Capitulo 1
Conceptos Generales de Sistemas Distribuidos

Contenido

SISTEMAS DISTRIBUIDOS

1. QUE ES UN SISTEMA DISTRIBUIDO(SD)?

Un sistema distribuido esta conformado por nodos los cuales tienen su propia memoria privada, sin memoria física compartida en el sistema. Cada nodo es una computadora completa, con un surtido completo de periféricos. Dichos nodos poseen interfaces de red que les permiten distribuirse por todo el mundo. Cada uno de los nodos puede ejecutar un sistema operativo diferente, cada uno con su propio sistema de archivos y bajo diferentes administraciones.
Agregando una capa de software encima de los sistemas operativos,, pese a que se posee entidades con hardware y sistemas operativos diferentes, se logra un cierto grado de uniformidad. Esta capa de software es conocida como middleware.

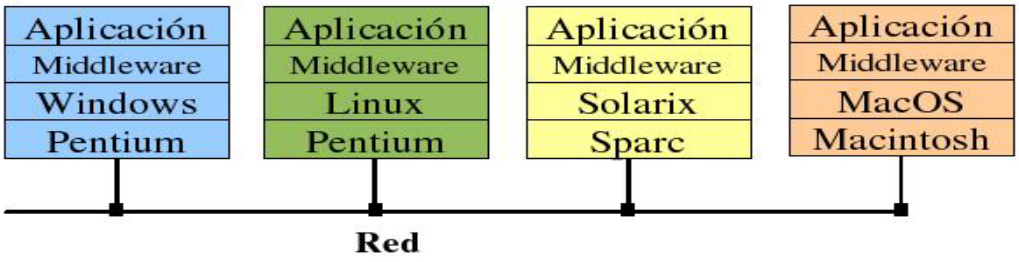

Figura 1. Sistemas hetereogéneos

Esta capa de software proporciona ciertas estructuras de datos y operaciones que permiten a procesos y usuarios de máquinas dispersas interoperar de forma consistente.

2. ELEMENTOS HARDWARE Y SOFTWARE QUE UTILIZAN LOS SDs.

2.1 Hardware de red.

Existen dos tipos principales de redes:
Redes de Área Local (LAN): Abarcan un edificio o un campus. Ejemplo: Ethernet, se describe en la norma IEEE 802.3.
Redes de Área Amplia (WAN): Abarca una ciudad, un país, etc. Internet. Esta red evolucionó partir de ARPANET. ARPANET fue una red experimental de conmutación de paquetes financiada por DARPA (Agencia de Proyectos de

Investigación Avanzada del Departamento de Defensa de US). Al evolucionar ARPANET se pudieron conectar muchos computadores y hasta otras redes LAN, conformando una federación de redes que hoy conocemos como Internet. En Internet podemos detectar dos tipos de nodos, los anfitriones (hosts) y los enrutadores (routers). Los anfitriones pueden ser los computadores personales, servidores, etc. Los enrutadores son dispositivos especializados en la función de conmutación de paquetes, ellos pueden aceptar paquetes por múltiples líneas de entrada y enviarlos por una de muchas líneas de salida. En Internet se tienen grandes redes que son operadas por la compañías de telefónicas y por los ISP (Proveedores de Servicios de Red).

2.2 Servicios de protocolos de red.

Todas las redes prestan ciertos servicios a sus usuarios (anfitriones y procesos) aplicando ciertas reglas que definen los intercambios de mensajes válidos.

Servicios de red.

Se tiene un *servicio orientado a conexión*, que sigue el modelo de los sistemas telefónicos . En el servicio orientado a conexión , el usuario de servicio primero establece una conexión, la usa y luego la libera. Ejemplo: Transferencia de archivos.

Se tiene un servicio sin conexión, sigue el modelo del sistema postal. Cada mensaje lleva la dirección de destino, y cada uno se enruta con independencia de los demás mensajes.

Protocolos de red.

