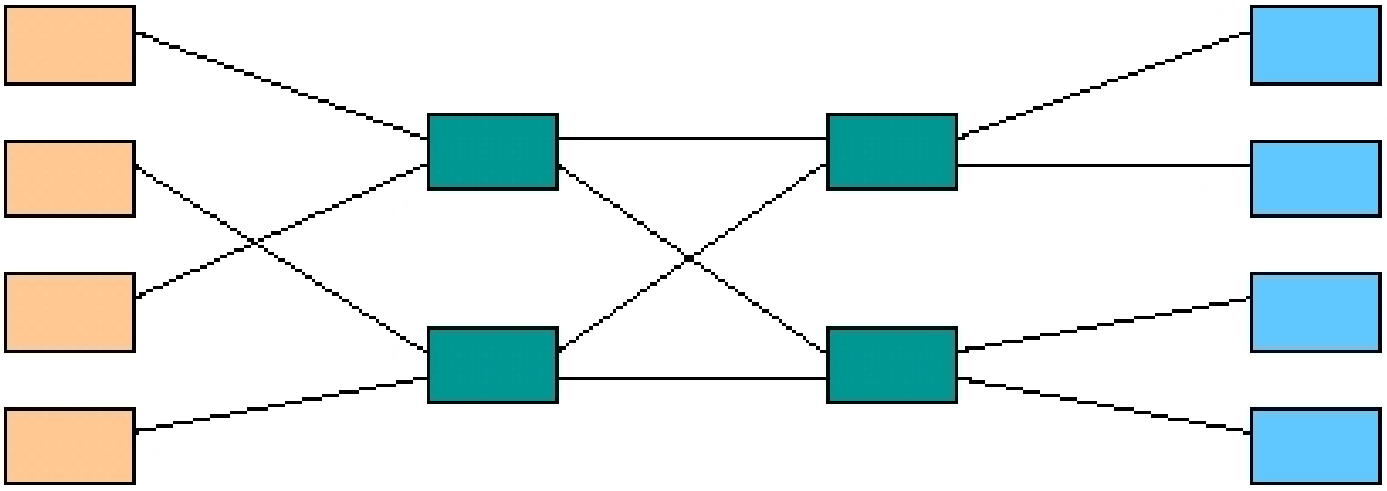
Modelos de Sistemas Distribuidos

Otros esquemas precisan menos conmutadores, por ej., la “red omega”posee conmutadores 2 x 2. Cada uno tiene 2 entradas y 2 salidas y cada conmutador. puede dirigir cualquiera de las entradas en cualquiera de las salidas.

Eligiendo los estados adecuados de los conmutadores, cada cpu podrá tener acceso a cada memoria.

Para “n” cpu y “n” memorias se precisan “n” etapas de conmutación. Un **problema** importante en la red omega es el **retraso**

Modelos de Sistemas Distribuidos



Red omega de conmutación.

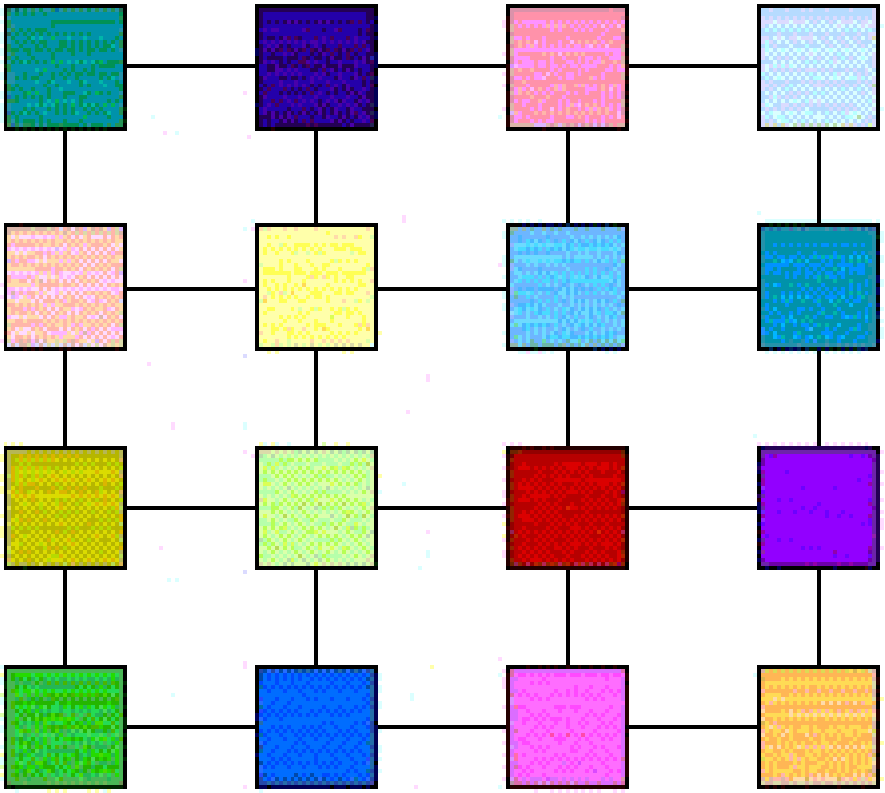
Modelos de Sistemas Distribuidos

Multicomputadoras con Conmutador

Multicomputadora: consta de estaciones de trabajo en una LAN. Cada cpu tiene acceso directo y exclusivo a su propia memoria particular. Existen diversas topologías, las más comunes son la retícula y el hipercubo.

Reticula: son fáciles de comprender, se basan en las tarjetas de circuitos impresos. Se adecúan a problemas con una naturaleza bidimensional inherente (teoría de gráficas, visión artificial, etc.

Modelos de Sistemas Distribuidos



Retícula.

Modelos de Sistemas Distribuidos

Hipercubo de dimensión 4

Se puede considerar como dos cubos, cada uno de ellos con 8 vértices y 12 aristas. Cada vértice es una cpu. Cada arista es una conexión entre 2 cpu. Se conectan los vértices correspondientes de cada uno de los cubos.

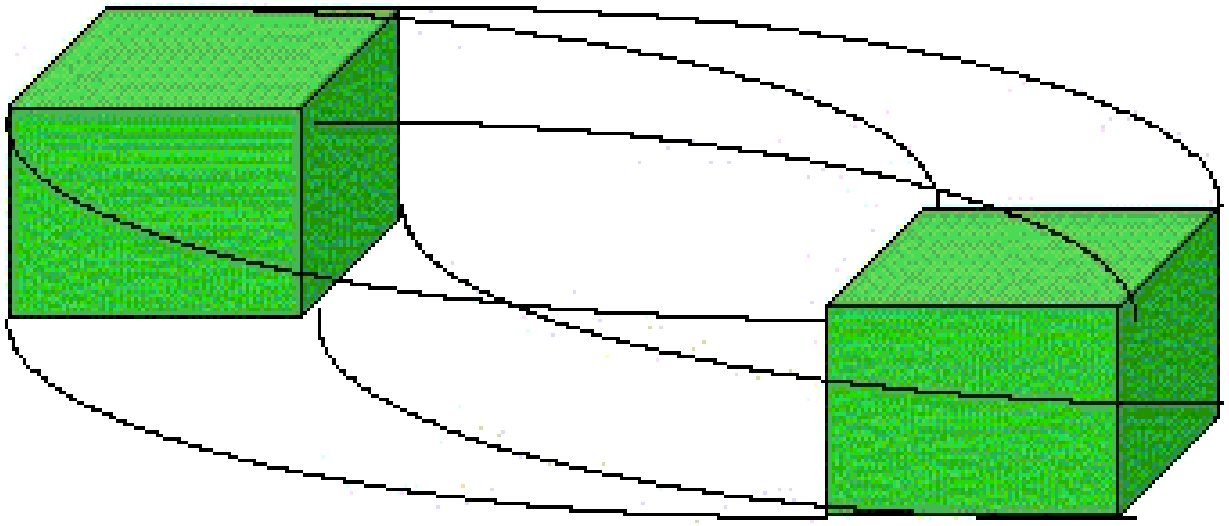
Algunos inconvenientes

Sólo se conectan los procesadores vecinos más cercanos:

Muchos mensajes deben realizar varios saltos antes de llegar a su destino.

La trayectoria más grande crece en forma logarítmica con el tamaño: en la retícula crece como la raíz cuadrada del número de cpu.

Modelos de Sistemas Distribuidos



Hipercubo de dimensión 4.

Modelos de Sistemas Distribuidos

Multicomputadoras con Base en Buses

Es un esquema sin memoria compartida. Cada cpu tiene una conexión directa con su propia memoria local.

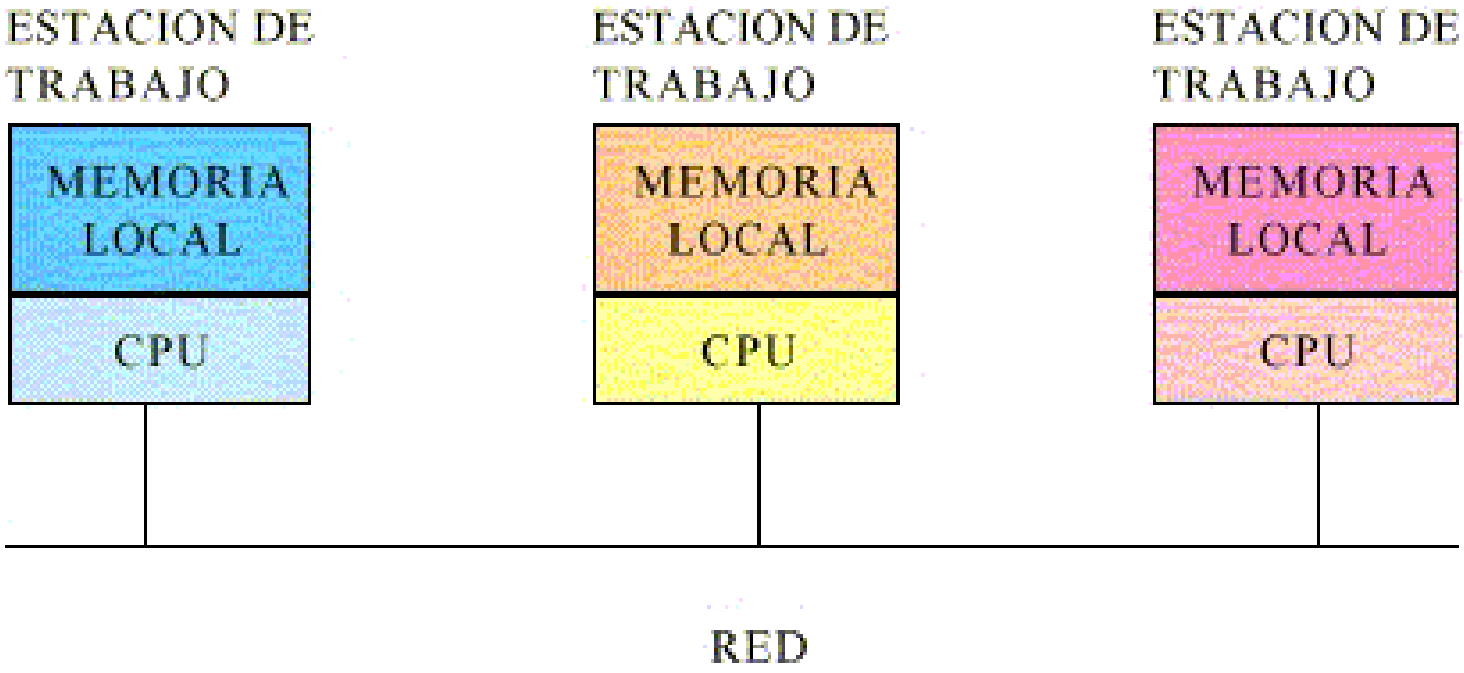
Un problema importante es la forma en que las cpu se comunican entre sí.

