Sistemas Operativos Distribuidos

Prof. David A. Pérez A.

david.perez@ciens.ucv.ve

Facultad de Ciencias - UCV

Introducción

- Revolución en los sistemas de computo.
  - Mainframes.
    - 10 Millones de dólares → 1 instrucción por seg.
  - Personal Computer.
    - 1000 dólares → 10 millones de instrucciones por seg.
  - Ganancia precio/rendimiento.
    - 10^{11}
  - No todo evoluciona de esta manera.
    - Rolls Royce → 10 dólares → Miles de manuales.

Introducción

- Invención y desarrollo de las redes de datos.
  - WANs y LANs a altas velocidades.
  - Millones de bits/segundo.
- Sólo existe una mosca en la sopa.
  - Ideas.
  - El software requiere ser radicalmente distinto.
  - En particular el sistema operativo.

¿Qué es un sistema distribuido?

- “Conjunto de entes interconectados que trabajan armoniosamente prestándose servicios unos a otros”.
- “Colección de computadoras independientes que aparecen ante los usuarios del sistema como una única computadora”.

SD vs. Sistemas Centralizados

Item	Description
Economics	Microprocessors offer a better price/performance than mainframes
Speed	A distributed system may have more total computing power than a mainframe
Inherent distribution	Some applications involve spatially separated machines
Reliability	If one machine crashes, the system as a whole can still survive
Incremental growth	Computing power can be added in small increments

SD vs. PC

Item	Description
Data sharing	Allow many users access to a common data base
Device sharing	Allow many users to share expensive peripherals like color printers
Communication	Make human-to-human communication easier, for example, by electronic mail
Flexibility	Spread the workload over the available machines in the most cost effective way

Desventajas de los SD

Item	Description
Software	Little software exists at present for distributed systems
Networking	The network can saturate or cause other problems
Security.	Easy access also applies to secret data

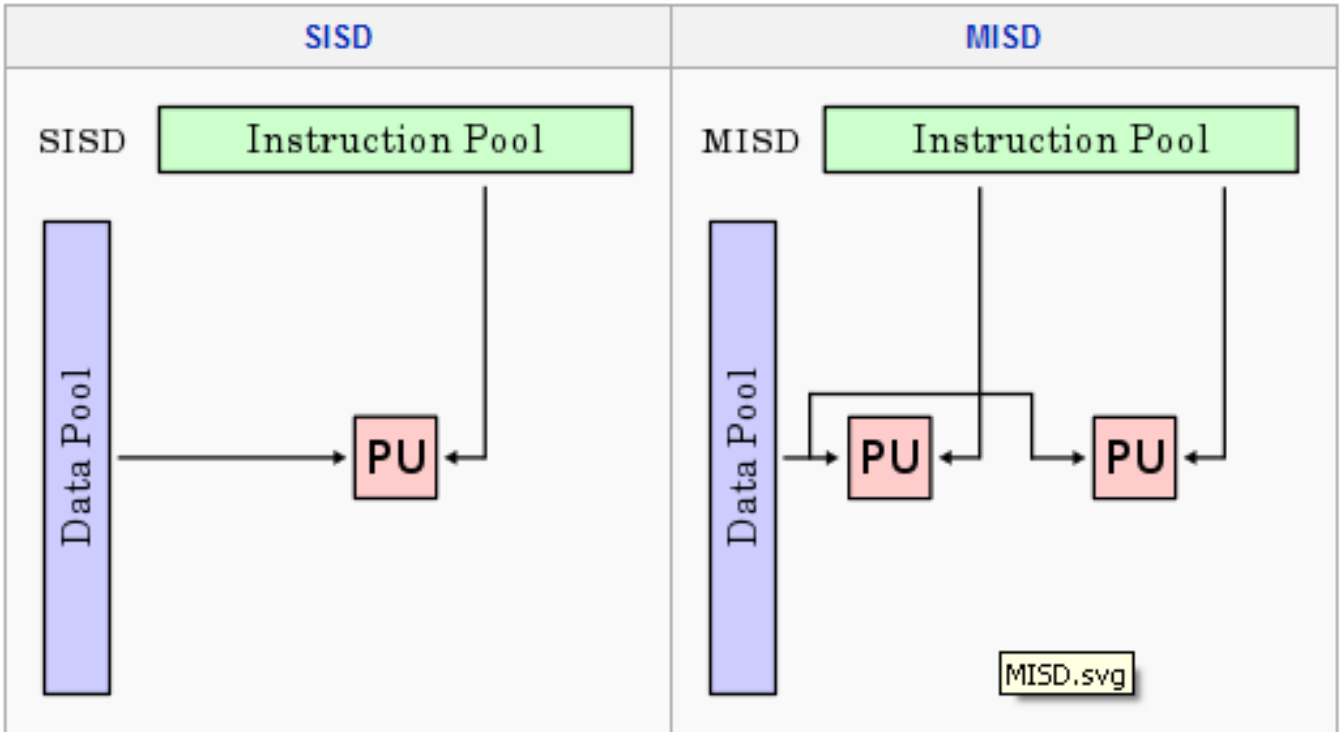
Hardware en SD

- Existen diversas maneras de organizar el hardware en un SD.
- Diversas clasificaciones.
  - Taxonomías de Flynn - 1972.
  - Rudimentaria.
  - Basada en dos características.
    - # de flujos de instrucciones.
    - # de flujos de datos.

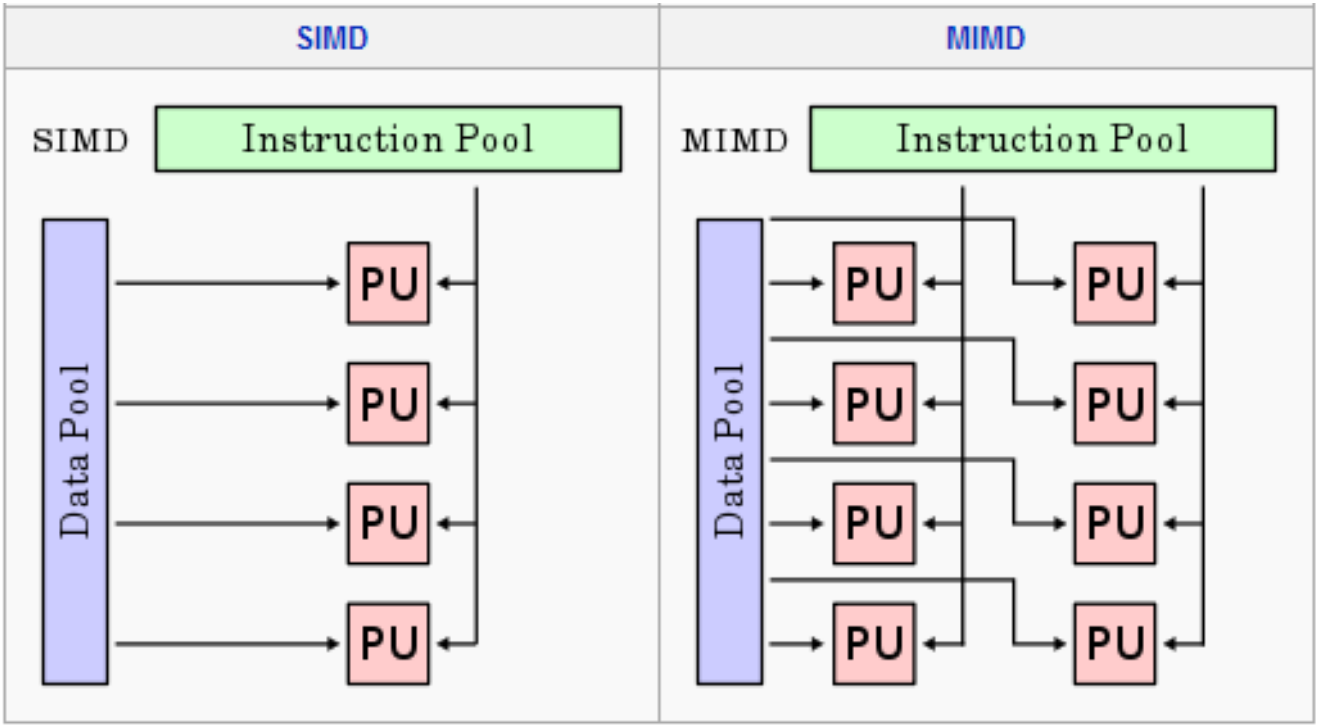
Taxonomías de Flynn

- SISD (Single Instruction, Single Data).
  - PCs a Mainframes.
- SIMD (Single Instruction, Multiple Data).
  - Suma de elementos de 100 vectores independientes.
- MISD (Multiple Instruction, Single Data).
  - Ningún computador se ajusta a este modelo.
- MIMD (Multiple Instruction, Multiple Data).
  - Datos independientes, PCs independientes.

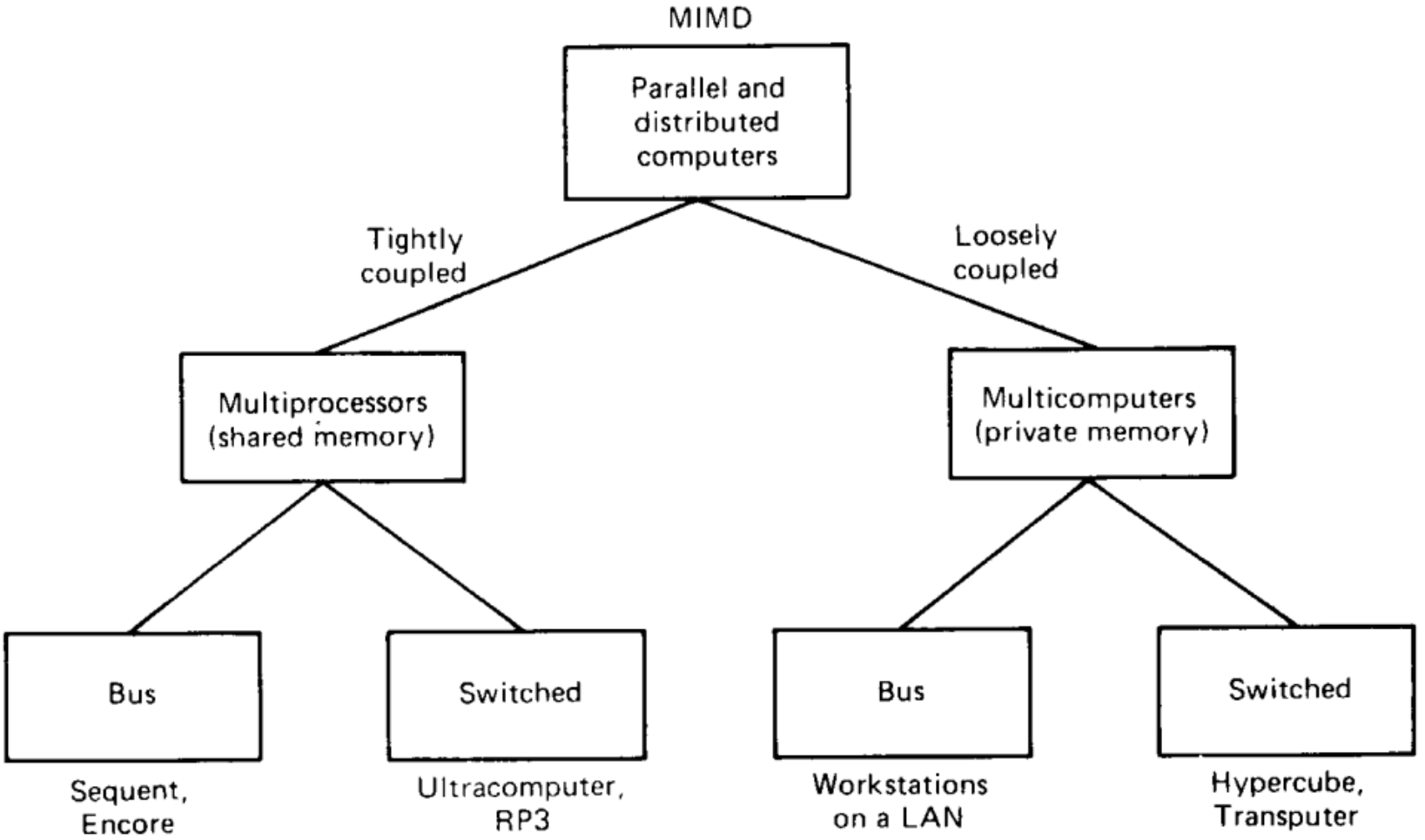
Taxonomías de Flynn



Taxonomías de Flynn



Una taxonomía diferente



Software en SD

- Aunque el hardware es importante, el software lo es más.
  - ¿Por qué?
- El software y en particular los sistemas operativos, no se pueden clasificar tan fácil como el hardware.