Un protocolo se define como el conjunto de reglas que rigen la comunicación entre computadoras específicas. Las redes modernas usan una pila de protocolos, modelo de capas, en cada capa se resuelven diferentes asuntos. Ejemplos TCP, IP

2.3 Tipos de Middleware.

2.3.1 Middleware basados en documentos

Se implementa un inmenso grafo dirigido de documento, cuyos nodos pueden apuntar a otros documentos. Ejemplo la Web, cada página web contiene texto, imagenes, asi como hipervínculos (apuntadores) a otras páginas web. Cada página contiene una dirección única, usando notación URL(Uniform Resource Locator) protocolo://nombre-DNS/nombre-archivo

2.3.2 Middleware basado en el sistema de archivos.

El objetivo de este tipo de middleware es mostrar que un sistema distribuido

se asemeje a un sistema de archivos global. La comunicación se logra haciendo que un proceso escriba datos en un archivo y que los otros los lean.
Modelos de transferencia:
El modelo de subir/bajar.
El modelo de acceso remoto.

Jerarquía de directorios: Adoptar una ruta global de tal forma que todos los usuarios tengan la misma vista de la jerarquía de directorios.

Transparencia de nombres:
Se deben distinguir dos forma de transparencia:
Transparencia de ubicación: implica que el nombre no da idea de dónde está situado el archivo. el nombre /servidorx/dir1/dir2/archivox, nos indica que el archivo esta ubicado en el servidorx, pero no dice donde está el servidor.
Independencia de ubicación: Un sistema en el que los archivos pueden cambiarse de lugar sin que cambie su nombre se dice que tiene independencia de ubicación.

2.3.3 Middleware basado en objetos compartidos
En este tipo de middleware todo elemento del sistema es un objeto. Ejemplo, CORBA, es un sistema C/S donde los procesos cliente pueden invocar operaciones sobre objetos situados en máquinas servidoras.

2.3.4 Middleware basado en coordinación
Se basan en la comunicación y sincronización para lograr que procesos independientes interoperen. Ejemplo, el proyecto Linda. En Linda se crea un espacio global para todo el sistema, los procesos de cualquier máquina se comunican por medio mecanismos abstractos denominados tuplas. De esta manera las tuplas se asemejan a una gigantesca memoria compartida global.

3. CARACTERÍSTICAS CLAVES.

Existen 6 características primordiales que determinan la esencia de un sistema distribuido.

3.1 Recursos compartidos:
Caracteriza el rango de recursos que pueden ser útilmente compartidos en un SD. Ejemplo, recursos hardware(discos, impresoras), recursos software(archivos, bases de datos etc.).
- Esto reduce costos, favorece el trabajo en equipo.

- El compartir información se convierte en un requerimiento esencial en las aplicaciones computacionales:
  - Permite compartir resultados, herramientas, librerías, etc.
  - Permite implantar entornos computacionales que apoyan el trabajo en equipo.

En los SD pueden compartirse recursos hardware como discos e impresoras hasta elementos software como ficheros, ventanas, bases de datos y otros objetos de datos.
En los sistemas multiusuario clásicos los recursos compartidos se hacen de manera directa. Los usuarios de estaciones de trabajo monousuario o computadoras personales dentro de un SD obtienen los beneficios de los recursos compartidos de una manera indirecta. Los recursos en un sistema distribuido están físicamente encapsulados en una de las computadoras y sólo pueden ser accedidos por otras computadoras mediante las comunicaciones (la red). Para que la compartición de recursos sea efectiva, ésta debe ser manejada por un programa que ofrezca un interfaz de comunicación permitiendo que el recurso sea accedido, manipulado y actualizado de una manera fiable y consistente. Surge el término genérico de **gestor de recursos**.
Un gestor de recursos es un modulo software que maneja un conjunto de recursos de un tipo en particular. Cada tipo de recurso requiere algunas políticas y métodos específicos junto con requisitos comunes para todos ellos. Éstos incluyen la provisión de un esquema de nombres para cada clase de recurso, permitir que los recursos individuales sean accedidos desde cualquier localización.
Un sistema distribuido puede verse de manera abstracta como un conjunto de gestores de recursos y un conjunto de programas que usan los recursos. Los usuarios de los recursos se comunican con los gestores de los recursos para acceder a los recursos compartidos del sistema. Esta perspectiva nos lleva a dos modelos de sistemas distribuidos: el modelo **cliente-servidor** y el modelo **basado en objetos**.

Desde el punto de vista del RM se presentan dos modelos:

Modelo Cliente/Servidor
- Existen procesos servidores actuando c/u como un RM.
- Hay una colección de procesos cliente c/u realizando una tarea que requerirá alcanzar un recursos Hardware y Software.
- Todos los recursos software y hardware son mantenidos por un proceso servidor.