El tráfico es sólo entre una cpu y otra; el volumen de tráfico será varios órdenes de magnitud menor que si se utilizara la red de interconexión para el tráfico cpu - memoria.

Topológicamente es un esquema similar al del multiprocesador basado en un bus.

Consiste generalmente en una colección de estaciones de trabajo en una LAN (red de área local).

Modelos de Sistemas Distribuidos



Conceptos de Software

El software *débilmente acoplado* de un SD

Permite que las *máquinas y usuarios sean independientes entre sí* en lo fundamental.

Facilita que interactúen en cierto grado cuando sea necesario.

Los equipos individuales se distinguen fácilmente.

Combinando los distintos tipos de *hardware distribuido con software distribuido* se logran distintas soluciones p.e. Sistemas Operativos de Red

Conceptos de Software

El usuario tiene una ***estación de trabajo*** *y* su propio S. O.

La mayoría de los requerimientos se resuelven localmente.

Es posible que un usuario se conecte de manera remota con otra estación de trabajo por medio *"**login remoto**"*..

Los comandos se envían a la máquina remota y la salida se muestra en la pantalla local. Para alternar con otra máquina, primero hay que desconectarse.

Las redes también disponen de copiado remoto de archivos de una máquina a otra, necesita que el usuario conozca: la posición de todos los archivos, y el sitio donde se ejecutan todos los comandos.

Necesitará un ***sistema de archivos global compartido***, accesible desde todas las estaciones de trabajo

Conceptos de Software

Servidores de archivos:

Aceptan solicitudes de los programas de usuarios

Los ***programas*** se ejecutan en las máquinas **clientes**, los pedidos se examinan, se ejecutan y la respuesta se envía de regreso; generalmente tienen un *sistema jerárquico de archivos.*

Las estaciones de trabajo pueden *importar* o *montar* estos sistemas de archivos (SA): incrementando sus *sistemas de archivos locales o* montando los servidores en lugares diferentes de sus respectivos sistemas de archivos.

Las rutas de acceso a un determinado archivo pueden ser *diferentes* para las distintas estaciones, vision diferente por los clientes del SA, el nombre del archivo depende del lugar desde el cual se tiene acceso, y de la configuración del SA

Conceptos de Software

El S. O. de este tipo de ambiente debe:

- controlar las estaciones de trabajo en lo individual,
- controlar a los servidores de archivo,
- encargarse de la comunicación entre los servidores.

No es necesario ejecutar el mismo S. O.

Si los clientes y los servidores ejecutan diversos S. O., como mínimo deben *coincidir en el* ***formato y significado*** *de todos los mensajes* que podrían intercambiar.

A esto se llama "**sistema operativo de red**": en donde cada máquina tiene un alto grado de *autonomía y existen pocos requisitos* a lo largo de todo el sistema.

NFS Network File System

Surgió para UNIX pero se amplió a otros S. O.

Soporta *sistemas heterogéneos,* por ej.: clientes de MS - DOS que hagan uso de servidores UNIX.

•Los equipos pueden ser también de *hardware heterogéneo.*

Los aspectos más interesantes son los relacionados con:

- *La arquitectura.*
- *El protocolo.*
- *La implantación.*

NFS Network File System

NFS debe permitir que las computadoras compartan un Sistema de Archivos Común. Las máquinas pueden estar en la misma LAN, o en una WAN.

NFS permite que cada máquina sea un **cliente y un servidor** al mismo tiempo. Cada servidor de NFS exporta uno o varios de sus directorios (y subdirectorios dependientes) para el acceso por parte de clientes remotos.

Los clientes tienen acceso a los directorios exportados mediante el montaje. Cuando un cliente monta un directorio (remoto), este se convierte en parte de su jerarquía de directorios. Si dos o más clientes montan el mismo directorio **se pueden comunicar al compartir archivos** en sus directorios comunes.

Protocolos de NFS

Objetivos es soportar un sistema heterogéneo en donde los clientes y servidores podrían ejecutar distintos S.O. en hardware diverso, por ello es esencial que la interfaz entre los clientes y los servidores esté bien definida. NFS logra este objetivo definiendo dos “protocolos cliente - servidor”. Un “**protocolo**” es un conjunto de: ***Solicitudes que envían*** los clientes a los servidores y ***Respuestas que envían*** los servidores de regreso a los clientes.

Un protocolo maneja el montaje y otro es para el acceso a los directorios y archivos. Los clientes pueden enviar mensajes a los servidores para el manejo de los directorios y la lectura o escritura de archivos y tener acceso a los atributos de archivo, tales como su modo, tamaño y fecha de la última modificación.

Protocolos de NFS

NFS soporta “servidores sin estado”, ya que no mantienen la información de estado relativa a los archivos abiertos y si un servidor falla y arranca rápidamente, no se pierde información acerca de los archivos abiertos y los programas cliente no fallan.

NFS utiliza el esquema de protección de UNIX, con los bits “rwx”. Puede utilizar la criptografía de claves públicas para dar validez al cliente y el servidor en cada solicitud y respuesta. Las claves utilizadas para la autentificación, así como otra información, están contenidas en el NIS (Servicio de Información de la Red).

La implantación del código del cliente y el servidor es independiente de los protocolos NFS.

Protocolos de NFS

Implementación del tipo Sun, consta de tres capas. La capa superior es la de llamadas al sistema:

1. Maneja las llamadas del tipo open, read y close.

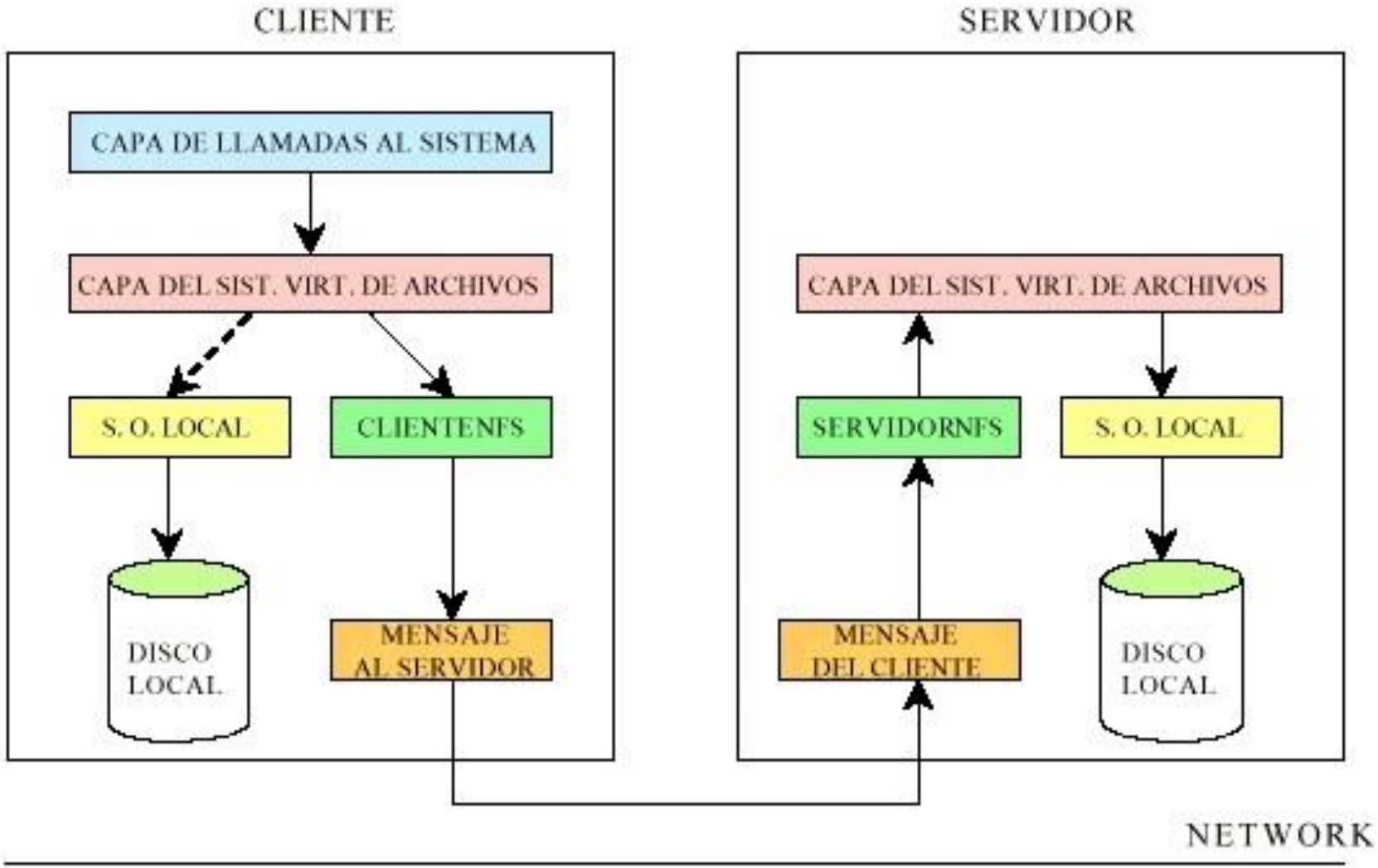
2. Analiza la llamada y verifica los parámetros.

3.Llama a la segunda capa, la capa del sistema virtual de archivos VFS.

La capa VFS mantiene una tabla con una entrada por cada archivo abierto que es similar a la tabla de nodos para los archivos abiertos en UNIX. La capa VFS tiene una entrada por cada archivo abierto, se la llama nodo-v (nodo-i virtual).

Los nodos-v se utilizan para indicar si el archivo es local o remoto. Para los archivos remotos, poseen la información suficiente como para tener acceso a ellos.

Estructura de capas de NFS



Sistemas de Multiprocesador con Tiempo Compartido

Los multiprocesadores operan como un sistema de tiempo compartido, pero con varias cpu en vez de una sola. Externamente un multiprocesador con 32 cpu de 3 MIPS actúa de manera muy parecida a una sola cpu de 96 MIPS.

Se corresponde con la imagen de un único sistema. La característica clave es la existencia de una sola cola para ejecución.

Los programas de los procesos están en la memoria compartida, también el S. O.

El planificador (de procesos) del S. O. se ejecuta como una "***región crítica***", con ello se evita que dos cpu elijan el mismo proceso para su ejecución inmediata.

