Sistemas Operativos 2

Unidad 6: Sistemas distribuidos

Arquitectura

René Ornelis
Primer semestre de 2025

Contenido

Índice de figuras

Arquitectura de los sistemas distribuidos

1 Transparencia

La transparencia es la característica principal que diferencia un sistema distribuido de un sistema de red. Se han identificado 8 formas de trasparencia:

- **Acceso:** permite acceder a archivos locales o remotos y a otros tipos de objetos usando las mismas instrucciones.
- **Ubicación:** permite acceder a un objeto determinado sin necesidad de conocer su localización.
- **Concurrencia:** permite a varios usuarios o programas de aplicaciones a operar concurrentemente o compartiendo datos sin una interfaz entre ellos.
- **Replicación:** permite múltiples instancias de archivos y otros datos usados e incrementa la veracidad y desempeño sin el conocimiento de las réplicas por parte de los usuarios o por los programas de aplicación. Es importante que los usuarios no accedan copias obsoletas
- **Fallos:** posibilita el ocultamiento de fallas, permitiendo a los usuarios y programas de aplicación completar sus tareas a pesar de fallas de componentes de hardware o de software. En el peor de los casos sólo se debe notar una degradación en el rendimiento.
- **Migración:** permite el movimiento de los objetos del sistema sin afectar la operación de los usuarios de los programas de aplicación.
- **Trasparencia en el Desempeño**: permite al sistema ser reconfigurado para mejorar el desempeño al variar la carga, por ejemplo.
- **Escalabilidad:** permite al sistema y a las aplicaciones expandirse en forma escalar sin cambiar la estructura del sistema o los algoritmos de aplicación.

De éstas las que se catalogan cómo las más importantes son la de acceso y ubicación y en conjunto forman la **transparencia de red.**

2 Consistencia

- **Actualización:** varios intentan actualizar unos distintos datos en distintas máquinas dentro de una transacción atómica (exclusión mútua distribuida)
- **Replicación:** si existen datos replicados en distintas máquinas, la actualización de una de las réplicas y el uso de un MULTICAST fallido
- **Caché:** el caché de una máquina puede quedar **anticuado** si otro proceso actualiza los datos correspondientes al caché.
- **Fallos:** son necesarios procedimientos de recuperación en cada máquina cuando falla otra máquina relacionada con un proceso conjunto.

- **Reloj:** existen algoritmos que se basan en **marcas de tiempos**, por ejemplo: compilación o reemplazo de páginas de memoria virtual, por lo que todos los componentes de la red deben tener la misma hora física
- **Interfaz de usuario:** En pantalla debe reflejarse lo que está pasando en sistema remoto.

3 Conceptos de hardware

Existen varios tipos de procesamiento que se pueden dar en un sistema distribuido dependiendo de la ubicación de los procesos concurrentes:

- **Multiprogramación:** los procesos comparten un mismo procesador
- **Ejecución paralela multiprocesador:** *n* procesos se ejecutan en *n* procesadores que comparten una memoria central común, es decir una máquina con *n* procesadores
- **Procesamiento distribuido (multicomputadora):** ejecución paralela de *n* procesos en *n* máquinas conectadas.

Entonces al ser requerido la ejecución de un proceso, el sistema debe decidir que tipo de procesamiento debe darle, según la **disponibilidad local** y **remota** de procesadores, lo cual no es tan simple como en un sistema centralizado. A esta tarea se le denomina **reparto de la carga**.

7 Diseño de sistemas operativos distribuidos

7.1 Servicios de comunicación entre procesos

7.1.1 Sockets cliente servidor

- BSD, POSIX y WINSOCKETS
- 90% de uso
- Esquema cliente servidor

7.1.2 Llamadas a procedimientos remotos

- Resguardos y suplentes
- Lenguaje de definición de interfaces
- Transparencia de parámetros
- Enlace dinámico
- Semántica de fallos