PROCESO ≡ Proceso en ejecución (posee un entorno de ejecución y al menos un hilo de control).
- El proceso cliente y servidor pueden estar en un mismo computador. Un servidor puede solicitar servicios a otro servidor.

- El esquema de nombrado varia dependiendo del RM que lo controla.

Modelo basado en objetos.
En este modelo toda entidad del sistema es visto como un objeto con una interfase de manejo de mensajes, la cual permite el alcance servicio ofrecidos por sus objetos.
- Lo objetos tienen un identificador único. Dicho objeto puede ser movido sin cambiar su identidad.
- Donde quiera que un proceso necesite un recurso debe enviar un mensaje de solicitud.
- Este mensaje es despachado a un objeto determinado, el cual realiza la operación solicitada y si es el caso envía la respuesta.
- Para referirse a un recurso se puede usar una manera uniforme (esquema de nombrado).
Existe un gestor de objetos, este gestor posee una colección de funciones y valores de datos que juntos caracterizan una clase.

3.2 Apertura:

Caracteriza si un sistema puede ser extendido o ampliado con respecto sus componentes hardware o software. Por ejemplo, la adición de nuevas características al sistema operativo, protocolos de comunicación y los servicios de recursos compartidos.
La apertura de los SD con respecto al software se determina por el grado de facilidad de adición de nuevos servicios y recursos, sin que esta adición modifique o duplique los recursos existentes en el sistema. Esta apertura es conseguida por especificación y documentación de las interfases. Estas interfases deben ser públicas.
Los sistemas distribuidos abiertos pueden construirse a partir de hardware y software heterogéneo(hardware y sw de propietarios diferentes).

3.3 Concurrencia.

Esta característica permite la ejecución de diferentes procesos en un computador.
- Cuando se tiene un solo procesador la concurrencia se consigue por interelevos de porciones de ejecución del procesador. Suponiendo que se tiene un sistema distribuido don existen M computadores, c/u con un procesador como mínimo habrían M procesos activos ejecutándose en línea.

En los SD la posibilidad de ejecución paralela ocurre por dos razones:
- Cuando muchos usuarios simultáneamente invocan comandos o interactuan con programas de aplicación. En este caso no se presenta conflicto pues cada aplicación se ejecuta de manera aislada en la misma

máquina.
- Cuando muchos procesos servidores ejecutándose concurrentemente, c/u responde a diferentes solicitudes de los procesos cliente. surge debido a la

existencia de uno o más procesos servidores para cada tipo de recurso. Estos procesos se ejecutan en distintas maquinas, de manera que se están ejecutando en paralelo diversos servidores, junto con diversos programas de aplicación. Las peticiones para acceder a los recursos de un servidor dado pueden ser encoladas en el servidor y ser procesadas secuencialmente o bien pueden ser procesadas varias concurrentemente por múltiples instancias del proceso gestor de recursos. Cuando los recursos compartidos son alcanzados por procesos que actúan concurrentemente es necesarios que el proceso servidor realice acciones de sincronización para evitar que ellos entren en conflicto. Por lo tanto las acciones de alcance de los recursos compartidos y su actualización, debe ser cuidadosamente planeada.

3.4 Escalabilidad

En el diseño de los sistemas se debe tener en cuenta el crecimiento o cambio del mismo. Al crecer un sistema distribuido pueden aparecer problemas tanto en el ámbito del hardware como en el del software. Por ejemplo, no se pueden añadir equipos por un mal dimensionamiento de las direcciones. A nivel de software, se pierden prestaciones, por ejemplo si se posee un sistema centralizado, entre mayor número de usuarios mayor cuello de botella en el servidor que posee el algoritmo de control central.

La escalabilidad, por lo tanto, se refiere a la capacidad del sistema para adaptarse a una demanda incrementada de servicio. Lo más común es que los ambientes distribuidos evolucionen y crezcan tanto en usuarios como en servicios, por lo tanto es deseable que pueda adaptarse a estas nuevas especificaciones sin causar interrupción de los servicios o pérdidas de desempeño significativas. Algunas reglas de diseño para la escalabilidad las mencionaremos enseguida.
Evitar entidades centralizadas.- El uso de una entidad centralizada como una base de datos o un servidor de archivos en especial vuelve al sistema no escalable debido a las siguientes razones:

La falla en el componente central causa la caída total del servicio en el sistema.