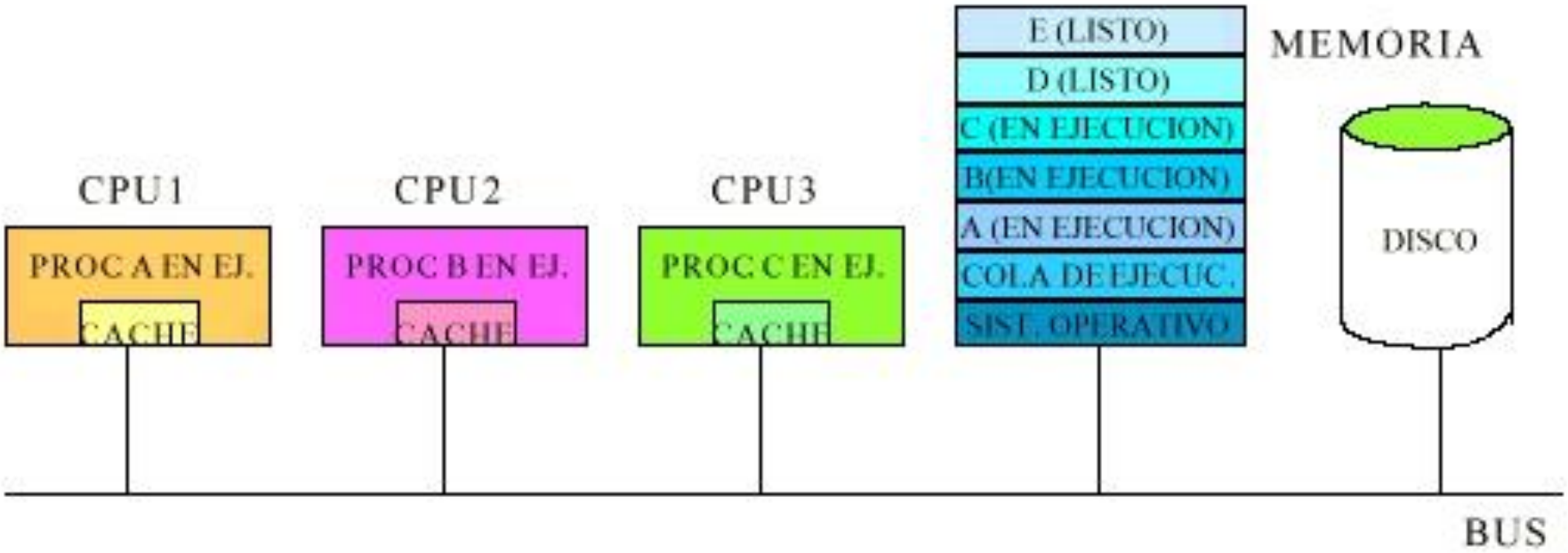
Sistemas de Multiprocesador con Tiempo Compartido

Ninguna cpu tiene memoria local, es decir que todos los programas se almacenan en la memoria global compartida.

Si todas las cpu están inactivas en espera de E/S y un proceso está listo para su Ejecución, se asigna a la cpu que se utilizó por última vez (para ese proceso), la hipótesis es que ningún otro proceso utilizó esa cpu desde entonces (hipótesis de Vaswani y Zahorjan).

Si un proceso se bloquea en espera de E/S en un multiprocesador, el S. O. puede suspenderlo o dejarlo en "espera ocupada", si se agota el tiempo de espera y no ha finalizado la E/S, se realiza una conmutación de procesos.

Sistemas de Multiprocesador con Tiempo Compartido



Un multiprocesador con una sola cola de ejecución.

Conclusiones

Sistema Débilmente Acoplado + Hardware Débilmente Acoplado = **NFS**.

Sistema Fuertemente Acoplado + Hardware Débilmente Acoplado = **Multicomputadoras**.

Sistema Fuertemente Acoplado + Hardware Fuertemente Acoplado = **Multiprocesadores** con tiempo compartido.

Aspectos del Diseño (Sist. Operativo Dist.)

Transparencia

De **localización**: los usuarios no pueden indicar la localización de los recursos.

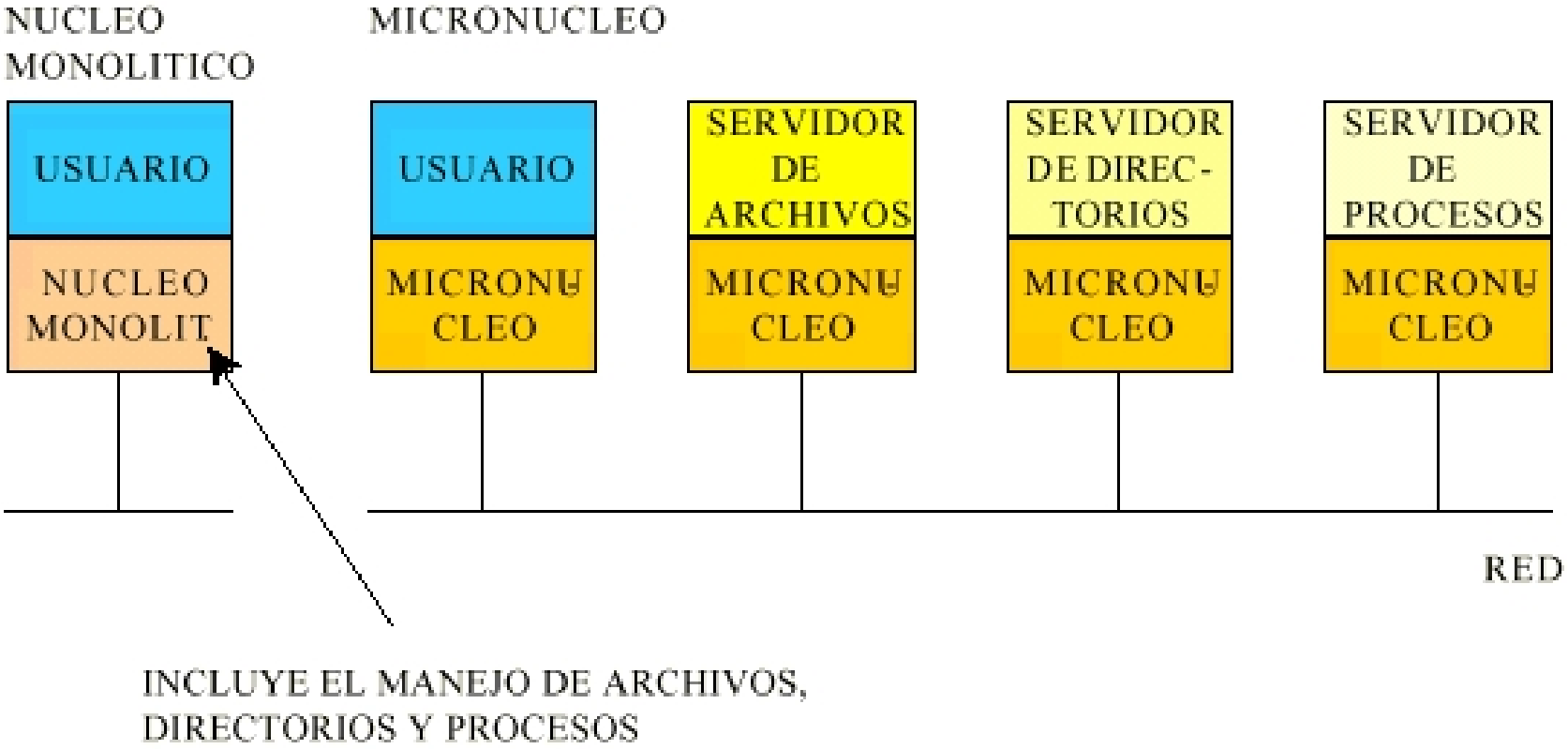
De **migración**: los recursos se pueden mover a voluntad sin cambiar sus nombres.

De **réplica**: los usuarios no pueden indicar el número de copias existentes.

De **concurrencia**: varios usuarios pueden compartir recursos de manera automática.

De **paralelismo**: las actividades pueden ocurrir en paralelo sin el conocimiento de los usuarios.

Aspectos del Diseño (Sist. Operativo Dist.)



Esquema de núcleo monolítico y de micronúcleo.

Aspectos del Diseño (Sist. Operativo Dist.)

Flexibilidad

Existen dos escuelas de pensamiento en cuanto a la estructura de los sistemas distribuidos:

Núcleo monolítico: Cada máquina debe ejecutar un núcleo tradicional que proporcione la mayoría de los servicios.

Micronúcleo (microkernel): el núcleo debe proporcionar lo menos posible. El grueso de los servicios del S. O. se debe obtener a partir de los servidores al nivel usuario.

Confiabilidad

Un importante objetivo de los sistemas distribuidos es que si una máquina falla, alguna otra debe encargarse del trabajo.

Aspectos del Diseño (Sist. Operativo Dist.)

Un aspecto de la confiabilidad es la disponibilidad, que se refiere a la fracción de tiempo en que se puede utilizar el sistema.

La disponibilidad se mejora mediante un diseño que no exija el funcionamiento simultáneo de un número sustancial de componentes críticos y la redundancia, es decir la duplicidad de componentes clave del hardware y del software.

Otro aspecto es la seguridad, significa que los archivos y otros recursos deben ser protegidos contra el uso no autorizado. también relacionado con la confiabilidad es la tolerancia a fallas, según la cual las fallas se deben ocultar brindando una recuperación transparente para el usuario, aunque haya cierta degradación de la performance.

Aspectos del Diseño (Sist. Operativo Dist.)

Desempeño

Algunas métricas del desempeño son el tiempo de respuesta, rendimiento (número de trabajos por hora), y uso del sistema y cantidad consumida de la capacidad de la red.

Se requiere el uso de protocolos de comunicaciones en los extremos (procesadores) que intervienen en la comunicación, con lo que se incrementa el consumo de ciclos de procesador. Para optimizar el desempeño frecuentemente hay que minimizar el número de mensajes y centralizar el trabajo en una sola máquina, prestando atención al tamaño de todos los cálculos.

Aspectos del Diseño (Sist. Operativo Dist.)

Escalabilidad

Existen cuellos de botella potenciales que se debe intentar evitar en los sistemas distribuidos de gran escala, algunas alternativas para evitar estos embotellamientos se deben utilizar:

- Componentes centralizados (Ej.: un solo servidor de correo para todos los usuarios.)
- Tablas centralizadas (Ej.: un único directorio telefónico en línea).
- Algoritmos centralizados(Ej.: realización de un ruteo con base en la información completa.)

Comunicación en los Sistemas Distribuidos

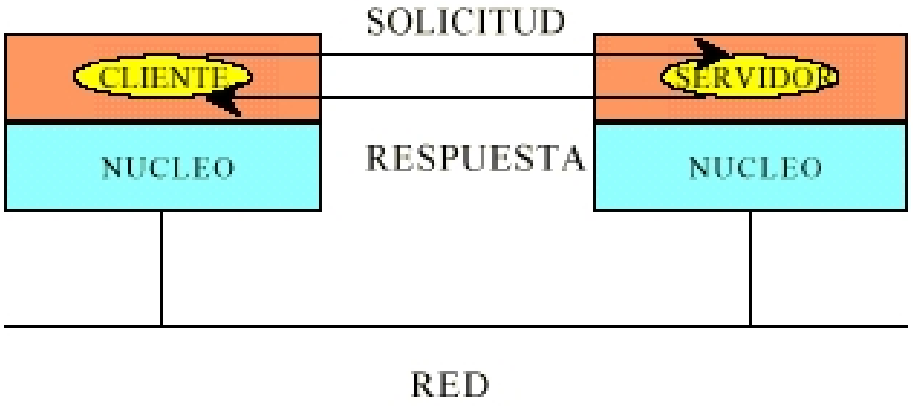
En un **Monoprocesador** la comunicación supone la existencia de la memoria compartida.