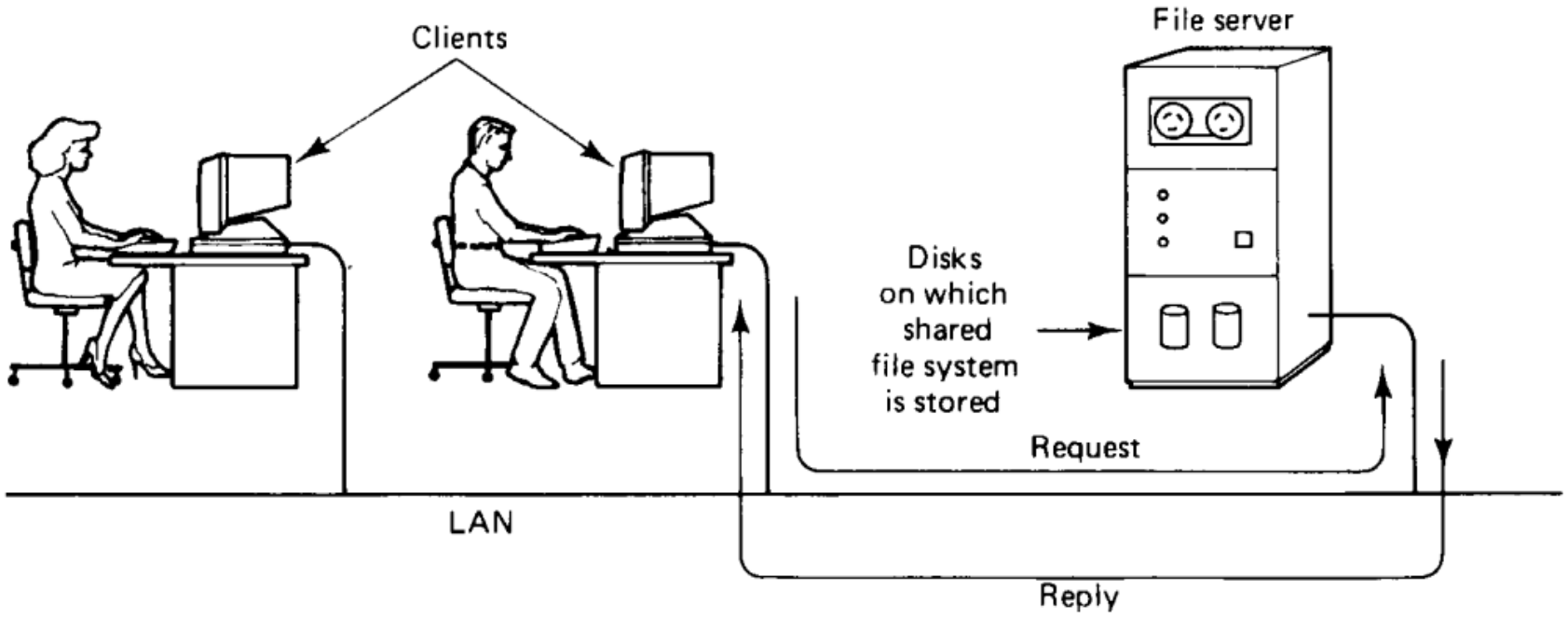
Software en SD

- Tratemos con esta separación:
  - Software débilmente acoplado.
    - Computadores independientes en red.
  - Software fuertemente acoplado.
    - Multiprocesador dedicado a la ejecución de un programa de ajedrez en paralelo.
      - Se asigna un tablero a cada CPU.

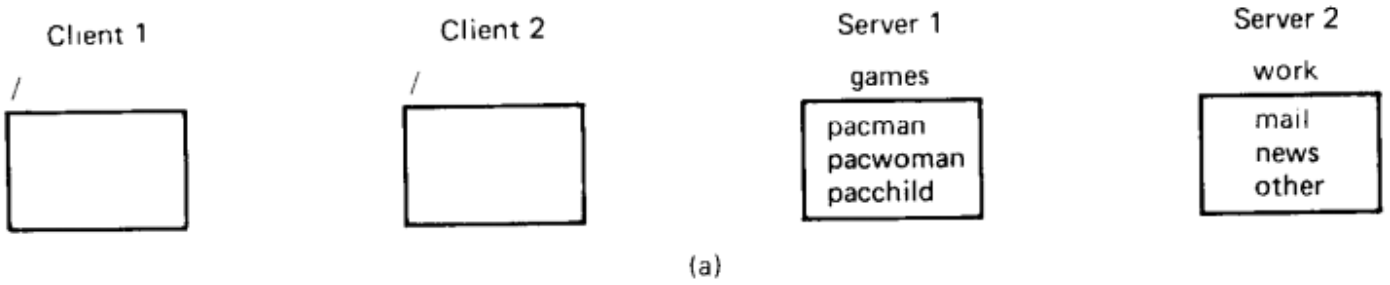
Sistema Operativo de Red

- Software débilmente acoplado en hardware débilmente acoplado.
- Ejemplos:
  - rlogin machine
  - rcp machine1:file1 machine2:file2
- Comunicación primitiva.

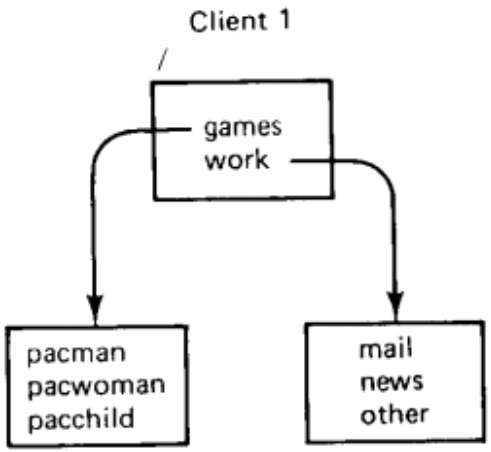
Sistema Operativo de Red



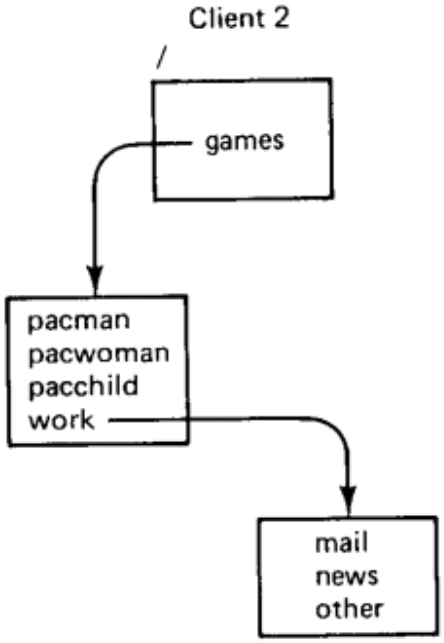
Sistema Operativo de Red



(a)



(b)



(c)

Sistema Operativo de Red

- ¿Qué es un Sistema Operativo de Red?
  - Ideas.
  - Máquina tiene un alto grado de autonomía.
  - Pocos requisitos a lo largo de todo el sistema.
- Reglas basadas en cliente-servidor.

Sistema Operativo Distribuido

- Software fuertemente acoplado en hardware débilmente acoplado.
- Ideas:
  - “imagen de un único sistema”
  - “uniprocesador virtual”

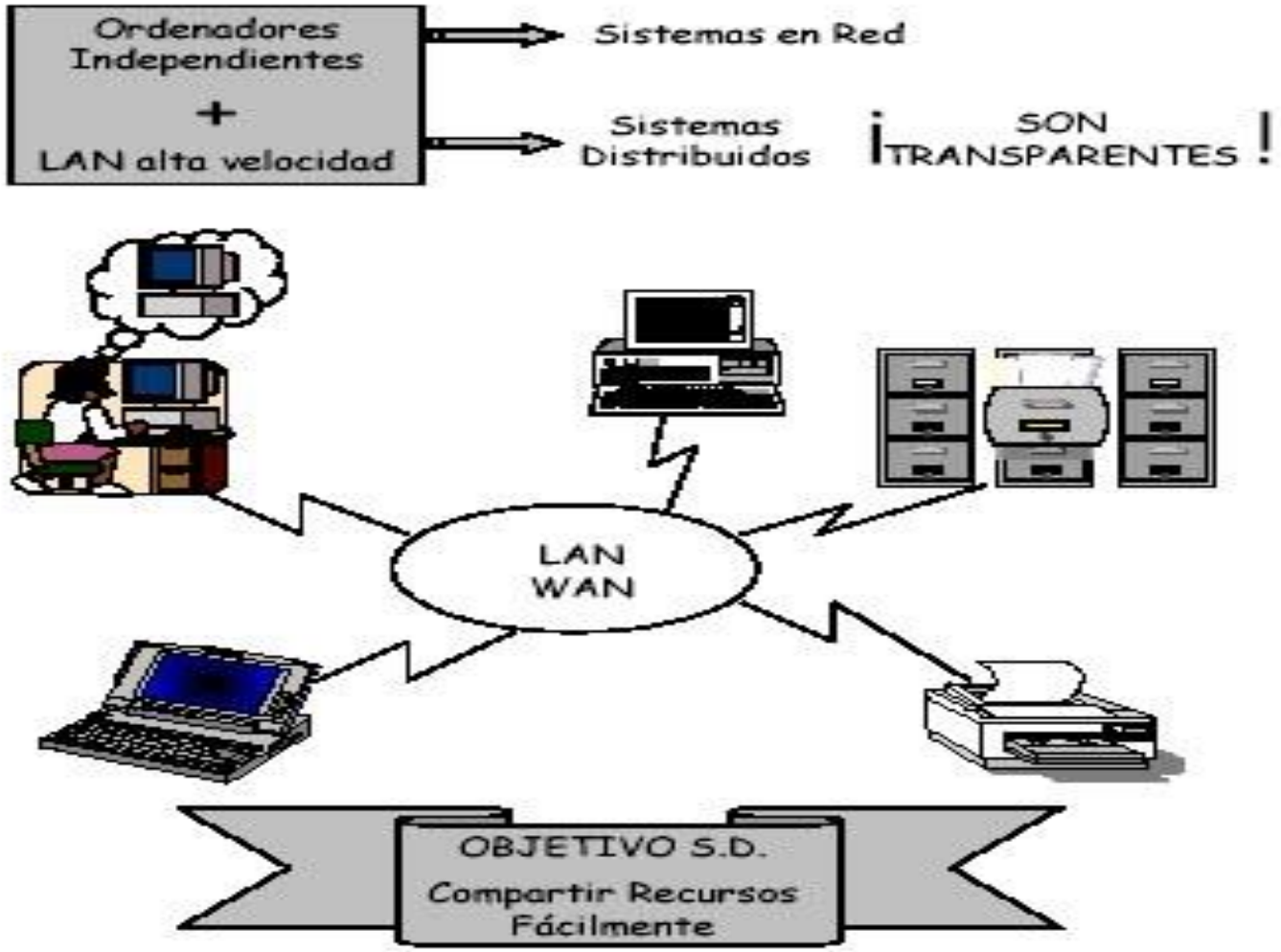
Sistema Operativo Distribuido

- Características:
  - Mecanismo de comunicación global entre procesos.
  - Esquema global de protección.
  - Administración global de procesos.
    - Cómo se crean, destruyen, inician y detienen.
  - Sistema de archivo con visión uniforme.

Sistema Operativo Distribuido

- Características:
  - Mismas interfaces.
    - Núcleos idénticos.
      - Coordinación de actividades globales.
  - ¿Cuál es el nivel de flexibilidad?
    - Visión en conjunto vs. Visión individual.

Sistema Operativo Distribuido



Aspectos de diseño

- Transparencia.
  - ¿Cómo engañan los diseñadores del sistema a todas las personas, de forma que la colección de máquinas es tan sólo un sistema de tiempo compartido?
  - Dos niveles:
    - Ocultar la distribución a los usuarios.
    - Hacer que el sistema sea transparente para los programas.

Aspectos de diseño

- Transparencia.
  - Es más difícil colocar una venda sobre los ojos del programador que los sobre los ojos del usuario final.
  - El concepto de transparencia puede aplicarse en varios aspectos de un sistema distribuido.