El desempeño de un componente centralizado se convierte en un cuello de botella cuando se tienen mas usuarios compitiendo.
Aunque el componente centralizado tenga la suficiente capacidad de desempeño, almacenamiento o velocidad, la red puede saturarse cuando el tráfico se incrementa por el acceso a dicho recurso.

En una red amplia, formada por varias redes interconectadas, es poco eficiente tener un tipo particular de peticiones atendido solo en un punto.
En general debe evitarse el uso de componentes centralizados, usando replicación de recursos y algoritmos de control para satisfacer este objetivo.

Evitar algoritmos centralizados.- Un algoritmo centralizado puede definirse como aquel que recolecta información de varios nodos, la procesa en uno solo y luego distribuye los resultados entre los demás. Su uso deberá ser restringido

3.5 Tolerancia a Fallos

Ante un fallo a nivel hardware y software, un sistema puede producir resultados incorrectos o pueden detenerse antes que ellos terminen la activad que estaban realizando. Para ello es necesario que el diseño del sistema distribuido posea un mecanismo tolerante a fallos. Estos mecanismo están basados en dos propuestas:

- Redundancia de hardware: En este caso se interconectan dos computadores para una sola aplicación, uno de ellos actúan como una máquina en stanby (host stanby) que entrara a actuar cuando la máquina activa falle.
- Recuperación de software: Ante un estado de fallo es necesario que un programa sea diseñado con el fin de preservar el estado permanente de la información.

Una de los requerimientos de los mecanismo tolerantes a fallos es el alto grado de disponibilidad que deben poseer. La disponibilidad de un sistema es una medida de la proporción de tiempo que esta disponible para su uso. Un fallo simple en una maquina multiusuario resulta en la no disponibilidad del sistema para todos los usuarios. Cuando uno de los componentes de un sistema distribuidos falla, solo se ve afectado el trabajo que estaba realizando el componente averiado.

3.6 Transparencia.

Esta caracerística esta definida como la acción de ocultar a los usuarios y al programador, la separación existente de los componentes en un SD, el objetivo es que el sistema sea visto como un todo en lugar de mostrar un conjunto de componente independientes. Gracias a esta característica los usuarios verán al sistema como un todo más que como una colección de componentes independientes.

Para explicar las 8 formas de transparencia identificadas en el manual de referencia RM-ODP [ISO 1996a], se introduce el término Objetos de Información (O.I). Los O.I, son entidades para las cuales la transparencia es

aplicada

Las formas de transparencia identificadas son:
- T. de Acceso: Permite que O.I sean alcanzado usando operaciones idénticas.

- T. Localización: Permite que los O.I sin conocer su ubicación en la red.

- T. Concurrencia: Permite a varios procesos operar concurrentemente usando O.I compartidos sin que exista interferencia entre dichos procesos.
- T. Replicación: Permite que instancias múltiples de O.I sean usados para incrementar la confiabilidad y el rendimiento sin conocimiento de la replicas por parte de los programas o procesos de usuarios.
- T. de Fallos: Permite el ocultamiento de fallas permitiendo a los procesos de los usuarios y a los programas completar sus tareas a pesar de las fallas de los componentes software y hardware.
- T. de Migración: Permite el movimiento de O.I dentro de un sistema sin afectar la operación de los procesos de usuario o programas.
- T. de Rendimiento: Permite al sistema ser configurado para mejorar el

rendimiento tal como varia la carga del sistema.
- T. de Escalabilidad: Permite al sistema y a las aplicaciones ampliar su escala sin cambiar la estructura o los algoritmos de la aplicación.

Las dos más importantes son las transparencias de acceso y de localización; su presencia o ausencia afecta fuertemente a la utilización de los recursos distribuidos. A menudo se las denomina a ambas transparencias de red. La transparencia de red provee un grado similar de anonimato en los recursos al que se encuentra en los sistemas centralizados.