En un **S.D**. no existe la memoria compartida y por ello los procesos deben apegarse a reglas conocidas como protocolos.

Estos protocolos toman frecuentemente la forma de varias capas y cada capa tiene sus propias metas y reglas. Los mensajes se intercambian de diversas formas, existiendo muchas opciones de diseño al respecto; una importante opción es la "**llamada a un procedimiento remoto**".

También hay que considerar las posibilidades de comunicación entre grupos de procesos, no solo entre dos procesos.

Comunicación en los Sistemas Distribuidos



ESQUEMA DE LA SENCILLEZ DEL MODELO C - S

NIVELES DE PROTOCOLOS
NECESARIOS SI TODAS LAS
MAQUINAS FUESEN IDENTICAS

Modelo cliente - servidor.

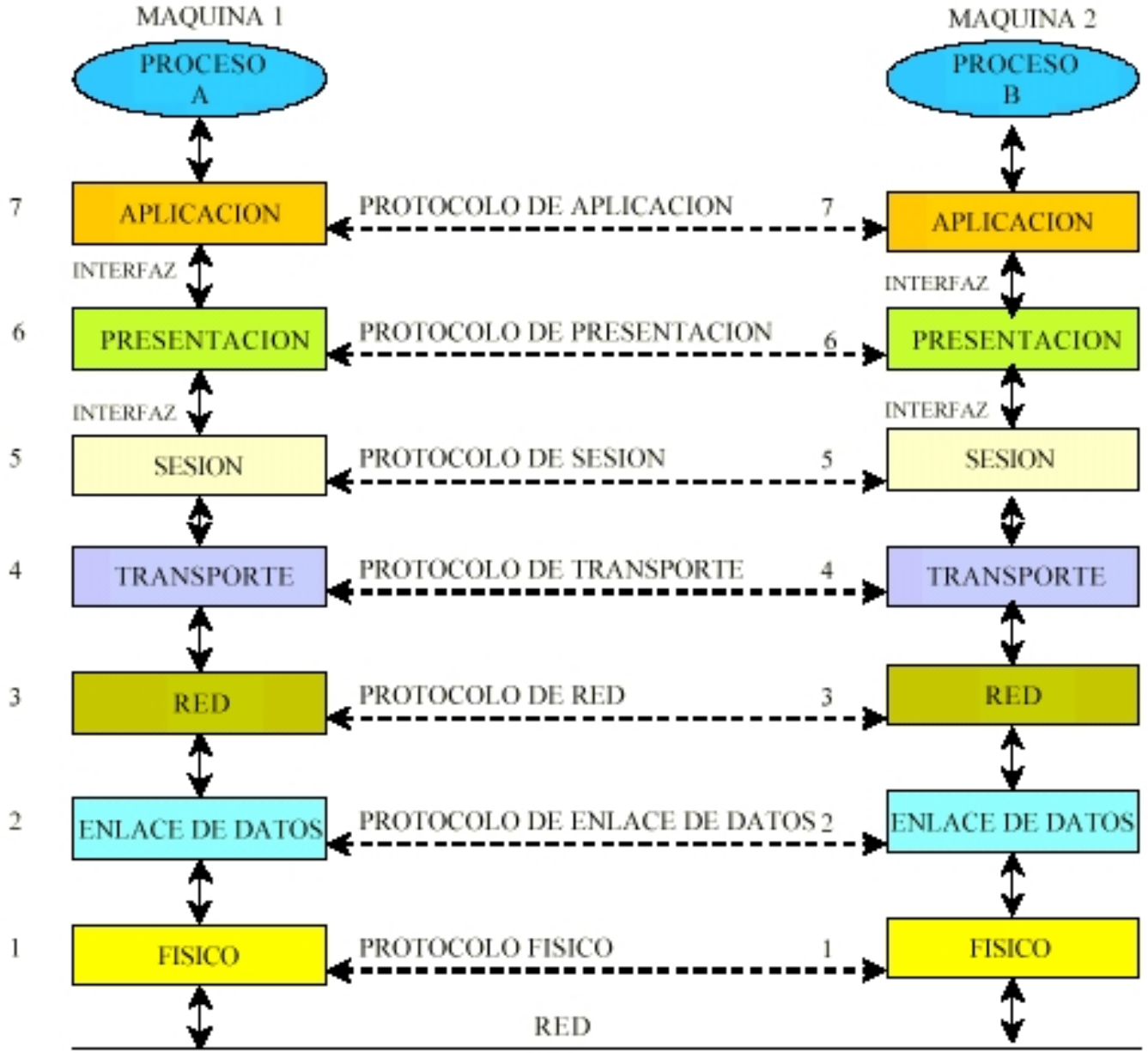
Comunicación en los Sistemas Distribuidos

Debido a la ausencia de memoria compartida, toda ***la comunicación en los sistemas distribuidos se basa en la transferencia de mensajes***; cuando el proceso "A" quiere comunicarse con el proceso "B", construye un mensaje en su propio espacio de direcciones.

Ejecuta una llamada al sistema para que el S. O. busque el mensaje y lo envíe a través de la red hacia "B".

La ISO (Organización Internacional de Estándares) desarrolló un modelo de referencia que:

- Identifica en forma clara los distintos niveles.
- Estandariza los nombres de los niveles.
- Señala que tipo de trabajo deber realizar cada nivel.



Capas, interfaces y protocolos en el modelo OSI.

Comunicación en SD

Migración de procesos: Cuando se presenta un proceso para su ejecución, no siempre se ejecuta en la instalación originadora, se distribuye por la red para nivelar la carga del trabajo, dividiendoló en varios subprocesos que puedan ejecutarse concurrentemente en instalaciones diferentes, existen dos técnicas que se usan:

Preferencia de hardware: El proceso tiene algunas características que hacen más apropiada su ejecución en algún procesador especializado.

Preferencia de software: El proceso puede requerir software que está disponible sólo en otra instalación, y no se pueda mover o sea más economico transferir el proceso.

Comunicación en SD

Llamadas a Procedimientos Remotos

Consideremos un usuario que solicita el acceso a un archivo remoto. Suponiendo que se localizo al servidor que contiene el archivo mediante el esquema de nominación apropiado, debe efectuarse la transferencia de datos para satisfacer la solicitud de acceso remoto del usuario.

Cada acceso es manejado por el servidor y ocasiona tráfico en la red; una lectura provoca él envió de un mensaje de solicitud al servidor y la respuesta al usuario con los datos solicitados.

Una de las formas más comunes de servicio remoto es el paradigma de llamadas a procedimientos remotos (RPC).

Comunicación en SD

RPC fue diseñada como una manera de abstraer el mecanismo de llamadas a procedimientos para usarse entre sistemas con conexiones en red.

Se trata de un entorno en donde los procesos se ejecutan en sistemas separados, debemos usar un esquema de comunicaciones basados en mensajes para proporcionar el servicio remoto.

Estos mensajes (paquetes de datos) se dirigen a un proceso de RPC que escucha un puerto del sistema remoto y contiene el nombre de un proceso que hay que ejecutar y los parámetros que deben pasarse a ese proceso.

Comunicación en SD

Luego se ejecuta el proceso según se solicitó y todo resultado se devuelve al solicitante en un mensaje. Un puerto es simplemente un numero incluido en el inicio de un paquete de mensajes.

Como se trata de una transferencia de mensajes por enlaces de comunicación poco confiables, los sistemas unen una marca de tiempo a cada mensaje. El servidor debe conservar un registro histórico de todas las marcas de tiempo de los mensajes que ya ha procesado, para que se ignore los mensajes que tengan una marca de tiempo ya registrada.

Comunicación en SD

Cliente/servidor: Sistemas es fuertemente acoplado ya que sus recursos son compartidos internamente.

Cuando se implementa esta clase de sistemas es necesarios estructurar el software en módulos manejables. Podemos considerar un módulo como una colección de instrucciones que realiza un servicio del sistema.

Servicios de los sistemas operativos: Ejecución de programas, Operaciones de E/S, Manipulación del sistema de archivos, Detección de errores.

Comunicación en SD

Para un mismo SO monolítico corriendo en diferentes plataformas, el núcleo independiente es exactamente el mismo, mientras que el dependiente debe re-escribirse.

El modelo de Cliente-Servidor es una forma de describir la iteración entre procesos, a través del paso de mensajes.

Kernel SO

Núcleo Dependiente: Interrupción Hardware, Bajo Nivel Memoria y Disco, Bajo Nivel dispositivos

Núcleo Independiente: Llamadas al Sist., Manejo Sist. Archivos, Planificación de Procesos

Comunicación en SD

El “modelo cliente - servidor” estructura al S. O. como un grupo de procesos en cooperación, llamados servidores, que ofrecen servicios a los usuarios y un grupo de procesos usuarios llamados clientes.

El “modelo cliente - servidor” se basa en un “protocolo solicitud / respuesta” es sencillo y sin conexión, no es complejo y orientado a la conexión como OSI o TCP/IP.

El cliente envía un mensaje de solicitud al servidor pidiendo cierto servicio y el servidor ejecuta el requerimiento , regresando los datos solicitados o un código de error si no pudo ejecutarlo correctamente.

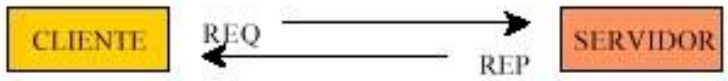
Comunicación en SD

Otra ventaja es que no se tiene que establecer una conexión sino hasta que ésta se utilice, la pila del protocolo es más corta y por lo tanto más eficiente. Si todas las máquinas fuesen idénticas sólo se necesitarían tres niveles de protocolos.

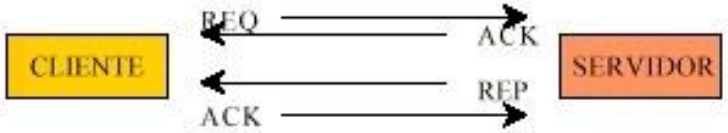
Para que un cliente pueda enviar un mensaje a un servidor, debe conocer la dirección (posición) un cambio de servidor obliga a cambiar los programas.

Un método de direccionamiento consiste en asignarle a cada proceso una única dirección que no contenga un número de máquina, y una forma es mediante un asignador centralizado de direcciones. a los procesos.

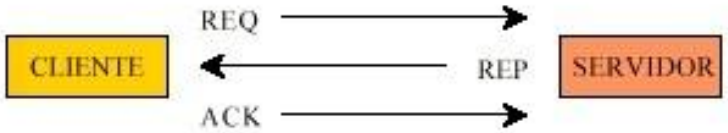
SOLICITUD / RESPUESTA DIRECTA, SIN RECONOCIMIENTOS



RECONOCIMIENTO DE CADA MENSAJE INDIVIDUAL



LA RESPUESTA ACTUA COMO RECONOCIMIENTO



VERIFICACION DE UN CLIENTE RESPECTO DE SI EL SERVIDOR ESTA ACTIVO

: Intercambio de paquetes en la comunicación cliente - servidor.

Comunicación en SD

Existe el problema, para el núcleo emisor, de saber a qué máquina enviar el mensaje en una LAN, el emisor puede transmitir un paquete especial de localización con la dirección del proceso destino, este paquete será recibido por todas las máquinas y todos los núcleos verifican si la dirección es la suya; si lo es, regresa un mensaje aquí estoy con su dirección en la red (número de máquina). El núcleo emisor utiliza esa dirección y la captura para evitar a posteriori una nueva búsqueda del servidor.