- no repetir
- al menos una vez
- a lo más una vez

- exactamente una vez

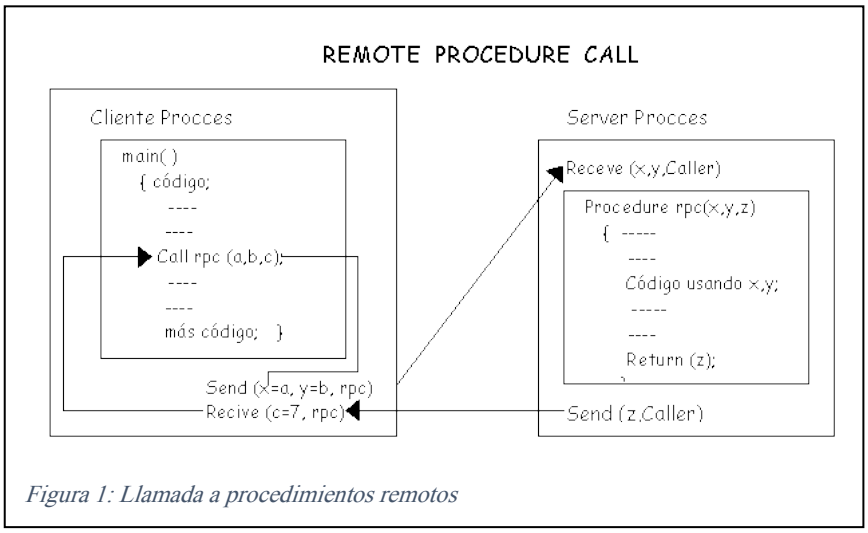

Figura 1: Llamada a procedimientos remotos

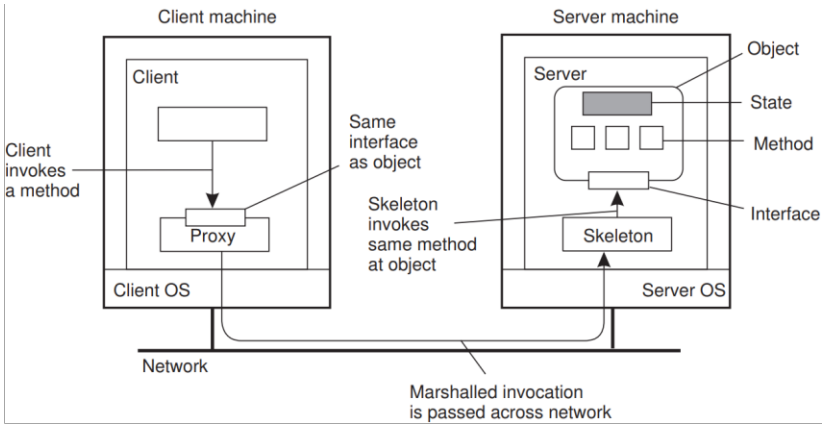

Figura 2: Estructura de la llamada a procedimiento remoto

7.1.3 Componentes distribuidas: CORBA y RMI

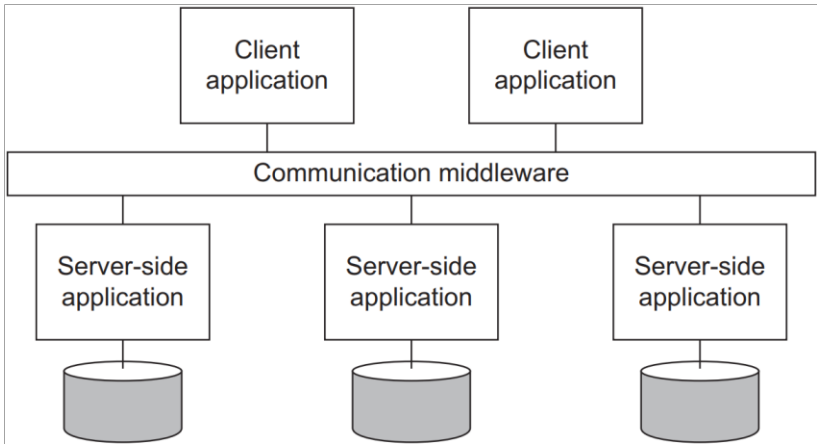

Figura 3: Arquitectura de middleware

7.2 Sincronización de procesos

7.2.1 Ordenamiento de eventos distribuidos

7.2.1.1 Relojes lógicos (Lamport)

- si *a* y *b* son dos eventos del mismo proceso y *a* ocurrió antes que *b* entonces *a* **precede a** *b*
- si *a* es el evento de envío de un mensaje de un proceso y *b* es el evento de recepción de ese mensaje en otro proceso, entonces *a* **precede a** *b*
- si *a* **precede a** *b* y *b* **precede a** *c* entonces *a* **precede a** *c*

Si dos eventos no están relacionados por la precedencia, entonces son concurrentes.

La implementación de estos relojes lógicos se puede realizar siguiendo las siguientes reglas