3.7 Consistencia

Adicional a las 6 características descritas anteriormente, se incluye la característica de Consistencia. Esta característica no es una característica exclusiva de los SD, pero la Consistencia es un necesidad imperativa, pues sin ella el SD no funcionaría correctamente (inconsistencias). La consistencia determina que la información contenida en todo el SD debe ser coherente entre las distintas partes donde se encuentra y distintos estados por los que pasa. Esta característica no es exclusiva de los SD, pues también su cumplimiento es de vital importancia en un sistema centralizado, pero en los SD cobran una mayor importancia, pues en los SD se presentan una serie de situaciones donde se pueden producir inconsistencias. Entre ellas se tienen:
Consistencias de actualización: Cuando varios procesos acceden concurrentemente a un dato para actualizarlo se pueden producir inconsistencias por que la actualización del dato no se realiza como una

única operación atómica en exclusión mutua. Este tipo de inconsistencia se evita utilizando transacciones. Las transacciones son las equivalentes a las entradas y salidas de una región crítica que se utiliza para proteger la consistencia de un dato compartido. Además se asegura que las operaciones incluidas en la transacción o se realizan todas o no se ejecuta ninguna.

Consistencias de replicación: Cuando un conjunto de datos debe mantenerse replicado en diversos ordenadores de la red, con una alta probabilidad de modificación por parte de cualquier usuario, claramente se pueden producir situaciones en las que tales datos no son iguales en todos los ordenadores al mismo tiempo. Ejemplo: Juegos multiusuario en red.
La replicación de datos en los SD puede ser utilizada para mejorar la velocidad general del sistema o para soportar tolerancia a fallos.

Consistencia de caché: Cuando se realiza una acceso a un recurso (por ejemplo, un archivo), se puede guardar copia de los datos en una memoria local del cliente (memoria caché) para facilitar su acceso en futuras referencias de dicho recurso, evitando tener que transferir de nuevo los datos por al red. El problema surge cuando un cliente actualiza datos que también residen en las memorias caché de otros clientes. Hay distintas técnicas para

asegurar la consistencia de las cachés y se suelen tratar en la gestión de memoria de los sistemas operativos distribuidos y en la arquitecturas de microprocesadores.

Consistencias de reloj: Muchos algoritmos utilizados en aplicaciones y programación dependen de unas marcas de tiempo(timestamps), que indican el momento en el que ha sucedido un evento. En los SD una marca de tiempo generada en un ordenador se puede pasar a cualquier otro ordenador donde se podrá comparar con otras marcas generadas localmente. El problema estriba en lo difícil que resulta el mantener la misma hora física en todos los ordenadores simultáneamente.

Consistencia de Interfaz de Usuario:
Cuando un usuario está trabajando con un aplicación interactiva distribuida, hay veces, por ejemplo pulsa el botón del ratón para realizar cierta acción. Esto generara la ejecución de operaciones tales como envío, transmisión y recepción de mensajes. Desde el punto de vista del usuario se presenta un tiempo de espera, hasta que la respuesta llegue. Este tiempo de espera si es muy grande determina una inconsistencia de interfaz, pues no se corresponde la acción de solicitud con lo reflejado en pantalla. Los estudios de ergonomía indican que el retardo en mostrar los datos correspondientes en una pantalla no debe ser mayor de un décima de segundo, si se quiere

dar la impresión de estar trabajando en una máquina local dedicada.

4. JUSTIFICACIÓN: POR QUE DISTRIBUIR?

VENTAJAS Y DESVENTAJAS DE LOS SISTEMAS DISTRIBUIDOS

4.1 Ventajas de los SD frente a los centralizados.

- La principal ventaja de la descentralización es la economía. Es más económico comprar varias CPU baratas y ponerlas a trabajar juntas, de esta manera se obtiene una mejor relación calidad/precio.
- Otra ventaja es la velocidad, un sistema centralizado tiene limitaciones físicas en su potencia de cálculo, en un sistema distribuido se puede obtener una mayor velocidad. No obstante no hay que olvidarse de la distribución de un trabajo en tareas, se presentan los factores de consumo de tiempo y sincronización.
- Otra ventaja, se adapta mejor a las aplicaciones que son inherentemente distribuidas tales como los robots de una cadena de montaje de una fábrica o bases de datos de empresas con múltiples sucursales. Los sistemas distribuidos son más fiables, pues al estar distribuida la carga y las labores entre múltiples máquinas, hay una menor dependencia del fallo de cualquiera de ellas.
- El crecimiento de un sistema distribuido puede hacerse de una manera incremental. En sistema centralizado si se dispone de un mainframe, la única

solución de crecimiento es cambiarlo por un modelo mejor y por ende más caro. En un sistema distribuido se dan las condiciones para que el añadir más procesadores al sistema sea más fácil, aprovechando además los procesadores que se tienen.