Es un esquema transparente, pero la transmisión provoca una carga adicional en el sistema que se puede evitar con una máquina adicional para la asociación de: los nombres de servicios, las direcciones de las máquinas.

Comunicación en SD

Conociendo la dirección del servidor, se le envía la solicitud del servicio requerido.

Las primitivas de transferencia de mensajes consideradas anteriormente se denominan ***primitivas de bloqueo o primitivas síncronas.***

El interés del S. O. está centrado en el manejo de los buffers y en la transmisión de los mensajes, desde el punto de vista de los lenguajes de programación el interés está centrado en el lenguaje de programación y sus facilidades de uso. Generalmente a las primitivas de envío se las conoce como ***send*** y a las de recepción como ***receive*** y ambas pueden ser con bloqueo o sin bloqueo.

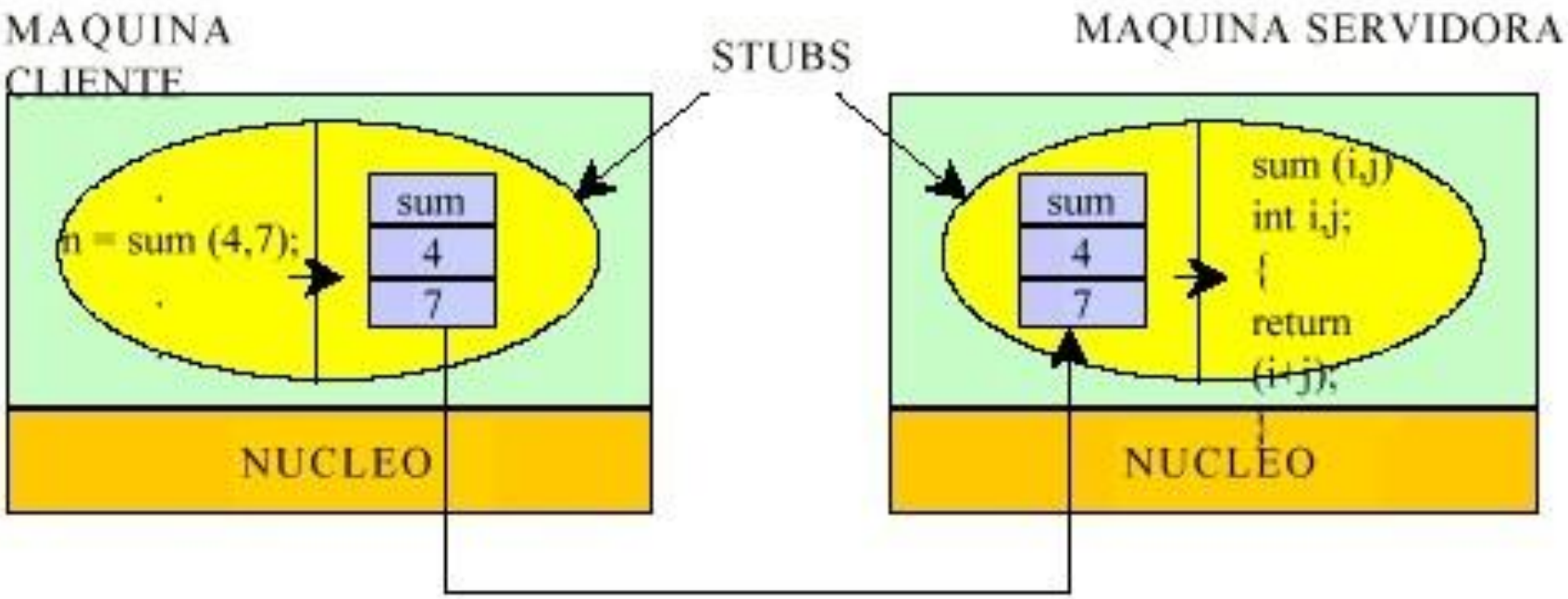
RPC (llamada a un procedimiento remoto)

Cuando un proceso en la máquina “A” llama a un procedimiento en la máquina “B” el proceso que realiza la llamada se *suspende*, la ejecución del procedimiento se realiza en “B” y la información se puede transportar de un lado al otro mediante los parámetros y puede regresar en el resultado del procedimiento.

El procedimiento que hace la llamada y el que la recibe se ejecutan en máquinas diferentes, es decir que utilizan espacios de direcciones distintos. La idea es que una llamada a un procedimiento remoto (RPC) se parezca lo más posible a una llamada local.

La RPC debe ser transparente.

RPC (llamada a un procedimiento remoto)



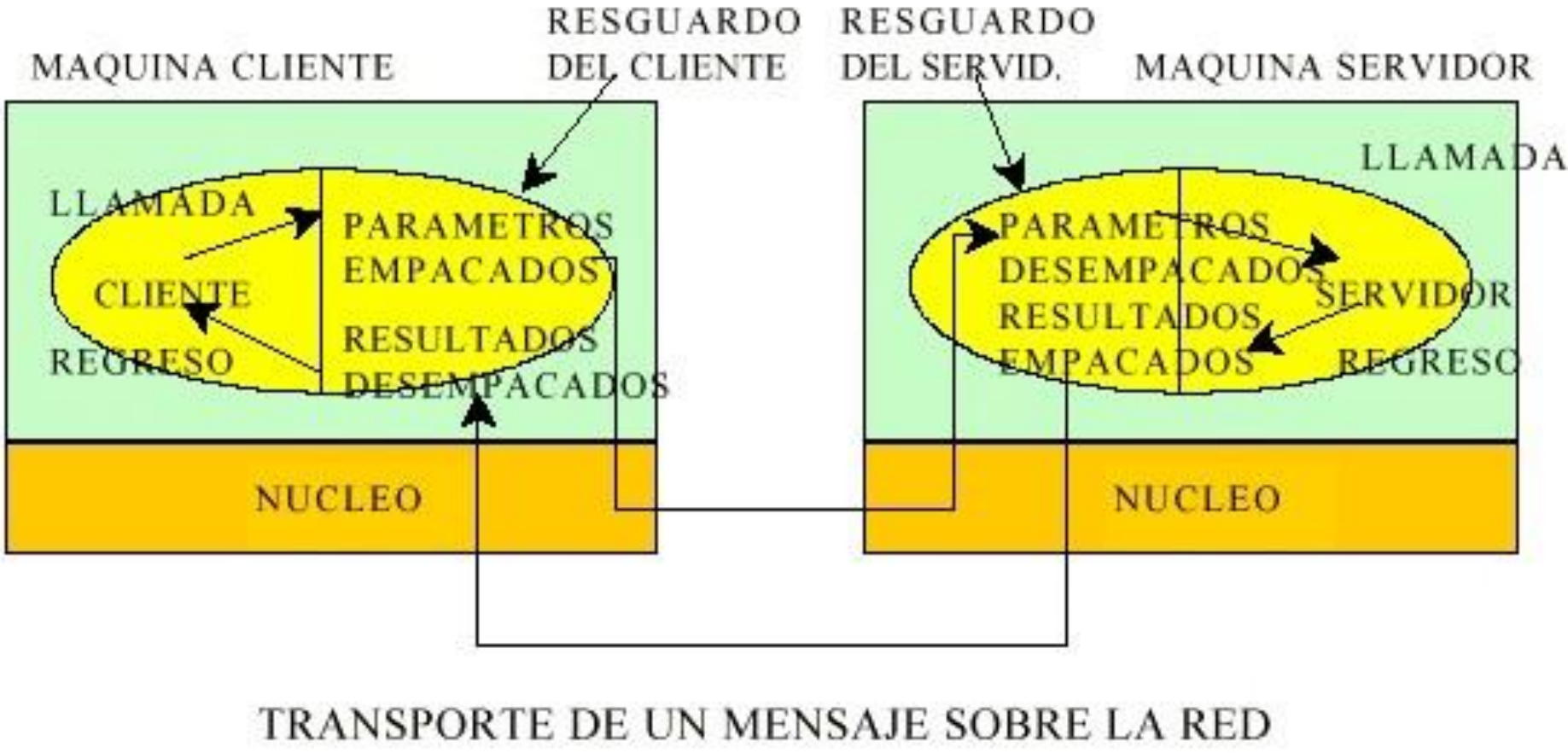
Ejemplo de cálculo remoto.

Transferencia de Parámetros en RPC

El empaquetamiento de parámetros en un mensaje se llama *ordenamiento de parámetros*. El mensaje también contiene el *nombre o número* del procedimiento por llamar; el servidor podría soportar varias llamadas y se le tiene que indicar cuál de ellas se necesita.

¿Cómo se debe representar la información en los mensajes? una forma es diseñar un estándar para los distintos tipos de datos y exigir a todos los emisores que conviertan sus representaciones internas a esta forma durante el ordenamiento. Habrá que conocer *la organización del mensaje y la identidad del cliente* asi como determinar de dónde provienen los procedimientos resguardo.

Transferencia de Parámetros en RPC



Transferencia de Parámetros en RPC

Es posible tener un *compilador* que lea las especificaciones del servidor y genere un resguardo del cliente que empaque sus parámetros en el formato estándar de los mensajes. Además que genere un resguardo del servidor, que los desempaque y llame al servidor.

Otro aspecto importante es cómo se transfieren los ***apuntadores***, pues un apuntador sólo tiene sentido dentro del espacio de direcciones del proceso en el que se utiliza. Una solución es copiar en el mensaje la estructura para la cual se utiliza el apuntador y enviarlo al servidor, en tal caso los cambios que realice el servidor mediante el apuntador afectarán directamente al buffer de mensajes en el resguardo del servidor.

Conexión Dinámica (Dynamic Binding) en RPC

Un método consiste en integrar dentro del código del cliente la dirección del servidor en la red, el problema es que resulta demasiado **rígido** si el servidor se desplaza, si se duplica o si cambia la interfaz, habría que localizar y volver a compilar los numerosos programas. Es por esto se usa una conexión dinámica para que concuerden los clientes y los servidores.

¿Cómo? Se especifica formalmente el servidor, indica el nombre del servidor, el número de versión y una lista de los procedimientos que proporciona. El servidor proporciona al conector su nombre, número de versión y un único identificador. El identificador generalmente tiene una longitud de 32 bits y un handle que se utiliza para localizarlo, este depende del sistema y puede ser una MAC, IP, X.500, etc.

Semántica de RPC en Presencia de Fallos

El objetivo de RPC es ocultar la comunicación al hacer que las llamadas a procedimientos remotos se parezcan a las llamadas locales. El problema se presenta cuando aparecen los errores, ya que las diferencias entre las llamadas locales y remotas no son tan fáciles de encubrir.

Se consideraran las siguientes situaciones:

–El cliente no puede localizar al servidor.

– Se pierde el mensaje de solicitud del cliente al servidor.

– Se pierde el mensaje de respuesta del servidor al cliente.

– El servidor falla antes de recibir una solicitud.

– El cliente falla después de enviar una solicitud.

El Cliente No Puede Localizar al Servidor

El servidor podría estar inactivo o podría estar utilizando una nueva versión de la interfaz y nuevos resguardos, que no serían compatibles con la interfaz y los resguardos del cliente.