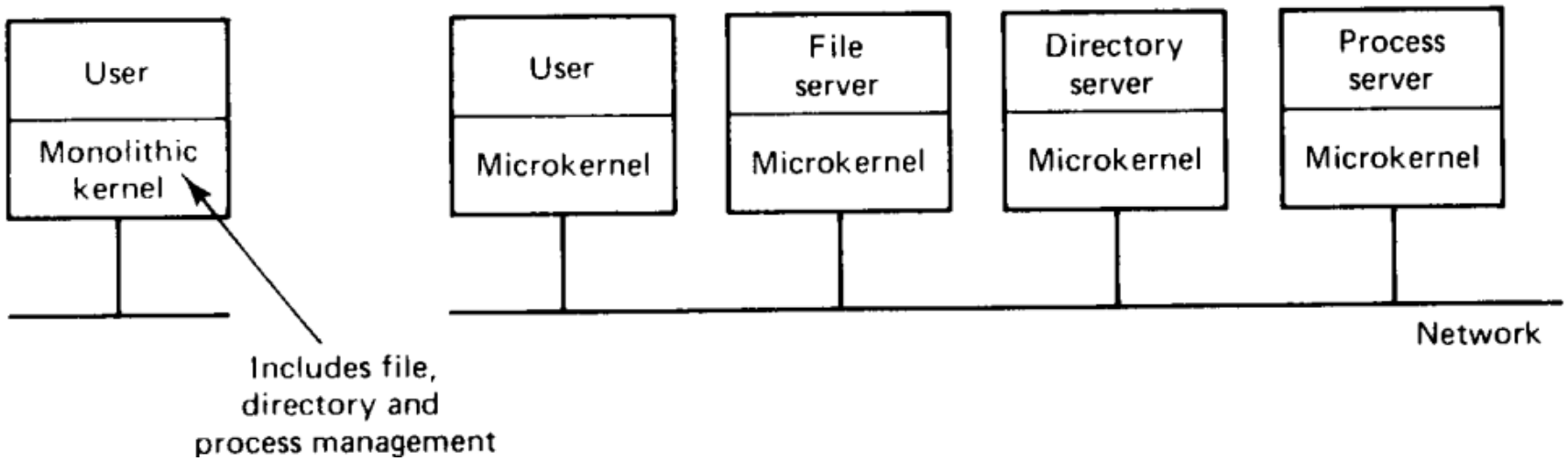
Transparencia

Kind	Meaning
Location transparency	The users cannot tell where resources are located
Migration transparency	Resources can move at will without changing their names
Replication transparency	The users cannot tell how many copies exist
Concurrency transparency	Multiple users can share resources automatically
Parallelism transparency	Activities can happen in parallel without users knowing

Aspectos de diseño

- Flexibilidad.
  - ¿Por qué es importante que un sistema operativo distribuido sea flexible?
    - Mantener abiertas opciones.
  - Existen dos escuelas de pensamiento en cuanto a la estructura de los sistemas distribuidos.
    - Ideas.

Flexibilidad



Aspectos de diseño

- Flexibilidad.
  - Ventajas y desventajas de cómo estructurar un SOD.
  - Comparación detallada (Douglis et al., 1991).
    - Sprite (Monolítico) vs. Amoeba (Microkernel).

Aspectos de diseño

- Confiabilidad.
  - Uno de los principales objetivos.
    - Mayor confiabilidad que los sistemas de un sólo procesador.
  - Similitud de un sistemas distribuido con un OR booleano.
  - Similitud de un sistemas distribuido con un AND booleano.

Aspectos de diseño

- Confiabilidad.
  - “un sistema distribuido es aquel del cual no puedo obtener un trabajo debido a que cierta máquina de la cual nunca he oído se ha descompuesto” (Leslie Lamport).
  - ¿Qué quiere decir esto?
    - Ironía.
  - Ideas para posibles soluciones.

Aspectos de diseño

- Desempeño.
  - Mejorar la velocidad de todo.
  - Métricas de desempeño.
    - Tiempo de respuesta.
    - # de trabajos por hora.
    - Uso del sistema.
  - ¿Qué papel juega la comunicación?
  - Prestar atención al tamaño del grano.

Aspectos de diseño

- Escalabilidad.
  - Ejemplo del sistema telefónico francés.
    - Minitel.

Concept	Example
Centralized components	A single mail server for all users
Centralized tables	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

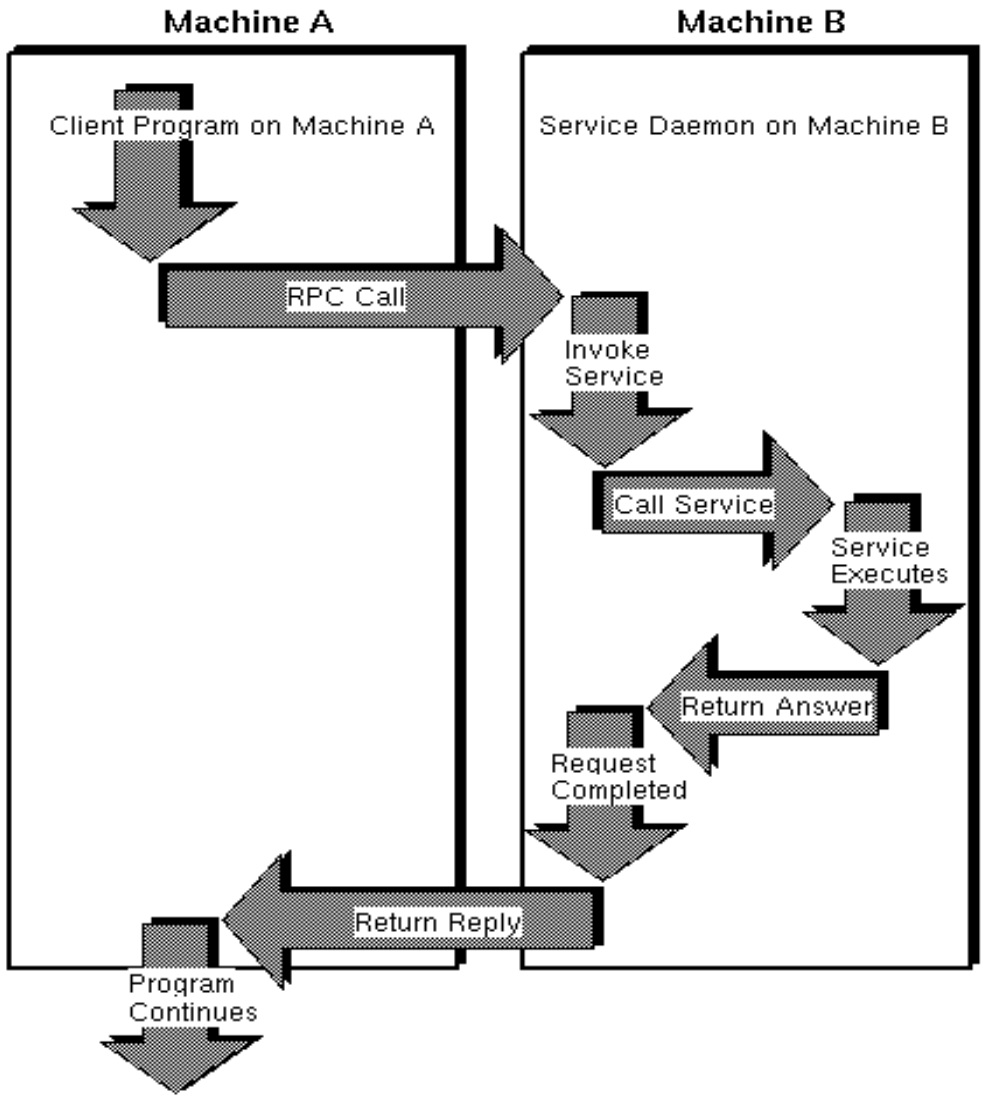
Aspectos de diseño

- Escalabilidad.
  - Características de los algoritmos descentralizados:
    - Ningún nodo conoce el estado completo del sistema.
    - Las decisiones se toman en base a información local.
    - La falla de un nodo no compromete el algoritmo.
    - No existe una hipótesis implícita de la existencia de un reloj global.

Remote Procedure Call

- Discutamos del paradigma cliente-servidor.
- Implicaciones de la E/S y el paso de mensajes.
- Birrell y Nelson (1984).
  - ¿Qué propusieron?

Remote Procedure Call



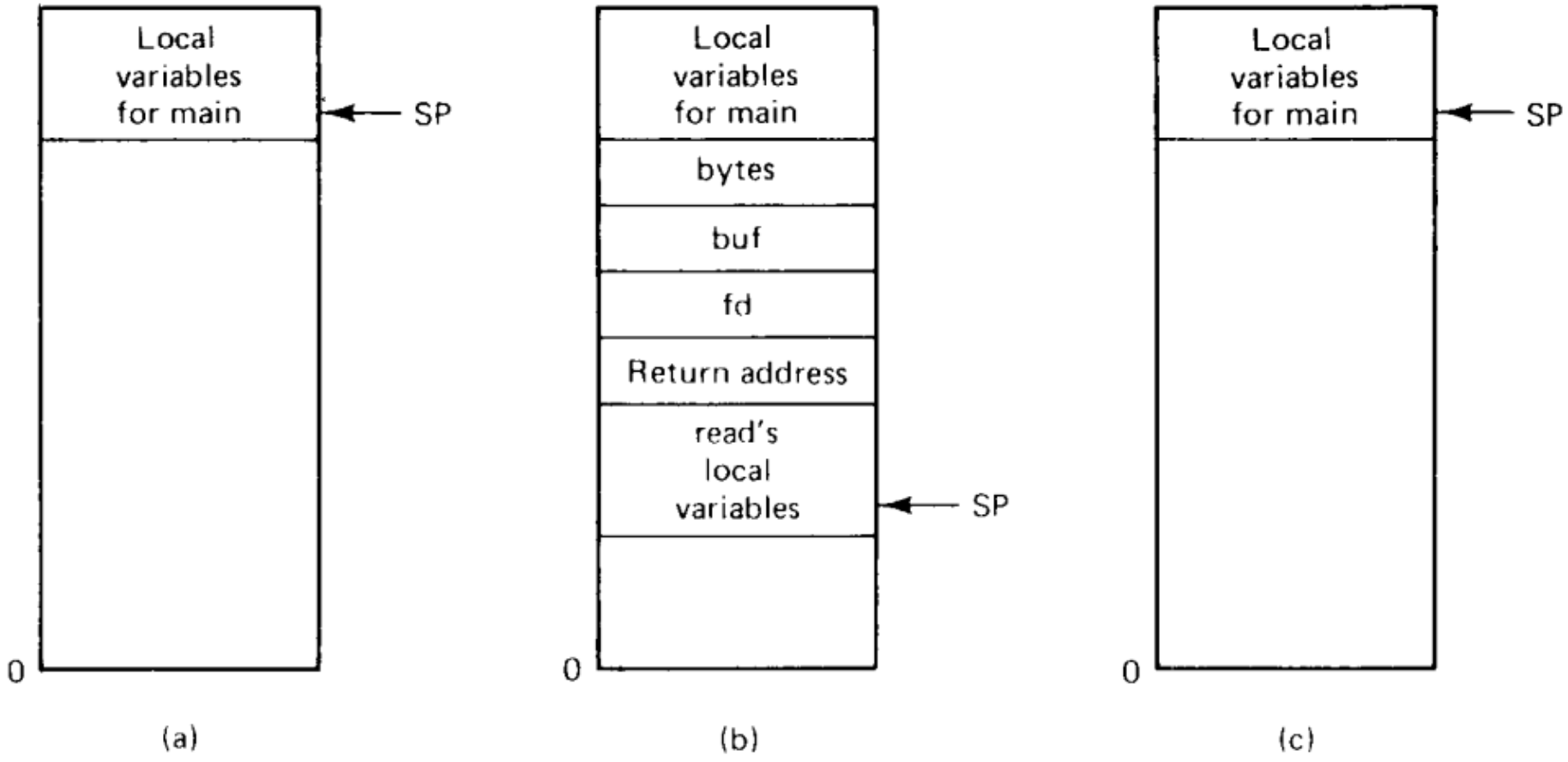
Remote Procedure Call

- La idea parece sencilla, pero existen algunas sutilezas.
  - Procedimientos en espacios de direcciones diferentes.
  - Necesidad de transferir parámetros y resultados.
  - ¿Qué ocurre en caso de fallas?

RPC – Operación básica

- Para entender el funcionamiento de RPC.
  - Es necesario entender una llamada convencional a procedimiento.
- count= read(fd, buf, nbytes);
- ¿Qué ocurre cuando se invoca a read?