- Cada proceso tiene un reloj lógico RLi que se inicia en 0 y se incrementa cuando sucede un evento significativo
- Cuando un proceso *i* envía a otro un mensaje *m(t)* pone *t*=RLi
- Cuando un proceso *q* recibe un mensaje *m(t)* actualiza su reloj lógico así: RLq=max(RLq,t) + 1

En este esquema no existe un ordenamiento global, pero se puede implementar a través de añadir el número de proceso al reloj lógico. Así (Ta, Pi) es la marca de tiempo del evento *a* en el proceso *i*. Dadas dos marcas de tiempo (Ta, Pi) y (Tb, Pj) se dice que (Ta, Pi) < (Tb, Pj) si y sólo si

- Ta < Tb
- Ta = Tb y Pi < Pj

7.2.1.2 *Relojes vectoriales*

- Cada proceso tiene un reloj vectorial RVi(i_1, i_2,i_3,..,i_n) que se inicia en (0,0,0,...,0)
- En cada evento significativo del proceso i, se modifica RVi[i]=RVi[i]+1
- Cuando un proceso *i* envía a otro un mensaje *m(t)* pone *t*=RVi
- Cuando un proceso *q* recibe un mensaje *m(t)* actualiza su reloj vectorial así:
  - RVq[j]=max(RVq[j],t[j]) para todo j != q
  - RVq[q]=max(RVq[q],t[q]) +1