En el servidor, cada uno de *los procedimientos regresa* un valor generalmente el "*1" indica un fallo*. También se suele utilizar una variable global de error a la que se asigna un valor que indica el tipo de error. Un tipo de error sería "no se pudo localizar al servidor". Otra posibilidad para el tratamiento de los errores es mediante una excepción provocada por el error se codifican procedimientos especiales que son llamados ante errores específicos y el problema es que se puede destruir la transparencia deseada, ya que se dificulta mantener la similitud entre procedimientos locales y remotos.

Pérdida de Mensajes de Solicitud

El núcleo (kernel) debe inicializar un cronómetro al enviar la solicitud; si el tiempo se termina antes de que regrese una respuesta o reconocimiento, el núcleo vuelve a enviar el mensaje.

Si el mensaje realmente se perdió, el servidor no podrá indicar la diferencia entre la retransmisión y el original y todo funcionará bien.

Si el número de mensajes perdidos supera cierto límite, el núcleo puede asumir que el servidor está inactivo y se regresa a la situación “no se pudo localizar al servidor”.

Pérdida de Mensajes de Respuesta

Este tipo de error genera mayores problemas que la pérdida de solicitudes.Se utiliza un *cronómetro*, si no llega una respuesta en un período determinado, se debe volver a enviar la solicitud. El problema es que el núcleo del cliente no está seguro de la razón por la que no hubo respuesta. Ciertas operaciones se pueden repetir con seguridad tantas veces como sea necesario sin que ocurran daños, otras no (transf.bancarias).

Solución: el núcleo del cliente asigna a *cada solicitud un número secuencial*, el núcleo del servidor mantiene un registro del número secuencial de recepción más reciente de cada uno de los clientes que lo utilicen. El servidor podrá discriminar entre una solicitud original y una retransmisión, pudiendo rechazar la ejecución de cualquier solicitud por segunda vez. Una protección adicional es tener un bit en el encabezado del mensaje para distinguir las solicitudes de las retransmisiones.

Fallos del Servidor

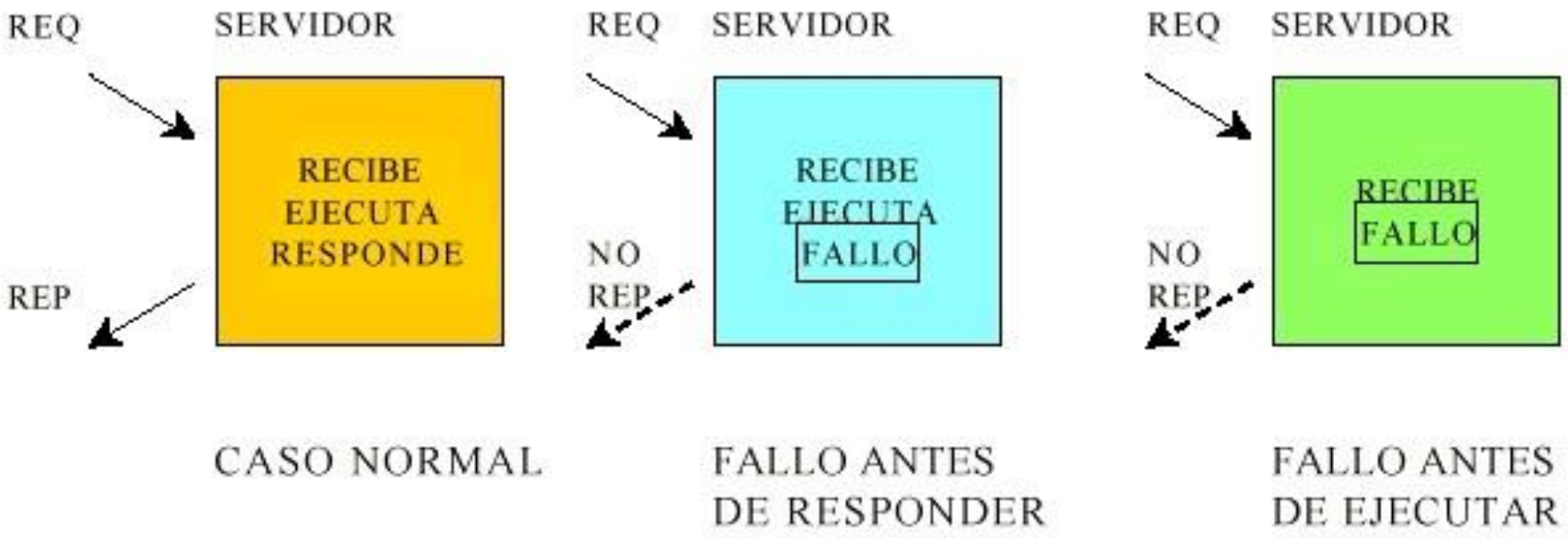
El núcleo del cliente no puede decidir si se ha presentado la segunda o la tercera situación.

Posibles soluciones:

Semántica al menos una: esperar hasta que el servidor vuelva a arrancar (o se reconecte a un nuevo servidor) e intente realizar de nuevo la operación. Mantiene el intento hasta recibir una respuesta para dársela al cliente. Garantiza que la RPC se ha realizado al menos una vez, pero es posible que se realice más veces.

Semántica a lo más una: no se reintenta y se informa del fallo. Garantiza que la RPC se realiza a lo más una vez, pero es posible que no se realice ni una sola vez.

Semántica de RPC en Presencia de Fallos



Situaciones posibles de fallos en el servidor.

Fallos del Servidor

Posibles soluciones:

Semántica de no garantizar nada: cuando un servidor falla, el cliente no obtiene ayuda o alguna promesa. La RPC se puede realizar en cualquier lugar, un número de veces que va desde “0” hasta “n” y resulta fácil de implantar.

Semántica de exactamente una: Es la solución deseable pero generalmente no existe forma de garantizar esto. El procedimiento de recuperación depende totalmente del momento en que ocurre el fallo. El cliente no tiene forma de descubrir ese instante.

En los sistemas con un único procesador el fallo de un servidor implica un fallo del cliente y la recuperación no es ni posible ni necesaria. En sistemas distribuidos es posible y necesario realizar cierta acción.

Fallos del Cliente

La cuestión es qué ocurre si un cliente envía una solicitud a un servidor y falla antes de que el servidor responda., se genera una solicitud de trabajo que al fallar el cliente ya nadie espera; se dice que se tiene un cómputo huérfano. Los problemas asociados: 1) Desperdicio de ciclos de cpu, 2) Posible bloqueo de archivos, 3) Apropiación de recursos valiosos, 4) Posible confusión: cuando el cliente reinicia y efectúa de nuevo la RPC, o la respuesta del huérfano regresa inmediatamente luego.

Las soluciones a los cómputos huerfanos son las siguientes: se crea un registro que indica lo que va a hacer el resguardo del cliente antes de que emita la RPC, el registro se mantiene en disco, luego del rearranque se verifica el contenido del registro y se elimina el huérfano explícitamente.

Aspectos de la Implantación en RPC

El desempeño o performance es fundamental en los sistemas distribuidos. El desempeño depende de manera crítica de la velocidad de la comunicación y esta de la implantación.

Protocolos RPC: Se debe elegir entre un protocolo orientado a la conexión o un protocolo sin conexión.

Protocolos orientados a la conexión:

- Se establece una conexión entre cliente y servidor,
- Todo el tráfico en ambas direcciones utiliza esa conexión,
- Se maneja a un nivel inferior mediante el software que soporta la conexión,

Es útil para la WAN, y desventajoso en LAN

Aspectos de la Implantación en RPC

Protocolos sin conexión tienen características opuestas a las indicadas. La utilización del protocolo IP (o UDP, integrado a IP) posee las siguientes ventajas: ya fue diseñado ahorrando trabajo, se dispone de muchas implantaciones, los paquetes IP se pueden enviar y recibir por casi todos los sistemas UNIX, los paquetes IP y UDP se pueden transmitir en muchas de las redes existentes. ***Reconocimientos:*** Cuando los RPC son de gran tamaño deben dividirse en muchos paquetes pequeños, surge la cuestión del reconocimiento, los paquetes serán reconocidos grupalmente o individualmente.

Ejemplo: un cliente desea escribir un bloque de datos de 4k en un servidor de archivos, pero el sistema no puede manejar paquetes mayores de 1k.

Aspectos de la Implantación en RPC

Una estrategia de reconocimiento es el protocolo detenerse y esperar (***Stop-And-Wait Protocol***): el cliente envía el paquete 0 con el primer 1k, este espera un reconocimiento del servidor, el cliente envía el paquete 1 con el segundo 1k, etc. La pérdida de un paquete significa que no llegará su reconocimiento y habrá que retransmitirlo.

Otra estrategia es el protocolo de chorro (***Blast Protocol***): el cliente envía todos los paquetes tan pronto como puede y el servidor reconoce todo el mensaje al recibir todos los paquetes; no hay reconocimiento individual de paquetes. La pérdida de un paquete puede significar la retransmisión de todo el mensaje o la repetición selectiva de la transmisión del paquete dañado o perdido.

Aspectos de la Implantación en RPC

Otra consideración más importante que el control de *errores* es el control del flujo está relacionado con la *capacidad finita* de recepción de paquetes adyacentes por parte de los chips de *interfaz de red*. Cuando un paquete llega a un receptor que no lo puede aceptar se presenta un error de sobre ejecución (overrun error) y el paquete se pierde.

Con el protocolo detenerse y esperar no se presentan los errores de sobre ejecución. Con el protocolo de chorro pueden ocurrir errores de sobre ejecución.

Una solución consiste en que el emisor inserte un retraso entre los paquetes para darle tiempo al receptor para genere la interrupción correspondiente al paquete y vuelva a estar listo para recepción.

Aspectos de la Implantación en RPC

Un problema adicional consiste en la posible pérdida de paquetes de reconocimiento del cliente al servidor. Para el cliente, que recibió la respuesta a su requerimiento, todo habrá terminado correctamente y para el servidor habrá una respuesta no reconocida. Una solución es reconocer también a los paquetes de reconocimiento, lo cual agrega complejidad y costo adicional.

Otra solución es que *el servidor inicialice un cronómetro* al enviar la respuesta, ésta se descartará cuando ocurra lo siguiente:

–llegada del reconocimiento,

–expiración del tiempo,

–llegada de un nuevo requerimiento del cliente.

Aspectos de la Implantación en RPC

Ruta Crítica: es la serie de instrucciones que se ejecutan con cada RPC, es importante determinar en qué parte de la ruta crítica se ocupa la mayor parte del tiempo que demanda la RPC.