RPC – Operación básica



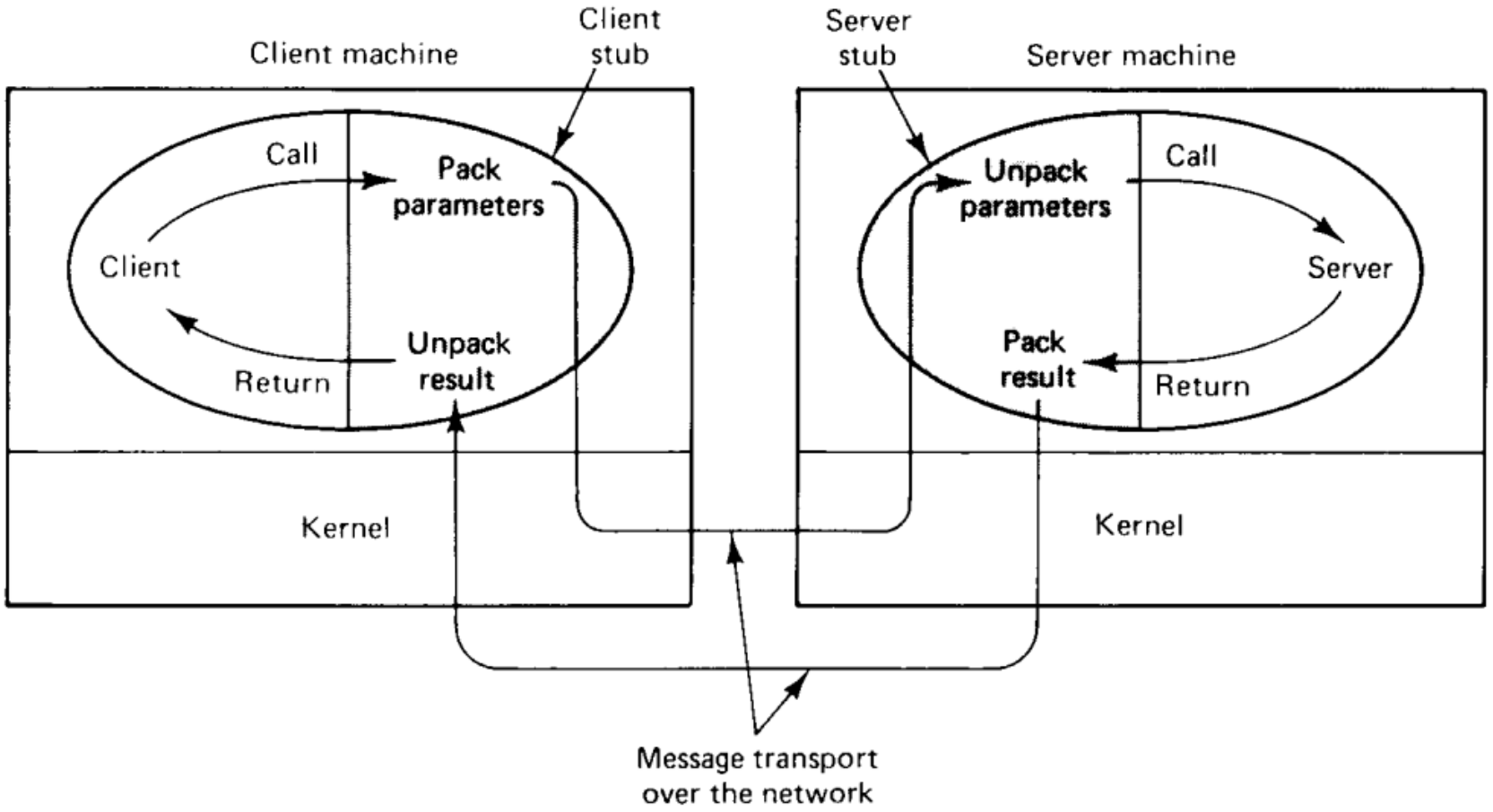
RPC – Operación básica

- Observaciones en cuanto al pase de parámetros.
  - Valor.
  - Referencia.
  - Copia/restauración.

RPC – Operación básica

- La idea detrás de RPC es que una llamada a procedimiento remoto se parezca lo más posible a una llamada local.
- ¿Cómo podemos lograr esto?
- Client Stub.
- Server Stub.

RPC – Operación básica



RPC – Operación básica

1. El cliente invoca al Client Stub.
2. Client Stub construye un mensaje y realiza una llamada al núcleo.
3. El núcleo envía el mensaje al núcleo remoto.
4. El núcleo remoto proporciona el mensaje al Server Stub.
5. El Server Stub desempaqueta los parámetros y llama al procedimiento.

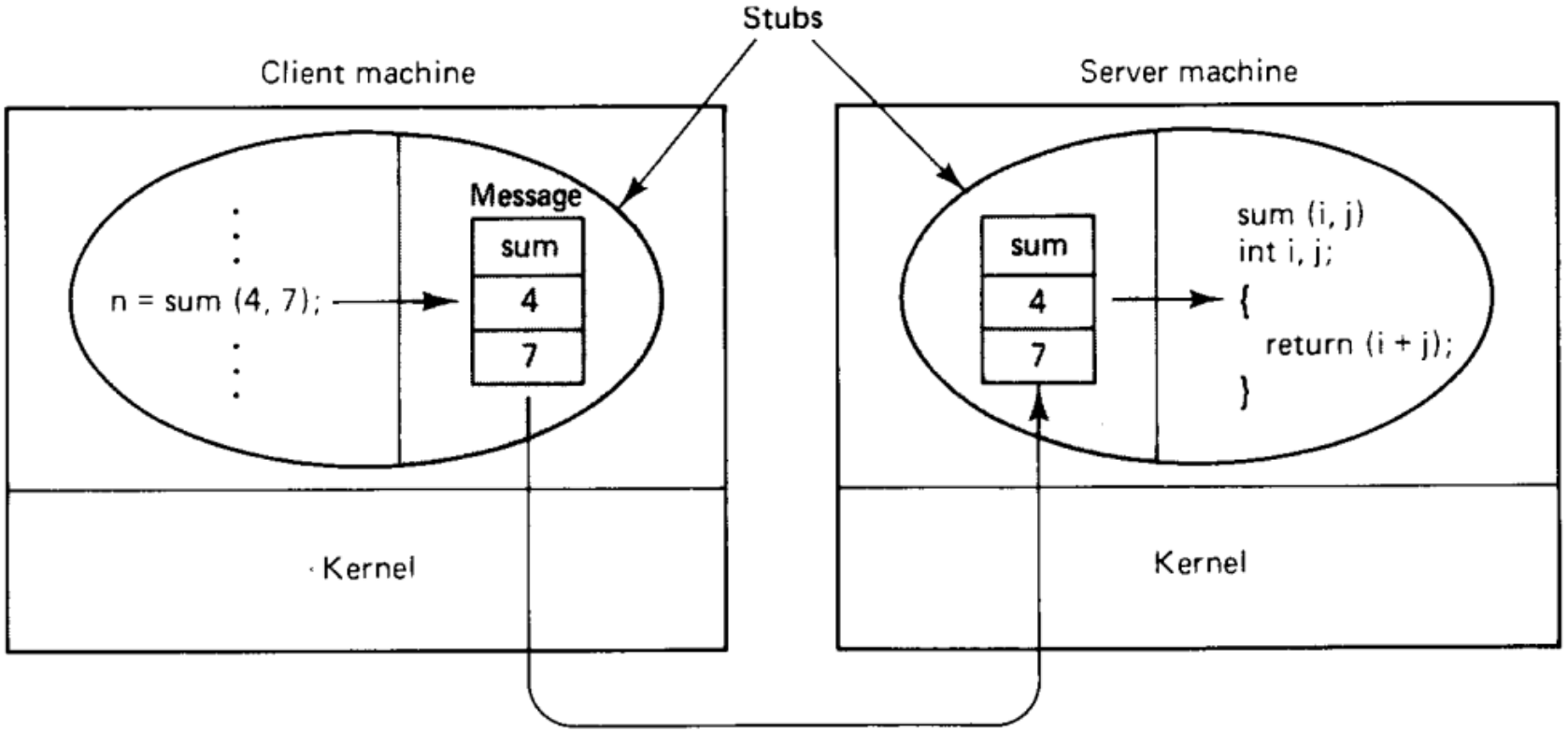
RPC – Operación básica

6. El procedimiento culmina y regresa el resultado al Server Stub.
7. El Server Stub empaqueta el resultado y hace una llamada al núcleo.
8. El núcleo remoto, envía el mensaje al núcleo cliente.
9. El núcleo del cliente da el mensaje al Client Stub.
10. El Client Stub desempaqueta el resultado.

RPC – Transferencia de parámetros

- ¿A qué nos referimos con esto?
- Ordenamiento de parámetros.
- Funcionamiento.

RPC – Transferencia de parámetros



RPC – Transferencia de parámetros

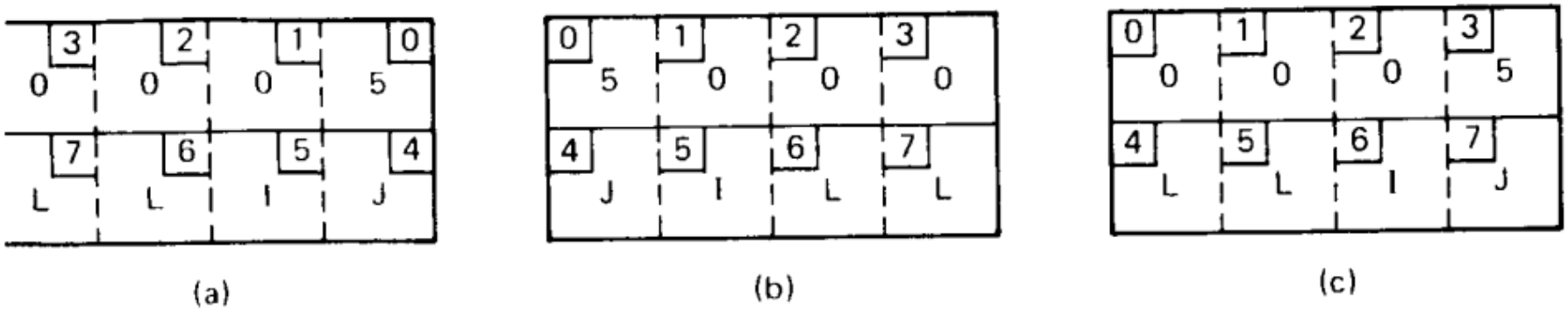


Fig. 2-20. (a) The original message on the 486. (b) The message after receipt on the SPARC. (c) The message after being inverted. The little numbers in boxes indicate the address of each byte.

RPC – Transferencia de parámetros

- Soluciones.
  - Ideas.
- Forma canónica.

RPC – Semántica en caso de falla

- Si el trasfondo de RPC es la transparencia.
  - ¿Qué hacer en caso de falla?
- Distinguiremos cinco clase de fallas.
  1. El cliente no puede localizar al servidor.
  2. Se pierde el mensaje de solicitud del cliente al servidor.
  3. Se pierde el mensaje de respuesta del servidor al cliente.
  4. El servidor falla antes de recibir una solicitud.
  5. El cliente fallas después de enviar una solicitud.

RPC – El cliente no puede localizar al servidor

- Razones.
  - El servidor esta inactivo.
  - El Server Stub es incompatible con el Client Stub.
- ¿Cómo solventar este inconveniente?
  - En C podríamos utilizar un valor especial, p.e: -1.
    - Implicaciones.
  - Solución alternativa.
    - Excepciones.

RPC – Pérdida de mensaje de solicitud

- Parece más fácil de tratar.
  - Ideas.
- Iniciar un temporizador en el núcleo al enviar la solicitud.
  - Funcionamiento.
  - ¿Qué pasa si se pierden todos los mensajes?
    - Regresamos al caso anterior.

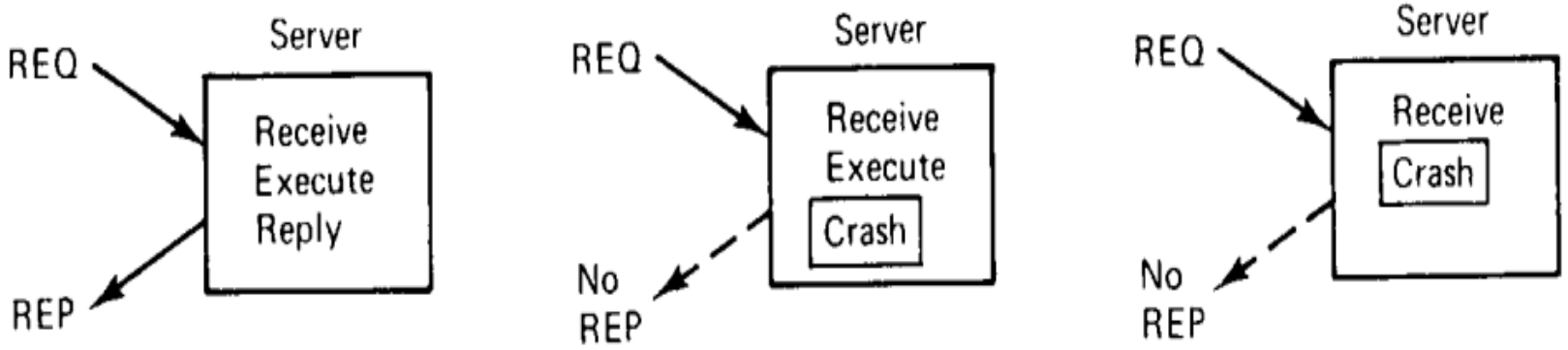
RPC – Pérdida del mensaje de respuesta

- Un poco más compleja de enfrentar.
  - Ideas.
- Solución basada en el caso anterior.
  - Funcionamiento.
  - Implicaciones.
    - Recuperar una porción de un archivo.
      - Idempotente.
    - Realizar una transferencia bancaria.

RPC – Fallas del servidor

- Posibles fallas del lado del servidor.
  - Ideas.
- ¿Dónde radica el problema?
  - El núcleo del cliente no puede diferenciar en que punto a ocurrido la falla.

RPC – Fallas del servidor



RPC – Fallas del servidor

- Soluciones.
  - Semántica de al menos una vez.
    - Mantener vivo el intento.
  - Semántica de a lo más una vez.
    - Darse por vencido de inmediato.
  - No se garantiza nada.
    - Desde 0 a un número grande.