7.2.2 Exclusión mutua distribuida

- Algoritmo centralizado o de árbitro por recurso

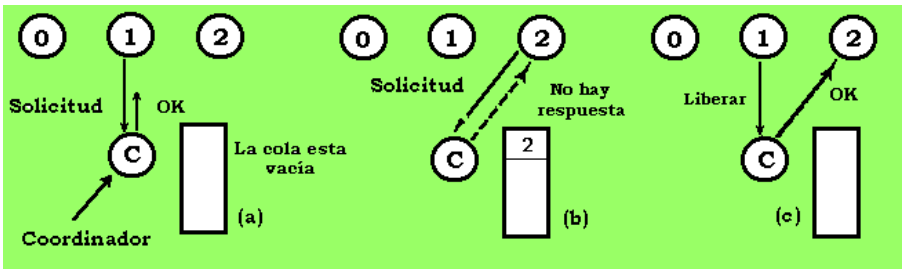

Figura 4: Exclusión mutua centralizada

- Anillos lógicos y paso de testigo

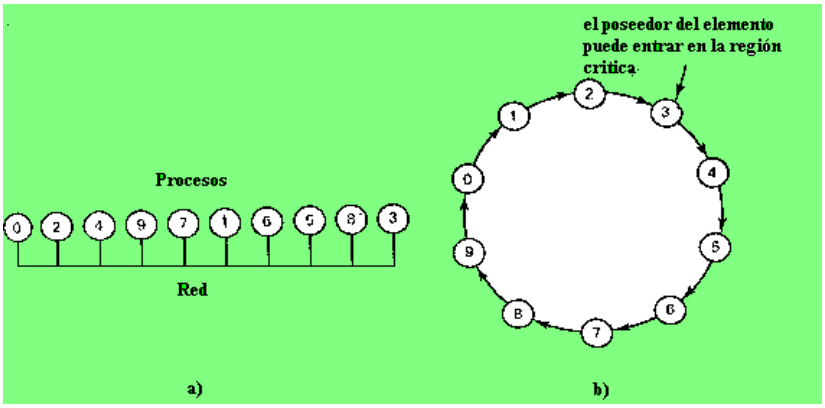

Figura 5: Exclusión mutua por anillo lógico

- Algoritmo distribuido

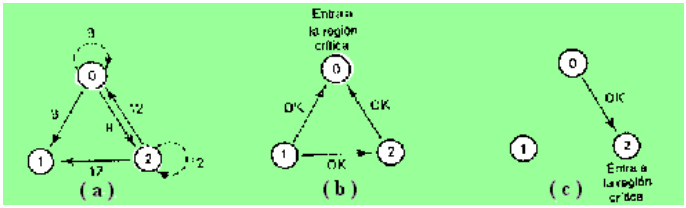

Figura 6: Exclusión mutua distribuida

7.3 Gestión de procesos

Definir en qué máquina se deben ejecutar los procesos.

- **Modelo de estación de trabajo**: Cada estación puede realizar cualquier trabajo
- **Pila de procesadores**: existen servidores/procesadores dedicados a realizar todo el proceso

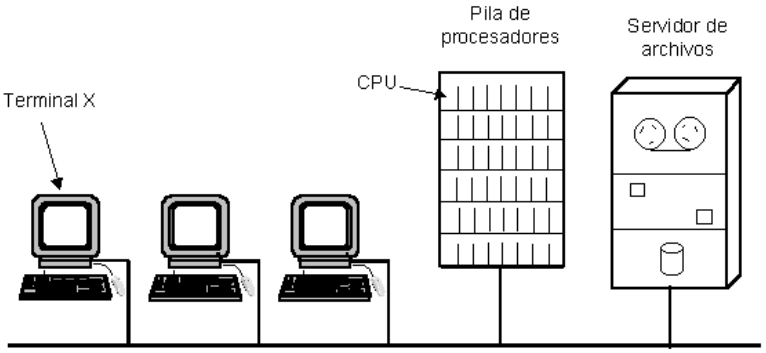

Figura 7: Pila de procesadores

- **Modelo híbrido**: procesos interactivos en la estación y procesos por lotes en la pila de procesadores

7.3.1 Asignación de procesos a procesadores en el modelo de estaciones de trabajo

- 80% - 100% de desperdicio de CPU
- el objetivo es decidir que procesador utilizar para un proceso equilibrando la carga del sistema y optimizando el rendimiento

- Localización de inactivas:

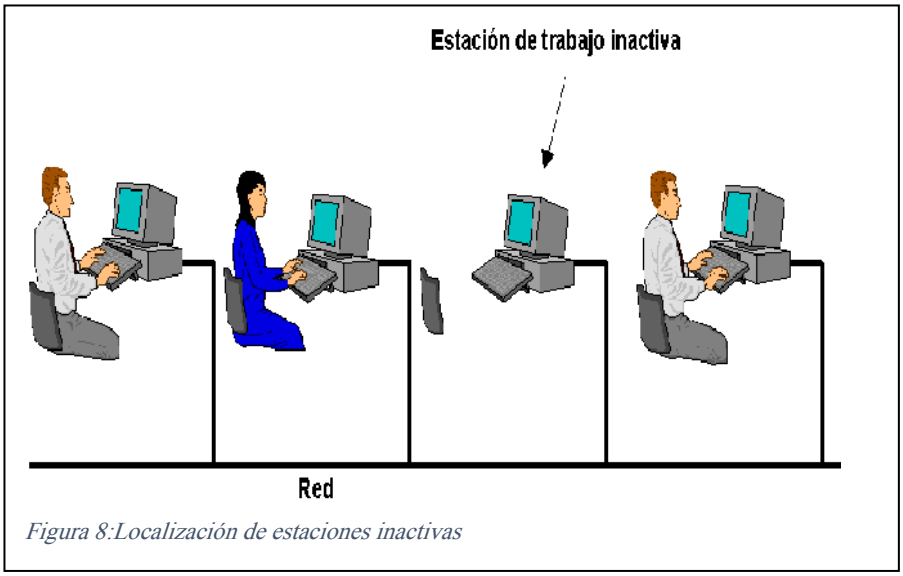

Figura 8:Localización de estaciones inactivas

- **Dirigidas por servidor**: se trata que cuando una estación necesite procesamiento, requerira a las demás estaciones capacidad de proceso. Es una petición por demanda
- **Dirigidas por el cliente**: una estación publica a todas las demás su disponibilidad de procesador. Es una oferta de procesador. Puede ser periódico o al alcanzar determinado umbral.

7.3.2 Algoritmos de distribución de la carga

Políticas por considerar:

- **Política de transferencia**: cuándo transferir
  - expulsivas: se puede migrar un proceso en cualquier momento. Trasladar la ejecución de una máquina a otra.
  - No expulsivas: Se asigna la ejecución, antes de iniciar el proceso. No hay traslado de la ejecución.
- **Política de selección:** qué proceso hay que transferir
  - sólo procesos en lotes
  - cualquier tipo de proceso:
    - basado en el uso del procesador: el que más consume
    - en el uso de recurso locales o remotos: puede ser conveniente trasladar un proceso a otra máquina que tenga los recursos que el proceso necesita.
- **Política de ubicación**: cuál será el nodo destinatario
  - mayor disponibilidad de procesador: uso intensivo de CPU
  - mayor disponibilidad de recurso: uso intensivo de recurso.
- **Política de información:** forma de obtener información de otros nodos

- **bajo demanda:** cuando se necesito proceso se requiere información, o cuando una estación llegue al estado "disponible" (% de disponibilidad)

- **periódicas:** cada cierto tiempo se envía información del estado de cada máquina a todas las máquinas

7.3.3 Migración de procesos

Consideraciones de migración de procesos

- tiempo de transferencia
- heterogeneidad de arquitecturas
- potencia de cada procesador
- tiempo restante de ejecución

7.4 Sistemas de archivos distribuidos

7.4.1 Nombrado

- espacio de nombre global: visión única del sistema de archivos

- Transparencia de la posición
- independencia de la posición

- resolución por servidor centralizado
- resolución por esquema distribuido: tabla de montaje replicada en cada nodo del
- sistema

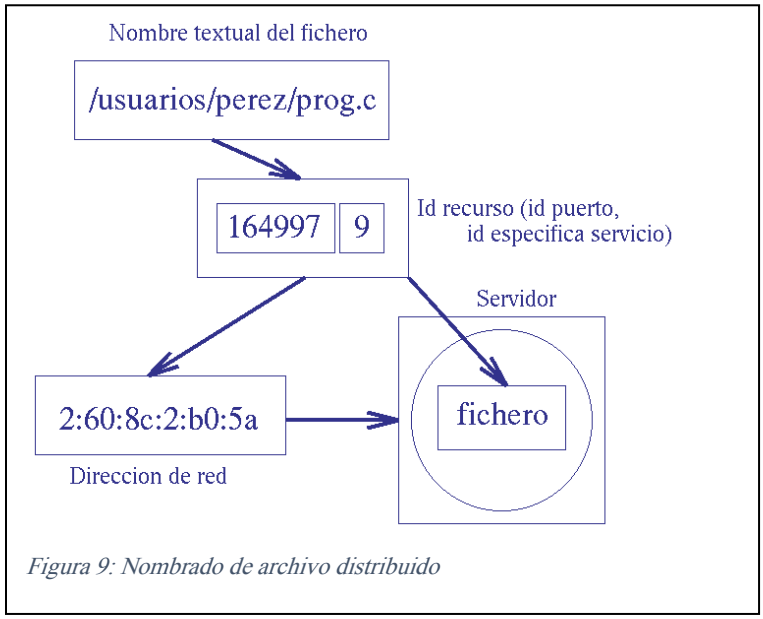

Figura 9: Nombrado de archivo distribuido