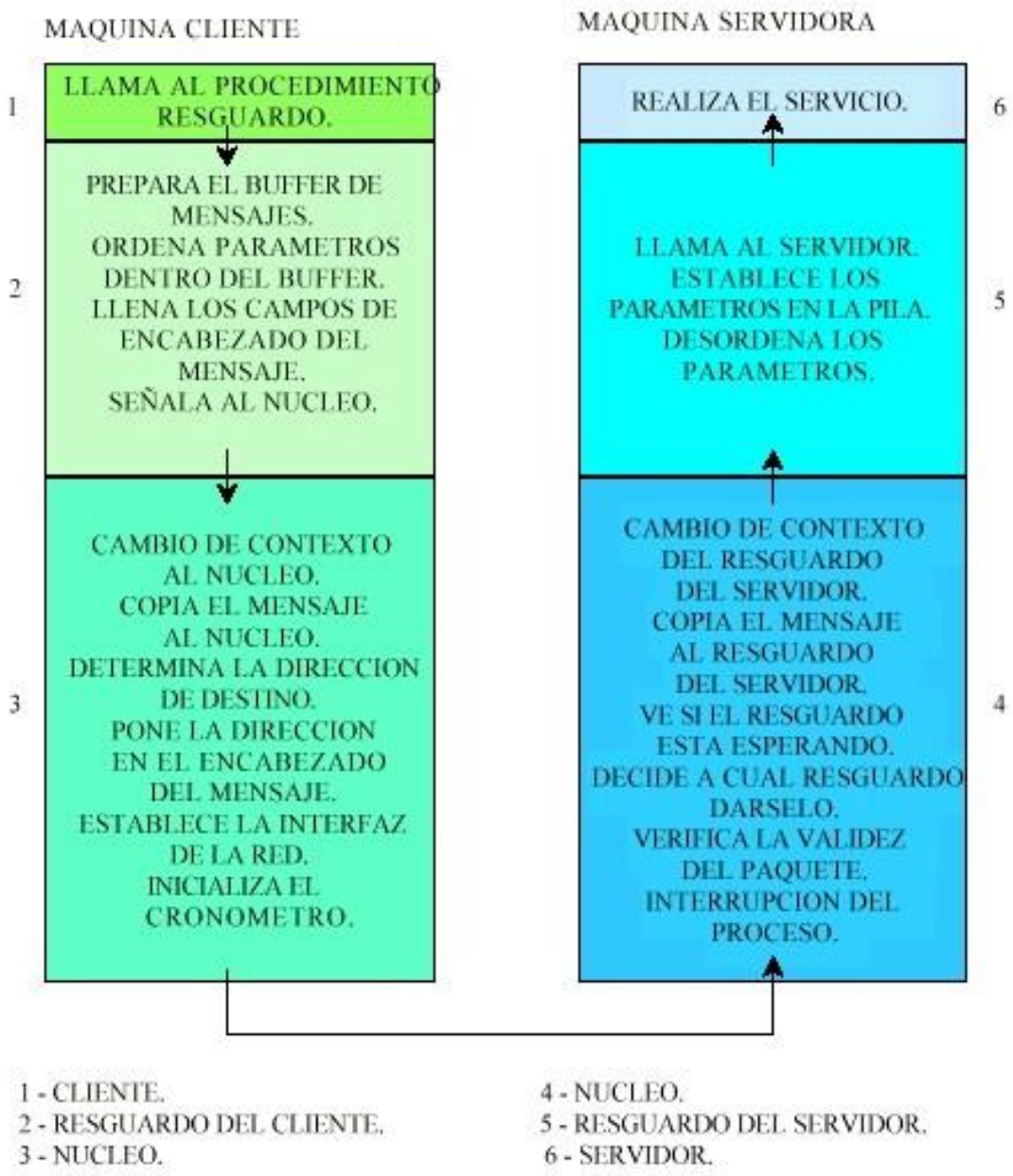
En RPC con transporte de 1k o más, la mayor parte del tiempo se ocupa en el tiempo de transmisión y en el tiempo que tarda el desplazamiento del paquete hacia adentro y afuera de la interfaz.

Es importante determinar en qué parte de la ruta crítica se ocupa la mayor parte del tiempo que demanda la RPC:

•Depende del tipo de RPC y de la cantidad de datos que se deben transportar.

Aspectos de la Implantación en RPC

- En RPC con *transporte mínimo* la mayor parte del tiempo se ocupa en:
  - El cambio de contexto al resguardo del servidor al llegar un paquete.
  - La rutina de servicio de interrupciones.
  - El movimiento del paquete a la interfaz de la red para su transmisión.
- En RPC con *transporte de 1k o más* la mayor parte del tiempo se ocupa en:
  - El tiempo de transmisión.
  - El tiempo que tarda el desplazamiento del paquete hacia adentro y afuera de la interfaz.



Ruta crítica del cliente al servidor.

Aspectos de la Implantación en RPC

Copiado La copia frecuentemente domina los tiempos de ejecución en RPC. El número de veces que se debe copiar un mensaje varía según el hardware, el software y el tipo de llamada.

Copias efectuadas: el núcleo del cliente copia el mensaje del resguardo del cliente en un buffer del núcleo para su transmisión; el núcleo copia el mensaje en software, a un buffer de hardware en la tarjeta de interfaz de la red.

El paquete se desplaza por la red hacia la tarjeta de interfaz de la máquina destino; cuando la interrupción correspondiente al paquete aparece en la máquina del servidor, el núcleo lo copia a un buffer del núcleo; el núcleo del servidor lo extrae del buffer de hardware; el mensaje, luego de ser inspeccionado, se copia al resguardo del servidor

Aspectos de la Implantación en RPC

Si la llamada transfiere como parámetro un arreglo de gran tamaño se agregan las siguientes copias: a la pila del cliente para la llamada del resguardo; de la pila al buffer de mensajes durante el ordenamiento dentro del resguardo del cliente; del mensaje recibido en el resguardo del servidor a la pila del servidor que antecede a la llamada al servidor.

El esquema anterior realiza 8 copias: si el tiempo de copia de una palabra de 32 bits es de 500 nseg, con 8 copias, cada palabra necesita 4 microseg., limita la velocidad máxima de transmisión de los datos a 1 mbyte / seg, independientemente de la velocidad de la red.

Aspectos de la Implantación en RPC

Manejo del Cronómetro

La mayoría de los protocolos inicializan un cronómetro cada vez que se envía un mensaje y se espera una respuesta. Si la respuesta no llega en el tiempo esperado se vuelve a transmitir el mensaje. Este proceso se puede repetir hasta un máximo previsto. El tiempo de cpu destinado al manejo de los cronómetros puede ser considerable.

El establecimiento de un cronómetro requiere construir una estructura de datos que especifique el momento en que el cronómetro debe detenerse y la acción a realizar cuando suceda.

La estructura de datos de un cronómetro se inserta en una lista de cronómetros pendientes cuando se inicializa el cronómetro y se retira de la lista cuando llega un reconocimiento antes de que termine el tiempo acordado.

Aspectos de la Implantación en RPC

Un valor de *lapso de expiración muy pequeño* hará que los cronómetros expiren con mucha frecuencia y se realicen muchas *retransmisiones innecesarias.*

Un valor de *lapso de expiración muy grande* hará que se demore en *retransmitir un paquete realmente perdido*.

Una alternativa al almacenamiento de los cronómetros en una tabla o lista ligada ordenada consiste en utilizar la tabla de procesos y cargar en ella un campo para su tiempo de expiración, si es que existe; rastrear periódicamente la tabla de procesos para comparar el valor de cada cronómetro con el tiempo actual. Este tipo de algoritmos se denominan algoritmos de barrido (sweep algorithms).

Memoria Compartida Distribuida (DSM)

La memoria compartida distribuida es una abstracción que se propone como alternativa a la comunicación por mensajes.

Memoria compartida basada en páginas: este esquema propone un espacio de direcciones de memoria virtual que integre la memoria de todas las computadoras del sistema, y su uso mediante paginación. Las páginas quedan restringidas a estar necesariamente en un único lugar. Cuando un programa intenta acceder a una posición virtual de memoria, se comprueba si esa página se encuentra de forma local. Si no se encuentra, se provoca un fallo de página, y el sistema operativo solicita la página al resto de las computadoras hasta que la petición llega a la computadora que tiene la página virtual solicitada en su memoria local.

Memoria Compartida Distribuida (DSM)

Memoria compartida basada en objetos

Una alternativa al uso de páginas es tomar el objeto como base de la transferencia de memoria. Aunque el control de la memoria resulta más complejo, el resultado es al mismo tiempo modular y flexible, y la sincronización y el acceso se pueden integrar limpiamente.

Otra de las restricciones de este modelo es que todos los accesos a los objetos compartidos han de realizarse mediante llamadas a los métodos de los objetos, con lo que no se admiten programas no modulares y se consideran incompatibles.

Memoria Compartida Distribuida (DSM)

Modelos de consistencia: La duplicidad de los bloques compartidos aumenta el rendimiento, pero produce un problema de consistencia entre las diferentes copias de la página en caso de una escritura. Si con cada escritura es necesario actualizar todas las copias, el envío de las páginas por la red provoca que el tiempo de espera aumente demasiado, convirtiendo este método en impracticable.

Para solucionar este problema se proponen diferentes modelos de consistencia, que establezcan un nivel aceptable de acercamiento tanto a la consistencia como al rendimiento. Nombramos algunos modelos de consistencia, del más fuerte al más débil: consistencia estricta, secuencial, causal, PRAM, del procesador, débil, de liberación y de entrada.

Sistema de Archivos

En un sistema distribuido es importante distinguir entre los conceptos de servicio de archivos y el servidor de archivos.

Servicio de archivos

Es la especificación de los servicios que el sistema de archivos ofrece a sus clientes.

Describe las primitivas disponibles, los parámetros que utilizan y las acciones que llevan a cabo.

Define precisamente el servicio con que pueden contar los clientes sin decir nada respecto de su implantación.

Sistema de Archivos

Servidor de archivos

Es un proceso que se ejecuta en alguna máquina y ayuda con la implantación del servicio de archivos. Puede haber uno o varios en un sistema.

Los clientes no deben ser conscientes de la forma de implantar el sistema de archivos: no precisan conocer el número de servidores de archivos, su posición o función; deberían ver al sistema distribuido de archivos como un sistema de archivos normal de monoprocesador.

Generalmente un servidor de archivos es un proceso del usuario (a veces del núcleo) que se ejecuta en una máquina.

Sistema de Archivos

Un sistema puede contener varios servidores de archivos, cada uno con un servicio distinto, por ejemplo un sistema con un servidor de archivos UNIX y otro servidor de archivos DOS, donde cada proceso usuario utilizará el servidor apropiado.

A diferencia de los sistemas de archivos clásicos, un sistema de archivos distribuido debe ser descentralizado, transparente y tolerante a fallos.

Sistema de Archivos

Transparencia

Es necesario que todas las computadoras lleven siempre y en todo momento una copia actualizada de la estructura de archivos y directorios para que puedan acceder a la localización física de los mismos.

Fallos del sistema

Que el sistema de archivos sea tolerante a fallos implica que el sistema debe guardar varias copias del mismo archivo en distintas computadoras para garantizar la disponibilidad en caso de fallo del servidor original.

Sistema de Archivos

Los componentes de un sistema distribuido de archivos son el verdadero servicio de archivos (realizan las operaciones en los archivos individuales: lectura, escritura, adición) y el servicio de directorios (crea y maneja directorios, añade y elimina archivos de los directorios, etc).

La Interfaz del Servicio de Archivos se encargará de la protección sobre el sistema en los accesos, utilizará técnicas similares a los sistemas monoprocesador. Administrando los tipos de accesos permitidos y la lista para control de acceso ya que se asocia a cada archivo una lista implícita o explícita de los usuarios que pueden tener acceso al archivo y de los tipos de acceso permitidos a cada uno de ellos.