RPC – Fallas del cliente

- ¿Qué ocurre si un cliente envía una solicitud a un servidor para que este realice cierto trabajo y falla antes de que el servidor responda?
- En este momento esta activa una labor de computo y ningún padre espera el resultado.
  - Huérfano.

RPC – Fallas del cliente

- Solución 1.
  - El Client Stub crea una entrada que indica lo que va a hacer cada invocación.
  - ¿Qué hacer cuando el cliente vuelve a iniciar?
  - Exterminación.
    - Desventajas.

RPC – Fallas del cliente

- Solución 2.
  - Reencarnación.
  - División del tiempo en épocas secuenciales.
  - ¿Qué hacer cuando el cliente vuelve a iniciar?

RPC – Fallas del cliente

- Solución 3.
  - Reencarnación sutil.
  - Modificación de la reencarnación, pero se trata de ubicar a los poseedores de los cómputos remotos.
  - ¿Qué hacer cuando el cliente vuelve a iniciar?

RPC – Fallas del cliente

- Solución 4.
  - Expiración.
  - Se asigna una cantidad de tiempo estándar T.
  - ¿Qué ocurre si el procedimiento no devuelve antes de T?
  - ¿Qué hacer cuando el cliente vuelve a iniciar?

Gestión Distribuida de Procesos

- La migración de procesos es la posibilidad de mover un proceso activo de una máquina a otra.
- Considerar.
  - Coordinar actividades de procesos.
  - Sistemas diferentes.
  - Gobernados por un reloj local.
  - Retardo en el intercambio de información.

Migración de procesos

- Transferencia de una parte suficiente del estado de un proceso.
- Posibilidad de ejecutarse en otra máquina.
- Balanceo de cargas.
- “La migración real de procesos en ejecución es trivial en teoría, pero cerca de lo imposible en la práctica” (Andrew Tanenbaum).

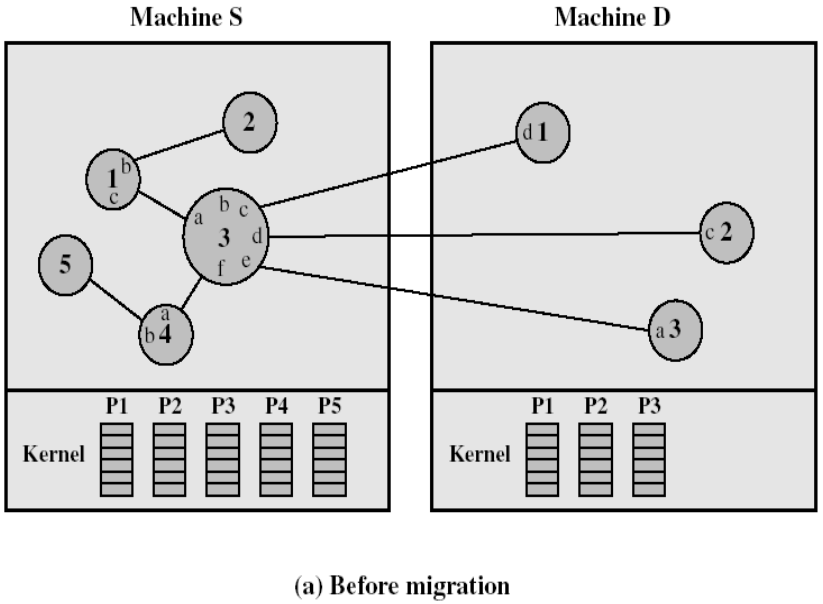
Migración de procesos

- Motivación:
  - Compartición de la carga.
  - Rendimiento de las comunicaciones.
  - Fiabilidad.
  - Utilización de características especiales.

Migración de procesos - Mecanismos

- Consideraciones:
  - ¿Quién da inicio a la migración?
  - ¿Qué “parte” del proceso emigra?
  - ¿Qué les ocurre a los mensajes y señales pendientes?

Migración de procesos - ¿Qué emigra?



(a) Before migration

Migración de procesos - ¿Qué emigra?

- El movimiento del PCB es sencillo.
- La dificultad recae en el movimiento del espacio de direcciones el proceso y en los archivos abiertos.

Migración de procesos - ¿Qué emigra?

- En cuanto al espacio de direcciones:
  - Transferir todo el espacio de direcciones en el momento de la migración.
  - Transferir sólo aquella parte del espacio de direcciones que reside en memoria principal.
  - Consideraciones en cuanto al manejo de hilos.
- En cuanto a los archivos abiertos.
  - Preguntar sobre archivos y caches.

Migración de procesos - ¿Qué emigra?

- Mensajes y señales.
  - ¿Qué ocurre con los mensajes y señales mientras dura la migración?
    - Ideas.
  - Almacenamiento temporal.

Un escenario de migración

- AIX de IBM (Automigración).
  1. Seleccionar una máquina destino, y enviar un mensaje de tarea remota.
     - ¿Qué información lleva el mensaje?
  2. En la máquina destino, un proceso servidor del núcleo crea un hijo y le cede el mensaje.
  3. El nuevo proceso extrae la información del mensaje, y es el encargado de replicar la imagen del proceso a emigrar.

Un escenario de migración

- AIX de IBM (Automigración).
  4. Se indica con una señal al proceso originario que la migración a terminado. Este proceso envía un mensaje final de terminación al nuevo proceso y se destruye.

Un escenario de migración

- ¿Cómo seria una migración, si el que inicia la migración no es el mismo proceso?
  - Ideas.

Un escenario de migración

- Cuando la migración la inicia otro proceso.
  1. Copiar la imagen del proceso y todo su espacio de direcciones a un archivo.
  2. Destruir el proceso a migrar.
  3. Copiar el archivo a otra máquina vía una transferencia de archivos.
  4. Volver a crear el proceso en la nueva máquina, a partir del archivo.

Negociación de la migración

- ¿A qué nos referimos con esto?
  - Ideas.
- Concepto de entidad iniciadora.
  - Starter.

Negociación de la migración

1. El iniciador que controla el sistema origen (S) decide que un proceso (P) debe emigrar a un sistema destino determinado (D). Entonces envía un mensaje al Iniciador de D solicitando la transferencia.
2. Si el iniciador de D está preparado para recibir al proceso, devuelve un acuse de recibo afirmativo.

Negociación de la migración

3. El iniciador de S le comunica su decisión al núcleo de S, a través de la llamada a un servicio (si el iniciador se esta ejecutando en S) o mediante un mensaje al KernJob (KJ) de la máquina S.
4. El núcleo de S se ofrece entonces para enviar el proceso D. En la oferta se incluyen estadísticas sobre P.

Negociación de la migración

5. Si D anda escaso de recursos, puede rechazar la oferta. En otro caso, el núcleo de D propone la oferta a su iniciador. En la propuesta se incluye la misma información recibida de S.
6. La decisión según la política del iniciador es comunicada a D por medio de una llamada MigrateIn.

Negociación de la migración

7. D reserva los recursos necesarios y envía a S una aprobación.

Negociación de la migración

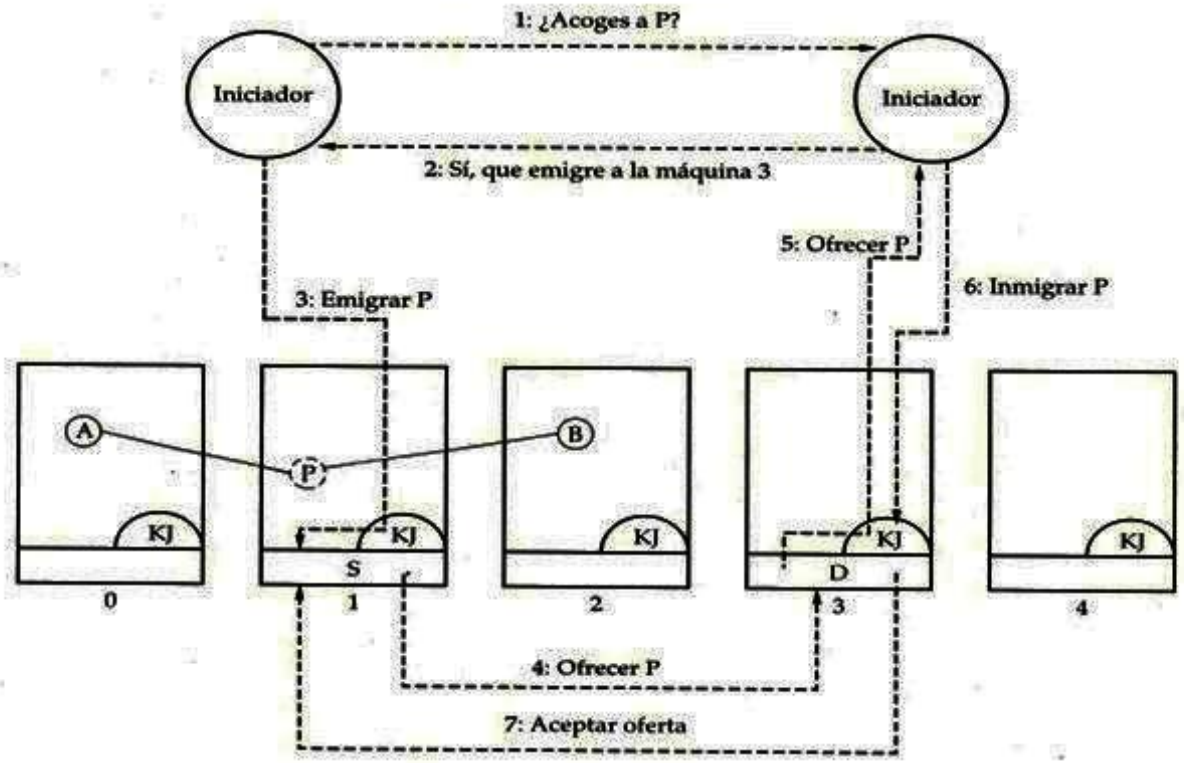


FIGURA 13.2 Negociación de la migración de procesos

Desalojo de procesos

- El proceso de negociación permite que un sistema destino rechace la migración.
- Adicionalmente, puede ser útil que un sistema desaloje un proceso que ha emigrado hacia él.
  - ¿Bajo que circunstancias?
- El sistema operativo SPRITE es un ejemplo.

Desalojo de procesos

- En SPRITE.
  - Un proceso esta casado con una única máquina.
    - Nodo de origen.
  - Si un proceso migra, se convierte en un proceso extranjero.
    - Nodo destino.

Desalojo de procesos

1. Un proceso supervisor lleva la cuenta de carga actual para determinar cuándo se pueden aceptar nuevos procesos extranjeros. Si el supervisor detecta actividad en dicha estación, se inicia un procedimiento de desalojo para cada proceso extranjero.
2. El proceso desalojado volverá a su nodo de origen.

Desalojo de procesos

3. El desalojo se realiza para todos los procesos extranjeros en el nodo.
   - Consideraciones.
4. El espacio de direcciones por completo de un proceso desalojado es transferido al nodo de origen.
   - Consideraciones.

Transferencias Apropiativas y No Apropiativas

- ¿A qué nos referimos?
  - Parcialmente ejecutado o creación finalizada.
  - Proceso que aún no han comenzado se ejecución.
- Ventajas y desventajas.

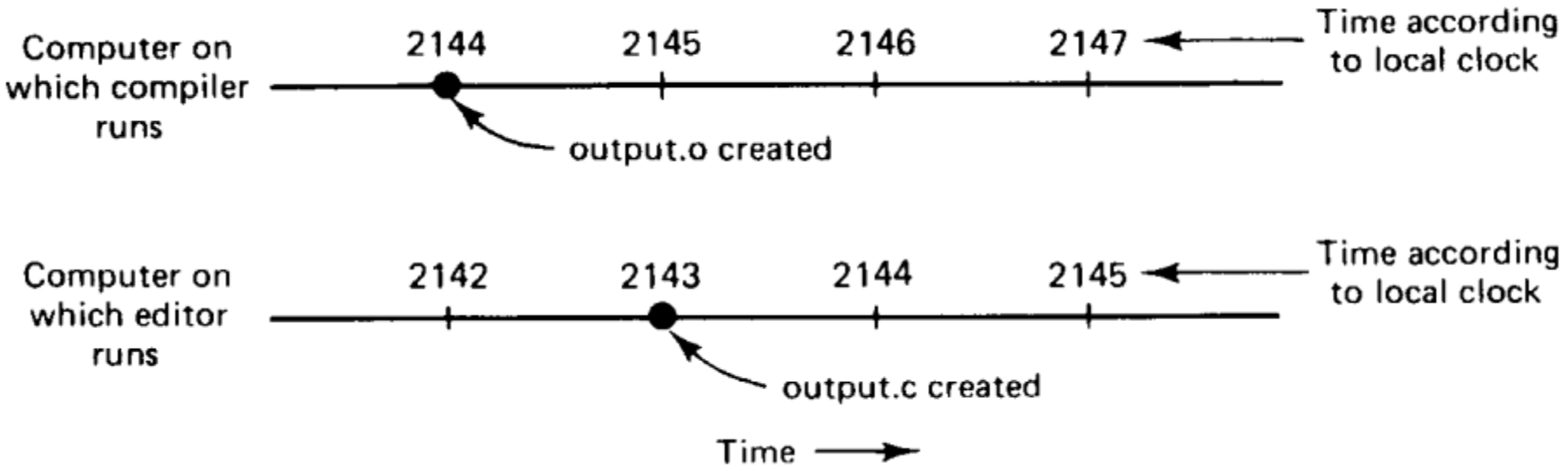
Sincronización

- ¿Cómo es la comunicación en un sistema distribuido?
- ¿Cómo se ataca la sincronización en los sistemas convencionales?
- ¿Por qué no hacer lo mismo acá?
  - Premisa en la existencia de memoria compartida.