- Mayor flexibilidad, la carga de trabajo se puede distribuir entre las máquinas disponibles en la forma más eficaz según el criterio adoptado (por ej. costos).

4.2 Desventajas de los SD

- El principal problema es el software, es el diseño, implantación y uso del software distribuido, pues presenta numerosos inconvenientes. Los *principales interrogantes* son los siguientes:

  - ¿Qué tipo de S. O., lenguaje de programación y aplicaciones son adecuados para estos sistemas?
  - ¿Cuánto deben saber los usuarios de la distribución?
  - ¿Qué tanto debe hacer el sistema y qué tanto deben hacer los

usuarios?

La *respuesta a estos interrogantes no es uniforme* entre los especialistas, pues existe una gran *diversidad de criterios* y de *interpretaciones* al respecto.

- Control de comunicación, la presencia de una red de comunicaciones puede generar en caso de fallo, la pérdida de mensajes y por consiguiente el mal funcionamiento del SD. La presencia de las redes en los SD genera nuevos problemas, tales como:
Pérdida de mensajes y saturación.
Latencia puede provocar que al recibir un dato ya esté obsoleto.
La red es un elemento crítico.

- La seguridad del sistema, el acceso ilegal a la red puede conseguir un acceso a la información que se maneja dentro del SD.

- Diagnostico a fallas más difícil de detectar.

5. APLICACIONES DE LOS SDs.

● Aplicaciones comerciales
Reservas de líneas aéreas
Aplicaciones bancarias

Cajeros y almacenes de cadenas de supermercados

● Aplicaciones para redes WAN
Correo electrónico
Servicios de noticias(news)
Servicio de transferencia de archivos (ftp)
Búsqueda de archivos
Servicios de consulta
World Wide Web(www)

● Aplicaciones Multimedia
Videoconferencia
Televigilancia
Juegos multiusuario
Enseñanza asistida por ordenador

Áreas de los SD
Comunicaciones
Seguridad Distribuida.

Bases de datos distribuidas.
Servidores Distribuidos de Archivos.
Lenguajes de programación distribuidas.
Sistemas Tolerantes a fallos.

6. CUESTIONES DE DISEÑO.

Se describen una serie de tópicos básicos de diseño que surgen específicamente de la naturaleza de los SD.

6.1 Nombrado: Los SD están basados en los recursos compartidos y en la transparencia de su distribución. Los nombres asignados a los recursos deben tener un significado global que es independiente de la localización de los objetos y ellos deben ser soportados por un sistema de interpretación de nombres que pueda traducir los nombres con el fin de permitir alcanzar dichos recursos. Durante el diseño se debe tener en cuenta el esquema de nombrado que permitirá, desde el punto de vista de recursos, escalar al sistema mediante una traducción eficiente de los nombres.

Un proceso que requiera alcanzar un recurso, que el no gestiona, debe poseer un nombre o un identificador para dicho recurso.
name: son cadenas de caracteres que son entendibles por el hombre.
identifier: son nombres que son interpretados solo por los programas.
resolved: Se entiende como la operación efectuada sobre un name o un identifier en la cual se asocia el name/identifier junto con otro atributo, por ejemplo identificador, dirección IP o un número de puerto.

La resolución de nombres puede involucrar varios pasos de traducción. En cada paso un identificador o nombre es mapeado a un identificador de más bajo nivel que puede ser usado para especificar un recurso cuando se comunica con algún componente software. En algún estado de la secuencia de comunicación, un identificador de comunicación además hará traducciones para producir direcciones de red e información de encaminamiento que son aceptables para las capas más bajas de red.

Durante el diseño del nombrado, se debe tener en cuenta:
- La selección de un apropiado espacio de nombres por cada tipo de recurso. Un espacio de nombres puede ser finito e infinito, estructurado o plano. Por ej, en los sistemas basados en objetos, los objetos son uniformente nombrados, ellos ocupan un solo espacio de nombres.
- Los nombres de recursos deben ser resueltos a identificadores de comunicación. En general esto es hecho manteniendo una copia del nombre y su traducción en un name service (N_S).