Sistema de Archivos

Los servicios de archivos se clasifican de la siguiente forma:

Modelo carga / descarga: las principales operaciones son la lectura de un archivo y la escritura en un archivo, la lectura transfiere todo un archivo de uno de los servidores de archivos al cliente solicitante, la escritura transfiere en sentido contrario y los archivos se pueden almacenar en memoria o en un disco local.

Modelo de acceso remoto: el sistema de archivos se ejecuta con todas las funciones en los servidores y no en los clientes.

Sistema de Archivos

La Interfaz del Servidor de Directorios

Proporciona operaciones para crear y eliminar directorios, nombrar y cambiar el nombre de archivos y mover archivos de un directorio a otro.

Se utiliza un sistema jerárquico de archivos, representado por un árbol de directorios.

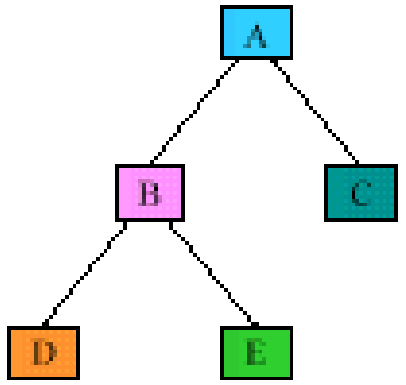
En ciertos sistemas es posible crear enlaces o apuntadores a un directorio arbitrario, se pueden colocar en cualquier directorio y se pueden construir gráficas de directorios.

En una jerarquía con estructura de árbol sólo se puede eliminar un enlace con un directorio si el directorio al cual se apunta está vacío.

Sistema de Archivos

Un aspecto fundamental de diseño en sistemas distribuidos es si todas las máquinas y procesos tendrán exactamente la misma visión de la jerarquía de los directorios.

En los sistemas que utilizan varios servidores de archivos mediante el montaje remoto generalmente los diversos clientes tienen una visión diferente del sistema de archivos, en este caso deberá existir un directorio raíz global al que todas las máquinas reconozcan .



Sistema de Archivos

Transparencia de los Nombres

La transparencia con respecto a la posición significa que el nombre de la ruta de acceso no sugiere la posición del archivo: se individualiza al servidor pero no se indica dónde está, por ello puede moverse dentro de la red sin necesidad de cambiar la ruta (Ej.: /servidor1/dir1/dir2/x).

Si se indica el servidor en las las rutas de acceso el sistema no puede desplazar el archivo a otro servidor en forma automática, porque cambiaría el nombre de la ruta de acceso. Un sistema donde los archivos se pueden desplazar sin que cambien sus nombres tiene independencia con respecto a la posición. El método usual para nombrar los archivos y directorios es el nombre de la máquina + la ruta de acceso.

Semántica de los Archivos Compartidos

Cuando se comparten archivos es necesario definir con precisión la semántica de la lectura y escritura.

En sistemas monoprocesador que permiten a los procesos compartir archivos, el problema que se puede presentar es por *retrasos en la red* por ej. Si un read ocurrido después de un write llega primero al servidor, obtendrá el valor previo al write.

Otro problema es el desempeño pobre de un sistema distribuido en donde todas las solicitudes de archivos deben pasar a un único servidor.

Una solución es permitir a los cliente mantener copias locales de los archivos de uso frecuente en sus cachés particulares.

Semántica de los Archivos Compartidos

Esto ocasiona el siguiente problema: un cliente modifica localmente un archivo en su caché, luego otro cliente lee el archivo del servidor, por lo tanto el segundo cliente obtendrá un archivo obsoleto. Una solución sería propagar inmediatamente todas las modificaciones de los archivos en caché de regreso al despachador, esto resulta ineficiente.

Otra solución es relajar la semántica de los archivos compartidos: los cambios a un archivo abierto solo pueden ser vistos en un principio por el proceso (o tal máquina) que modifico el archivo. Los cambios serán visibles a los demás procesos (o máquinas) sólo cuando se cierre el archivo y sea actualizado en el servidor.

Esta regla se conoce como la ***semántica de sesión***.

Modelos de acceso

Dado la complejidad del acceso a los archivos a través del S.D., surgen dos modelos de acceso (interfaz del servicio de archivos): el carga/descarga, y el de acceso remoto

El primero simplifica el acceso permitiendo únicamente las operaciones de cargar y descargar un archivo. El acceso a cualquier parte del archivo implica solicitar y *guardar una copia local del archivo completo*, y sólo se puede escribir de forma remota el archivo completo. Este método sería ineficiente a la hora de realizar pequeñas modificaciones en archivos muy grandes, como podrían ser bases de datos.

El modelo de acceso remoto es mucho más complejo, y permite todas las operaciones típicas de un sistema de archivos local.

Modelos de acceso

Caché de archivos: el objetivo para la memoria cache es reducir el tráfico en la red, si los datos necesarios para satisfacer la solicitud de acceso no se encuentra en la memoria caché, se trae una copia del servidor al usuario y los accesos se llevan a cabo con la copia en memoria caché. La idea es conservar allí el bloque de discos de acceso más reciente, para así manejar localmente los accesos repetidos a la misma información y no aumentar el tráfico de la red. Cuando se modifica una de estas copias, los cambios se deben reflejar en el archivo maestro para preservar la consistencia pertinente. Al problema para mantener las copias actualizadas respectos al archivo maestro se denomina problema de consistencia de memoria caché.

Semántica de consistencia

Una máquina cliente se enfrenta al problema de decidir si una copia de datos en memoria caché local es consistente con la copia maestra.

Si una maquina cliente determina que sus datos en memoria caché están desfasados, ya no pueden servir para el acceso y hay que colocar en la memoria caché una copia actualizada.

Implementación de un S. Distribuido de Archivos

Incluirá los siguients aspectos:

Uso de los archivos

Estructura del sistema

Ocultamiento

Duplicación ó Replica

Control de Concurrencia

Implementación de un S. Distribuido de Archivos

Uso de Archivos: Es poco usual compartir archivos, los procesos promedio utilizan pocos archivos pero antes de implantar un sistema de archivos hay que analizar los patrones de uso y distintas propiedades de dichos archivos.

Estructura del Sistema: En ciertos sistemas no existe distinción entre un cliente y un servidor, ya que todas las máquinas ejecutan el mismo software básico, y si una máquina desea dar servicio de archivos lo debe hacer exportando los nombres de los directorios seleccionados, para otras máquinas los puedan acceder.

En otros sistemas el servidor de archivos y el de directorios son sólo programas del usuario, y se puede configurar un sistema para que ejecute o no el software de cliente o servidor en la misma máquina.

Implementación de un S. Distribuido de Archivos

Los clientes y servidores también podrían ser máquinas totalmente distintas en términos de hardware o de software.

Un aspecto estructural a considerar es si los servidores de archivos, directorios o de otro tipo deben contener la información de estado de los clientes.

Una posibilidad es que los servidores no deben contener los estados, deben ser sin estado: Cuando un cliente envía una solicitud a un servidor: el servidor la lleva a cabo, envía la respuesta y elimina de sus tablas internas toda la información relativa a esa solicitud, el servidor no guarda información relativa a los clientes entre las solicitudes.

Otra posibilidad es que los servidores conserven información de estado de los clientes entre las solicitudes.

Implementación de un S. Distribuido de Archivos

Ocultamiento

En un sistema cliente - servidor, cada uno con su memoria principal y un disco, existen cuatro lugares donde se pueden almacenar los archivos o partes de ellos:

El disco del servidor.
La memoria principal del servidor.
El disco del cliente.
La memoria principal del cliente.

Réplica

Frecuentemente los sistemas distribuidos de archivos proporcionan la réplica de archivos como un servicio.

Existen varias copias de algunos archivos y cada copia está en un servidor de archivos independiente.

Implementación de un S. Distribuido de Archivos

Las principales razones para la réplica son el aumentar la confiabilidad al disponer de respaldos independientes de cada archivo, permitir el acceso a archivos aún cuando falle un servidor de archivos, repartir la carga de trabajo entre varios servidores.

Un sistema es transparente con respecto a la réplica si la misma se administra sin intervención del usuario.

Tendencias en los Sistemas Distribuidos de Archivos

Es probable que los cambios en el hardware tengan un efecto muy importante en los futuros sistemas distribuidos de archivos. También es probable el impacto del cambio en las expectativas del usuario.

Consideraciones Respecto del Hardware: El abaratamiento de la memoria principal permitirá disponer de servidores con memorias cada vez mayores, se podría alojar directamente en memoria el sistema de archivos logrando mayor sencillez y desempeño.

La disponibilidad de redes de fibra óptica de alta velocidad permitiría esquemas tales como: un servidor de archivos en la memoria principal del servidor, la eliminación del disco del servidor y del caché del cliente, simplificando significativamente el software.

Tendencias en los Sistemas Distribuidos de Archivos

La construcción de interfaces de red especializadas podrán permitir resolver por hardware problemas difíciles de soportar por software, cada interfaz de red tendrá un mapa de bits con un bit por cada archivo en el caché, se podrían habilitar cerraduras por archivo y para modificar un archivo un procesador activaría el bit correspondiente en la interfaz.

Escalabilidad: Una tendencia es hacia los sistemas cada vez más grandes. Sin embargo aquellos que operan bien para cientos de máquinas podrían fallar al trabajar con miles o decenas de miles. Generalmente los algoritmos centralizados no se escalan bien, convirtiendo al servidor centralizado en el cuello de botella; por ello se podría separar el sistema en unidades más pequeñas relativamente independientes entre sí.

Tendencias en los Sistemas Distribuidos de Archivos

WAN: Generalmente los sistemas distribuidos se asocian con redes de área local (LAN), pero cada vez será mayor la necesidad de conectarlos entre sí cubriendo grandes áreas (nacionales, regionales, continentales, etc.).

Los sistemas de archivos deberán soportar estas necesidades teniendo presente la heterogeneidad de los equipos, códigos de representación (ASCII, EBCDIC, etc.), formatos, etc.

Deberá atenderse a los cambios de tendencia en los requerimientos de las aplicaciones.

Un problema adicional e inherente en los sistemas distribuidos masivos es el ancho de banda de la red, que puede resultar insuficiente para el desempeño esperado.

Tendencias en los Sistemas Distribuidos de Archivos

Usuarios Móviles: Los usuarios de equipos móviles (laptop, notebook, etc.) están gran parte del tiempo desconectados del sistema de archivos de su organización, necesitando como solución el ocultamiento. Cuando está conectado, el usuario carga al equipo móvil los archivos que cree necesitará después, los utiliza mientras está desconectado y al reconectarse, los archivos en el caché deben fusionarse con los existentes en el árbol de directorios, logrando la sincronización. La conexión para la sincronización puede ser problemática si se utiliza un enlace de ancho de banda reducido. Lo deseable sería un sistema distribuido totalmente transparente para su uso simultáneo por parte de millones de usuarios móviles que frecuentemente se desconecten.

Tendencias en los Sistemas Distribuidos de Archivos

Tolerancia de Fallos

La difusión de los sistemas distribuidos incrementa la demanda de sistemas que esencialmente nunca fallen.

Los sistemas tolerantes a fallos requerirán cada vez más una considerable *redundancia* en hardware, comunicaciones, software, datos, etc.

La *réplica de archivos* sería un requisito esencial.

También debería contemplarse la posibilidad de que los sistemas funcionen aún con la *carencia de parte de los datos*. Los tiempos de fallo aceptables por los usuarios serán cada vez menores.