Sincronización

- Características de los algoritmos distribuidos:
  - La información relevante se encuentra distribuida.
  - Los procesos toman decisiones sólo con base a la información local.
  - Debe evitarse un único punto de fallo en el sistema.
  - No existe un reloj común o alguna otra fuente precisa del tiempo global.

Sincronización de relojes



- ¿Es posible sincronizar todos los relojes en un sistema distribuido?

Relojes

- La mayoría de las computadoras poseen un circuito para el registro del tiempo.
  - Reloj vs. Cronómetro.
- Cristal de cuarzo trabajado con precisión.
  - Tensión → Oscilación a un frecuencia.

Relojes

- A cada cristal se le asocian dos registros:
  - Contador.
  - Registro mantenedor.
- ¿Cómo controlar el número de interrupciones?
- Cada interrupción recibe el nombre de marca de reloj.

Relojes

- ¿Cuál es el problema con los relojes?
- Distorsión de reloj.
- Ideas para solventar esta situación.

Relojes

- La sincronización no tiene que ser absoluta.
- ¿Qué pasa si dos procesos no interactúan?
- “Lo que importa por lo general, no es que todos los procesos concuerden de manera exacta en la hora, sino que coincida en el orden en que ocurren los eventos” (Lamport, 1990)

Relojes

- Relojes lógicos.
  - La importancia radica en la consistencia interna de los relojes, no su particular cercanía al tiempo real.
- Relojes físicos.
  - Existe una consistencia interna, y además un umbral permitido de discrepancia con el tiempo real.

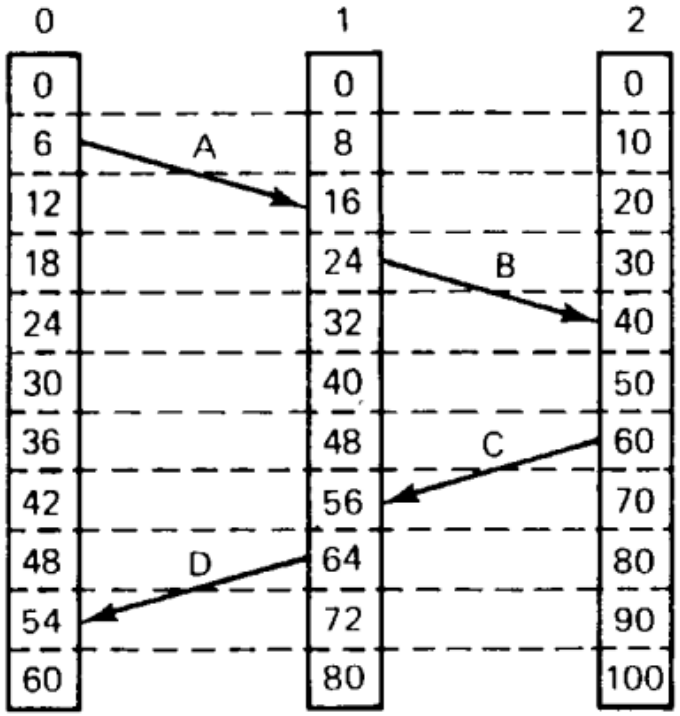
Relojes lógicos

- Lamport definió una relación llamada ocurre antes de.
  - A → B “A ocurre antes que B”
  1. Si A y B son eventos en el mismo proceso y A ocurre antes que B, entonces A → B.
  2. Si A es el evento del envío de un mensaje por un proceso y B es el evento de la recepción del mensaje, entonces A → B.
  3. Si A → B y B → C, entonces A → C.

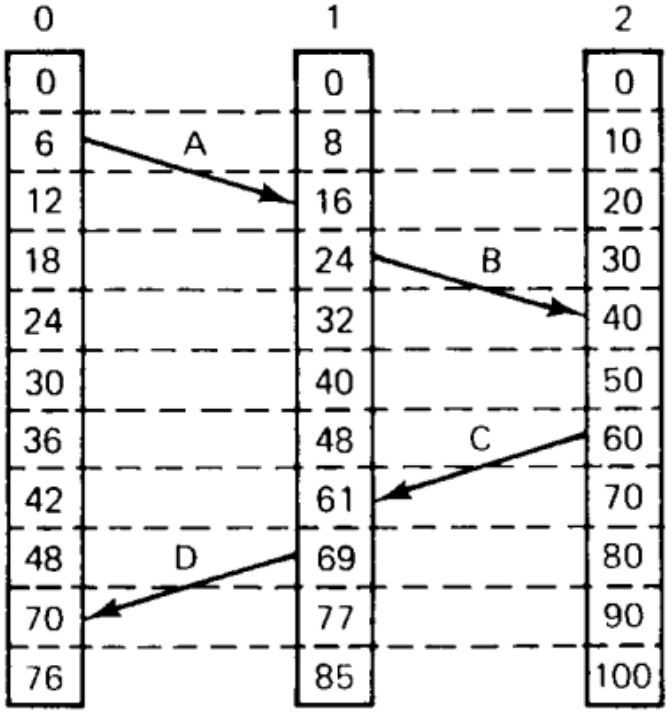
Relojes lógicos

- Lo que necesitamos es una forma de medir el tiempo tal que, a cada evento A, le podamos asociar un valor de tiempo C(A) en el que todos los procesos estén de acuerdo.
- Si A → B, entonces C(A) < C(B).
- Adicionalmente:
  - El tiempo C siempre debe ser creciente, nunca decreciente.

Relojes lógicos



(a)



(b)

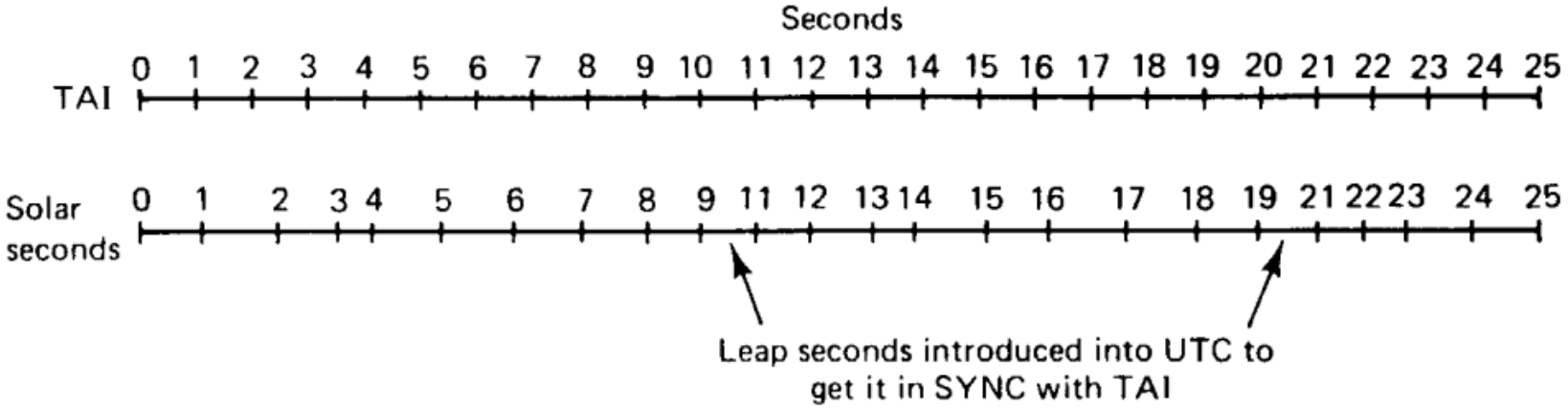
Relojes lógicos

1. Si A ocurre antes de B en el mismo proceso, $C(A) < C(B)$.
2. Si A y B son el envío y la recepción de un mensaje, $C(A) < C(B)$.
3. Para todos lo eventos A y B, $C(A) \neq C(B)$.

Relojes físicos

- Mediciones con base al sol.
- Reloj atómico → 1948.
- Tiempo Atómico Internacional (TAI).
- Coordinated Universal Time (UTC).

Relojes físicos



- Ejemplo NTP.

Deadlock en SD

- Objetivos.
  - Comprender y describir el problema de abrazo mortal distribuido.
  - Diferenciar el abrazo mortal por recursos y por comunicación.
  - Comprender y comparar diferentes enfoques para manejar el problema de abrazo mortal.

Deadlock en SD

- “Los bloqueos en los sistemas distribuidos son similares a los bloqueos en los sistemas con un procesador, sólo que peores” (Tanenbaum).

Deadlock en SD

- Dos tipos de bloqueos distribuidos:
  - Por comunicación.
    - A envía B envía C envía A.
  - Por recursos.
    - Acceso exclusivo a dispositivos de E/S, archivos, etc.
- Al final esta distinción es opcional.
  - Un buffer o canal de comunicación puede manejarse como un recurso.

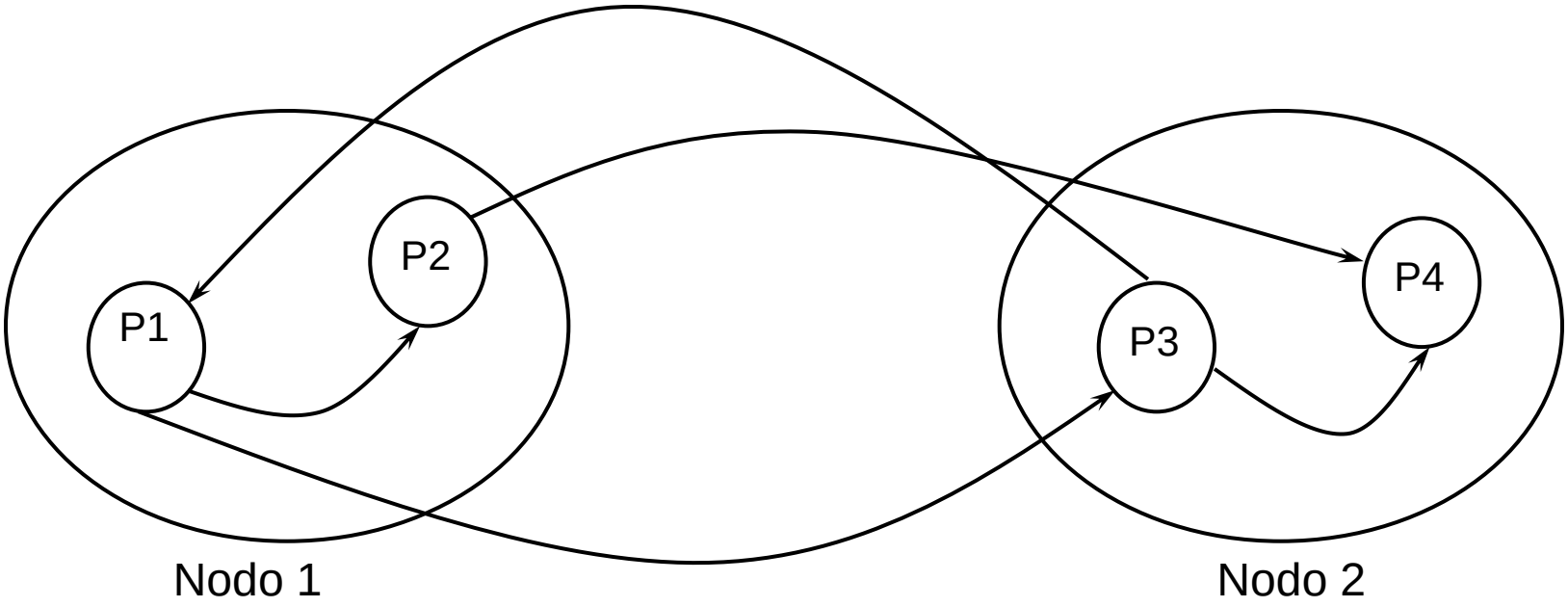
Deadlock en SD

- Estrategias para el manejo:
  1. Algoritmo del avestruz.
  2. Detección.
  3. Prevención.
  4. Evitarlos.
- Nos centraremos en:
  - Detección y prevención.
  - ¿Por qué?