Para casos como archivos, los cuales son alcanzados frecuentemente, la resolución de nombres de ellos es mejor relegarla al gestor de recursos. Es posible hacer uso de técnicas de cacheo para preservar los nombres evitando así una comunicación innecesaria con el gestor de recursos o el servidor de nombres.

Donde esta el N_S?
El N_S nos proporciona la dirección del servidor solicitado pero como podemos obtener la ubicación del N_S?.
Para determina la ubicación del N_S se tienen las siguientes alternativas:
a- Ubicar la N_S en un computador con dirección bien_conocida, e incluirla al compilar todos los clientes y servidores.
b- Provisión dinámica de la dirección del N_S por parte de los sistemas operativos del cliente y servidor.
c- Uso de mensajes de búsqueda por difusión para localizar el N_S. Una vez el proceso N_S reciba este mensaje, contesta al emisor con su dirección.

Como proceder si se cambia la dirección del N_S?

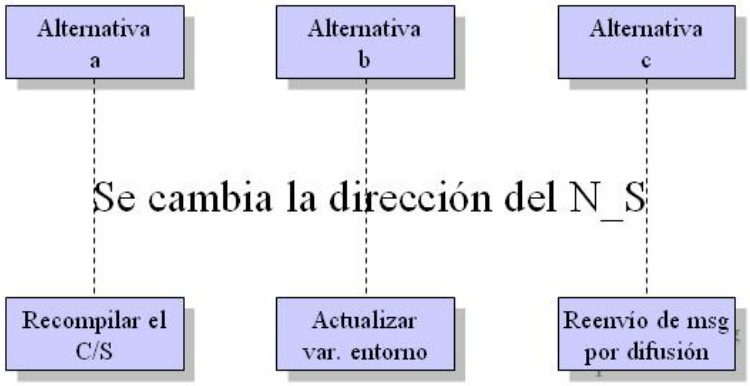

Figura 2. Alternativas de ubicación del N_S

6.2 Comunicación: En la comunicación las técnicas de rendimiento y confiabilidad se convierten en factores críticas para su eficiencia. Un asunto de diseño es optimizar la implementación de comunicación en el SD, mientras se retiene un modelo de programación de alto nivel para su uso.

Debido a que los componentes de un SD están lógica y físicamente separados, para llevar a cabo una tarea ellos necesitan interactuar para lo cual se requiere de mecanismo de comunicación.

La comunicación implica, operaciones por parte de interlocutores y sincronización.

La comunicación entre procesos involucra el envío y recepción de:
- Información de comunicación para la sincronización de procesos.
- Información desde el entorno de los procesos de envío al entorno de los procesos de recepción.

.
Las construcciones de programación usadas son primitivas de comunicación, tales como send() y receive(). Utilizando estas primitivas se realizan

Acciones de paso de mensajes entre procesos: Esta acción involucra un mecanismo de comunicación (un canal o puerto) y la aceptación por parte del proceso de recepción del mensaje.

Existen dos tipos de mecanismos de comunicación:
- Síncrona: Cada operación finaliza cuando se completa la dupla send() - receive().
- Asíncrona: Las operaciones pueden completarse por separado.

Dependiendo del comportamiento de las primitivas de comunicación con respecto a la transmisión de mensajes a continuación se muestran los diferentes casos presentados (ver tabla 1)

Tipo	***send()***	***receive()***
síncrona	bloqueante	bloqueante
asíncrona	no bloqueante	bloqueante
asíncrona	no bloqueante	no bloqueante

Tabla 1.

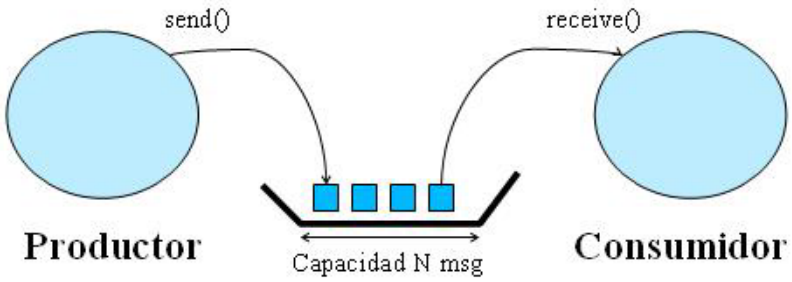

Figura 3. Comunicación de procesos (send-receive)

receive() no bloqueante requiere notificación por interrupción, evento o

encuesta.
receive() bloqueante es fácil de programar en entornos con varios hilos.