Deadlock en SD

- Definición.
  - Es una situación en la que se encuentran un conjunto de procesos donde cada proceso está esperando por un evento que sólo puede ser producido por otro proceso perteneciente a ese mismo conjunto.

Deadlock en SD



Detección centralizada de deadlock

- Utilizar un algoritmo centralizado para la detección de bloqueos y tratar de imitar al algoritmo no distribuido.
- Cada nodo construye su grafo de asignación de recursos.
- Un nodo especial llamado coordinador, mantiene el grafo de asignación de recursos de todo el sistema.

Detección centralizada de deadlock

- ¿Cómo se construye y mantiene el grafo global del sistema?
  - Se envía un mensaje al coordinador por cada arista agregada o eliminada en cualquier grafo.
  - Cada proceso puede enviar de manera periódica una lista de aristas agregadas o eliminadas desde la última actualización.
  - El coordinador puede solicitar la información cuando la necesite.

Detección centralizada de deadlock

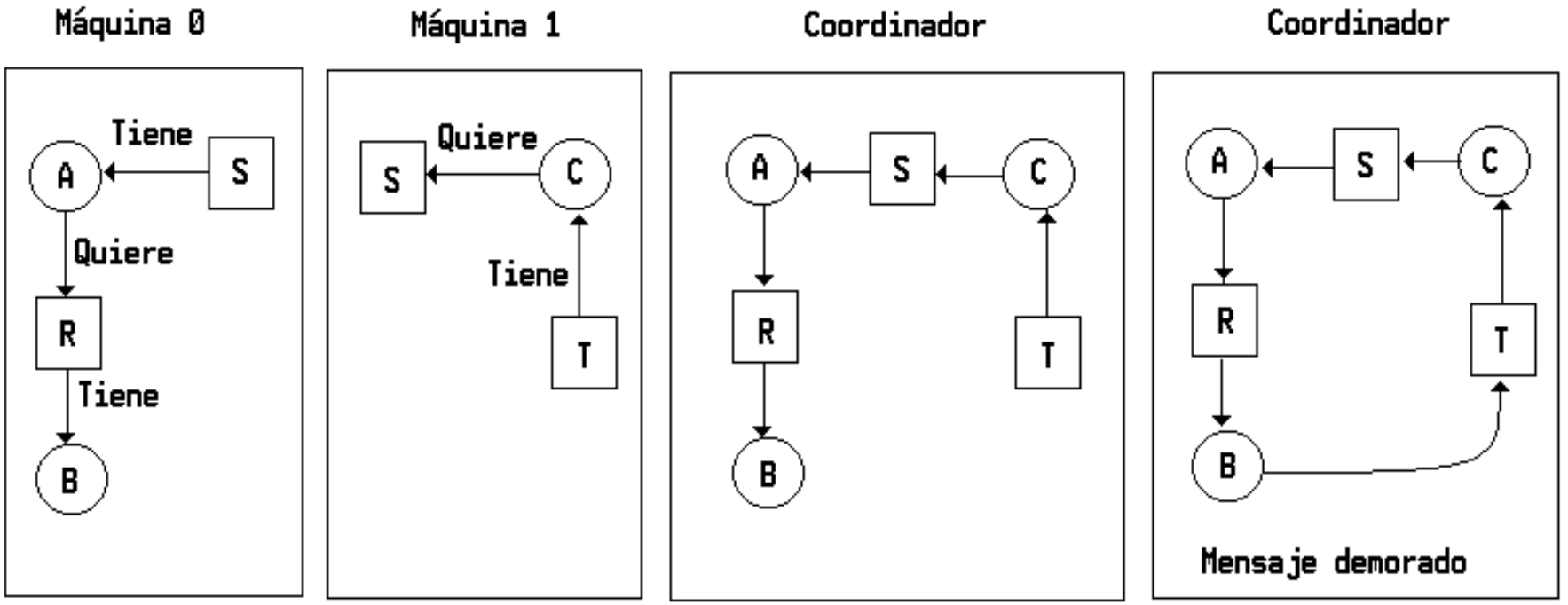


Fig.23.14. - Detección centralizada de Abrazo Mortal.

Detección centralizada de deadlock

- Falso bloqueo.
  - Información incompleta o con retraso.
- Soluciones.
  - Ideas.
  - ¿Cómo funciona su idea?
    - Descríbala.

Detección centralizada de deadlock

- Características.
  - El nodo de control puede mantener el grafo global constantemente o lo puede construir en un momento dado.
  - Es conceptualmente simple y fácil de implementar.
  - La resolución del deadlock es optima ya que el nodo de control tiene toda la información y puede tomar la decisión más acertada.

Detección centralizada de deadlock

- Desventajas.
  - Embotellamiento generado alrededor del nodo de control o coordinados.
  - Punto único de falla.

Detección distribuida de deadlock

- La responsabilidad de detectar el deadlock es compartida equitativamente entre todos los nodos.
- Estudiaremos el algoritmo de:
  - Chandy-Misra-Hass (1983).

Chandy-Misra-Hass

- Se utiliza cuando un proceso debe esperar por cierto recurso.
- Se utiliza un mensaje especial.
  - Mensaje de exploración.
  - El mensaje consta de:
    - El proceso recién bloqueado.
    - El proceso que envía el mensaje.
    - El proceso a cual se envía.

Chandy-Misra-Hass

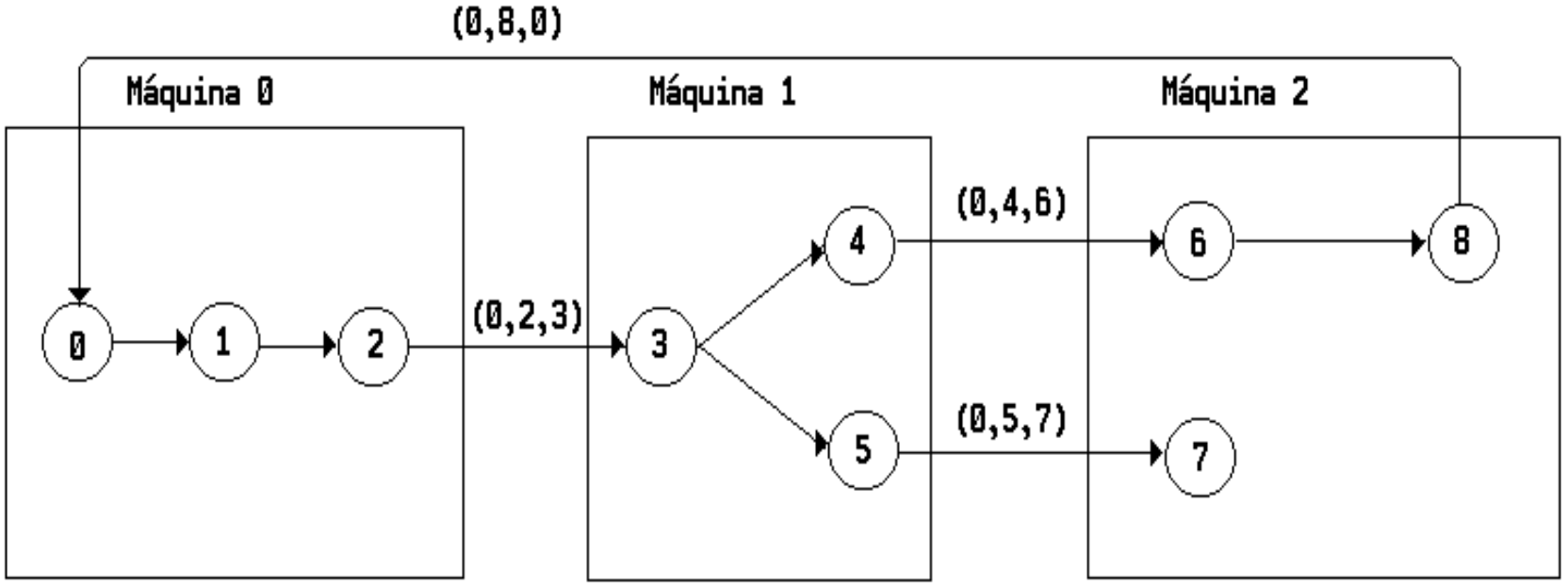


Fig. 23.16. - Detección distribuida.

Algoritmo Chandy-Misra-Hass

- Todos los procesos poseen en su memoria local las siguientes estructuras para poder llevar a cabo el algoritmo propuesto:
  - **ultimo(i):** en este arreglo se almacena el número de secuencia más grande en cualquier consulta enviada o recibida por Pk (el valor de m más grande).
  - **IDUltimo(i):** almacena el Id del proceso que envió el mensaje más reciente, y cuyo valor m se almacenó en Ultimo(i).
  - **num(i):** almacena la cantidad de mensajes consulta(i,m,k,j) que envió Pk y aún no han sido respondidos con su correspondiente mensaje respuesta(i,m,j,k). Si num(i) es 0, significa que Pk ha recibido respuesta de cada una de las consultas enviadas.
  - **espera(i):** es verdadero si Pk está ocioso. Se hace falso cuando Pk pasa a estado En Ejecución.
- Cada proceso p almacena localmente su conjunto dependiente en el arreglo dependientes(p).

Algoritmo Chandy-Misra-Hass

Algoritmo Pseudoformal

Var ultimo: arreglo [1..i] de entero **inicializa** 0;
IDultimo: arreglo [1..i] de entero;
num: arreglo [1..i] de entero;
espera: arreglo [1..i] de lógico **inicializa falso**

Cuando Pi inicia la consulta **hacer**
ultimo(i):=ultimo(i)+1;
espera(i):=**verdadero**;
paratodo (Pj / Pj € dependientes(i)) **hacer**
enviar consulta(i,ultimo(i),i,j) **a** Pj;
fparatodo
num(i):=card(dependientes(i));

fin

Algoritmo Chandy-Misra-Hass

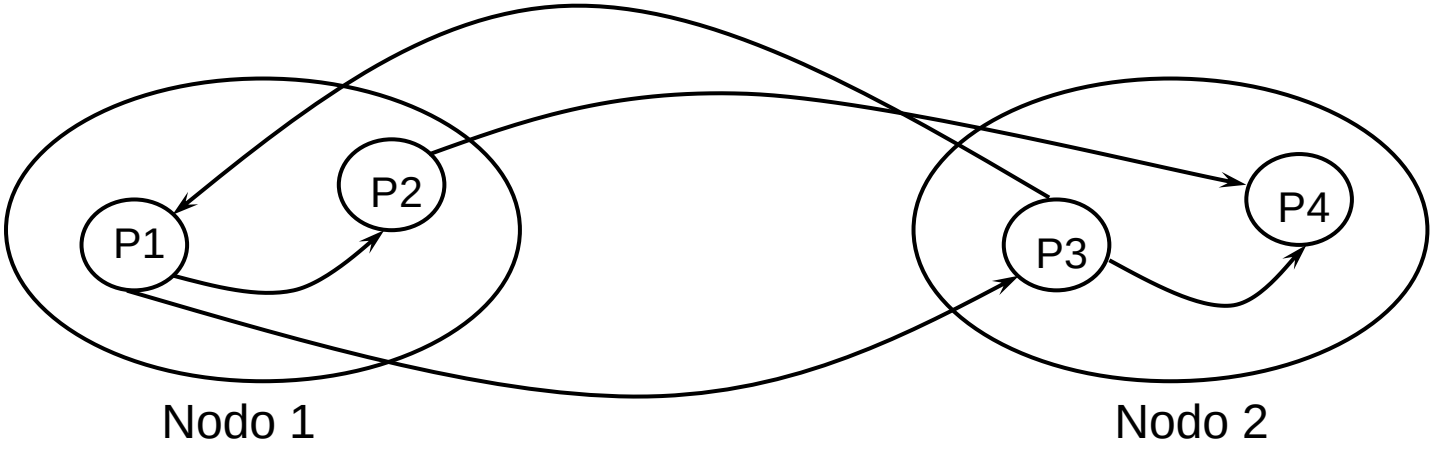
```
Cuando Pi está en estado En Ejecución hacer
        espera(i):=falso;
fin

Cuando Pk está en estado Ocioso y recibe el mensaje consulta(i,m,j,k) hacer
        Si m>ultimo(i) entonces
                ultimo(i):=m;
                IDultimo(i):=j;
                espera(i):=verdadero;
                paratodo (Pr / Pr € dependientes(k)) hacer
                        enviar consulta(i,m,k,r) a Pr;
                fparatodo
                num(i):=card(dependientes(k));
        sino
                si espera(i) y m=ultimo(i) entonces
                        enviar respuesta(i,m,k,j) a Pj;
                fsi
        fsi
fin
```

Algoritmo Chandy-Misra-Hass

```
Cuando Pk está en estado Ocioso y recibe el mensaje respuesta(i,m,r,k) hacer
        Si m=ultimo(i) y espera(i) entonces
                num(i):=num(i)-1;
                si num(i)=0 entonces
                        si i=k entonces
                                Pk es deadlocked
                        sino
                                enviar respuesta(i,m,k,j)
                                {donde j=IDUltimo(i)}
                        fsi
                fsi
        fsi
fin
```

Algoritmo Chandy-Misra-Hass



- Asignación.
  - Realizar la corrida del algoritmo de Chandy-Misra-Hass, con el estado inicial de la figura.