En la comunicación entre procesos se presentan dos modelos
- Comunicación Cliente/Servidor
- Comunicación por multidifusión

Comunicación Cliente/Servidor:
Este modelo esta orientado hacia el suministro de servicios. Durante el intercambio de información se realizan las siguientes acciones:

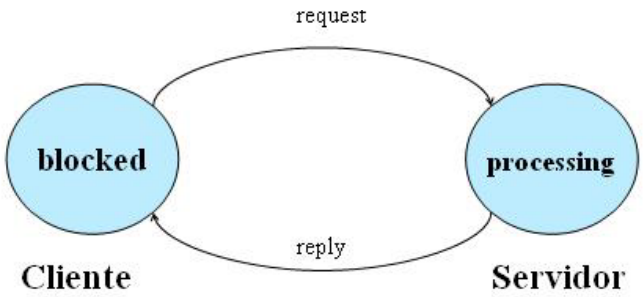

Figura 4. Comunicación Cliente/Servidor

- Transmisión de una solicitud (request) desde un proceso cliente a un servidor.
- Procesamiento de la solicitud por parte del proceso Servidor.
- Transmisión de la respuesta al proceso Cliente.

Cada solicitud posee un identificador que es usado para transmitir la respuesta al cliente. Típicamente cuando un servidor inicia su actividad este se registra con un servicio de nombres, indicando su dirección de red y el nombre del servicio que este proveerá. Los clientes obtienen la dirección de red del servidor consultando por un servicio al servicio de nombres, para así habilitar la comunicación con el proceso servidor.

Comunicación por multidifusión(multicast): Se usa el paso de mensaje, pero en este caso dicho mensaje no va dirigido a un solo proceso servidor sino a un grupo de procesos servidores.
El motivo de su uso se da en los siguientes casos:
- Ubicación de objetos: Por ejemplo el cliente envía un mensaje que contiene el nombre de un directorio solo aquel proceso que contiene el nombre del directorio responde a la solicitud.

- Tolerancia a fallos: Se envía una solicitud de un servicio a un grupo de servidores. Aquellos que ofrezcan dicho servicio responderán a la solicitud.

Si uno de los servidores falla existirá otro servidor que procese la solicitud.

- Múltiple actualización: Actualización de una marca de tiempo global al sistema.

6.3 Estructura del software Como se había mencionado la apertura del sistema se logra a través del diseño y construcción de componentes software con interfaces bien definidas. La abstracción de información es una importante técnica de diseño para SD. Los servicios pueden ser vistos como un gestor de objetos. La interface a un servicio puede ser vista como un conjunto de operaciones. Durante el diseño de un SD se debe estructurar un sistema donde nuevos servicios puedan ser introducidos y que inter-trabajen con los servicios existentes sin tener que duplicar elementos de servicios existentes.
La apertura en un SD significa que un SD puede ser configurado a la necesidad particular de una comunidad de usuarios o un conjunto de aplicaciones. Se dice que el SD es abierto cuando el hardware y software de diferentes compañías puedan trabajar juntos.
Desde el punto de vista de software, se dice que un servicio es abierto cuando pueden ser desarrollados e instalados fácilmente con la habilidad de adaptarse a las necesidades de diferentes tipos de aplicación. El SD debe proporcionar facilidades para que lenguajes convencionales trabajen con mecanismo RPC o comunicación multidifusión.

6.4 Localización de carga de trabajo: Cuando se presentan cambios en la carga de trabajo del sistema un asunto a considerar es optimizar su efecto en el sistema de tal manera que el rendimiento del sistema sea afectado en el menor grado posible.

Modificaciones en la estación servidora pueden optimizar el rendimiento del procesador y la capacidad de rendimiento del procesador y la capacidad de memoria, determinando la más grande tarea que pueda realizar entre usuarios.
Modificaciones al modelo anterior serian:
- Processor pool model: Incluir un procesador que dinámicamente localicen las tareas de los usuarios. La otra es una extensión del software que habilitan la localización de las tareas cuando estas están inactivas o subutilizadas.
- Multiprocesador con memoria compartida.

6.5 Mantenimiento de la consistencia: Este tópico de diseño en los SD distribuidos es un problema que conlleva altos costos, pues un mal manejo de este tópico impactará sobre el rendimiento del SD.