Chandy-Misra-Hass

- Características.
  - Cada nodo mantiene una porción del grafo global.
  - Cada nodo participa en la detección de un ciclo global.
  - Es más resistentes a fallas, y ningún nodo está sobrecargado con la detección de deadlock.

Chandy-Misra-Hass

- Desventajas.
  - Debido a las estructuras de datos su diseño resulta muy complejo.

Detección jerárquica de deadlock

- Los nodos se organizan en forma jerárquica y cada nodo detecta deadlocks en los que estén involucrados sus descendientes.

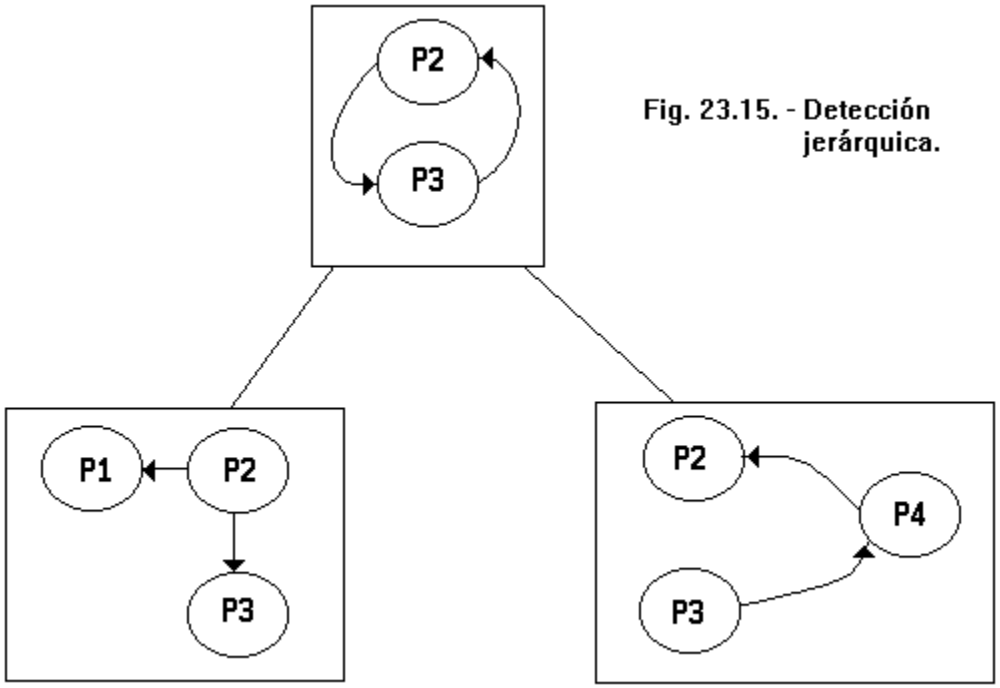


Fig. 23.15. - Detección jerárquica.

Detección jerárquica de deadlock

- Características:
  - Es el intermedio entre el centralizado y el distribuido.
  - Explota los patrones de comunicación locales a un grupo de nodos para detectar el deadlock.
  - No es dependiente de la falla de un nodo.
  - Un nodo no está sobrecargado por la detección de deadlock en los cuales muy posiblemente no está involucrado.

Resolución del deadlock

- La persistencia de un abrazo mortal tiene dos efectos negativos sobre el rendimiento de un sistema:
  - Los recursos en manos de procesos interbloqueados no están disponibles a otros procesos.
  - El tiempo mientras persista el deadlock se suma al tiempo de respuesta de cada proceso interbloqueado.

Resolución del deadlock

- La espera por recursos agrega arcos y nodos al grafo de asignación de recursos.
- La resolución de deadlock remueve arcos y nodos del grafo de asignación de recursos.

Prevención distribuida de deadlock

- Técnicas clásicas:
  - Conservar sólo un recurso a la vez.
  - Exigir que los procesos soliciten todos los recursos desde un inicio.
  - Liberar todos los recursos retenidos cuando se solicita uno nuevo.
  - Ordenar los recursos.
    - Adquisición de recursos en orden creciente o decreciente.

Prevención distribuida de deadlock

- Algoritmo Espera-Muere.

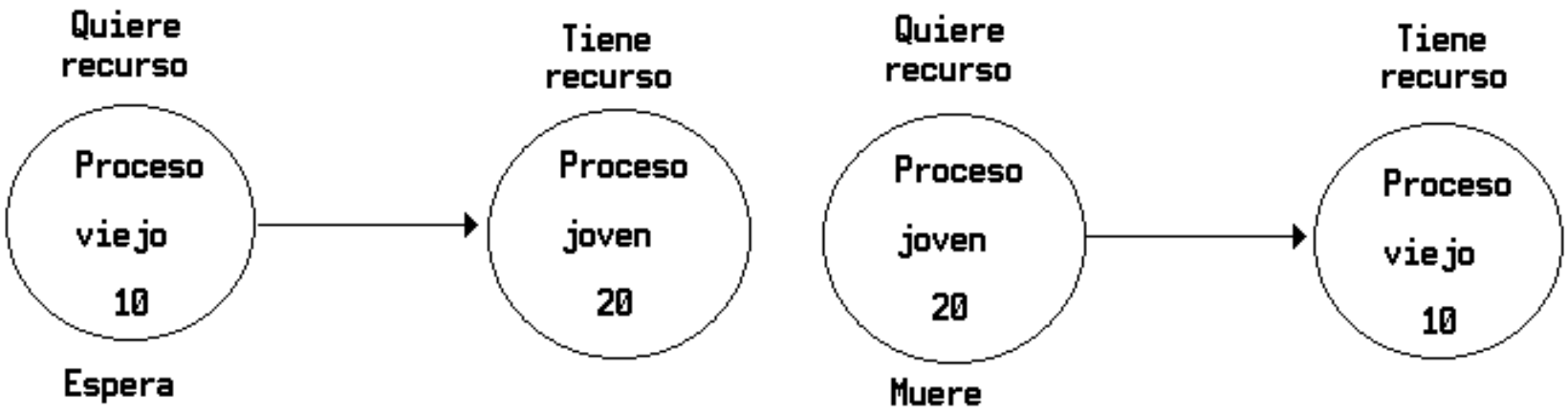


Fig. 23.17. - El algoritmo Wait-Die.

Prevención distribuida de deadlock

- Algoritmo Herida-Espera.

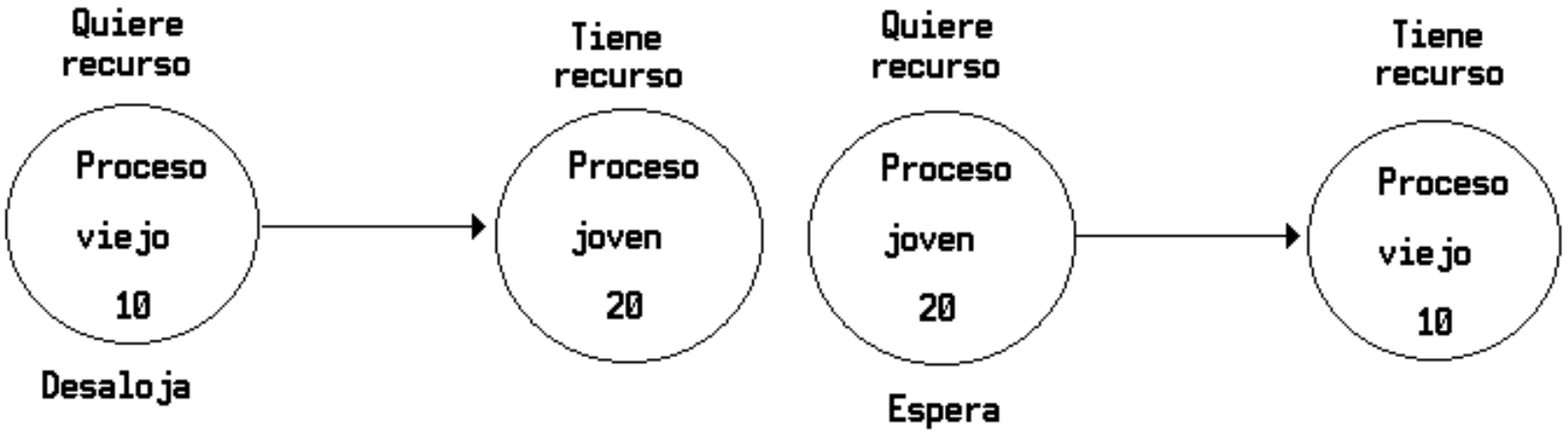


Fig. 23.18. - El algoritmo Wound-Wait.

Memoria compartida distribuida

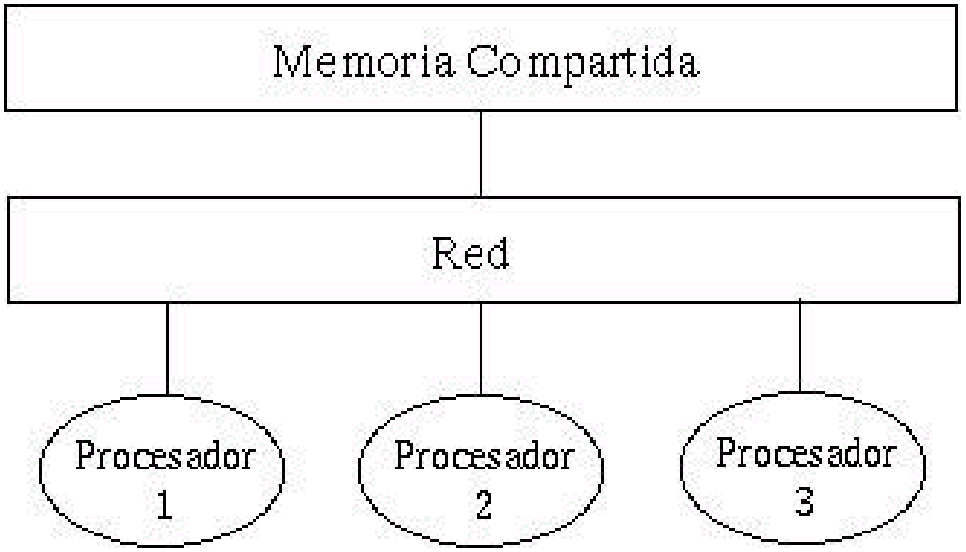
- Una clasificación de los sistemas MIMD respecto a su memoria es:
  - Sistemas de memoria compartida.
  - Sistemas de memoria distribuida.
  - Sistemas de memoria compartida y distribuida.
- Ideas de cada uno.

Memoria compartida distribuida

- En 1986 Li y Hudak propusieron un esquema de memoria conocido como:
  - Memoria compartida distribuida.
    - Colección de estaciones de trabajo.
    - Conectadas a través de una LAN.
    - Compartiendo un solo espacio de direcciones virtuales con páginas.

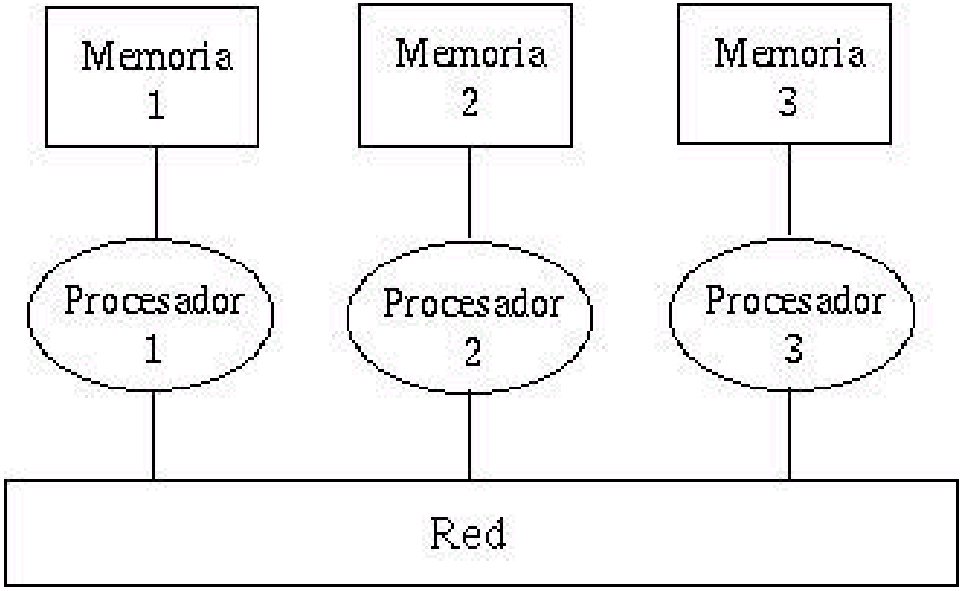
Memoria compartida distribuida

- Sistemas de memoria compartida.



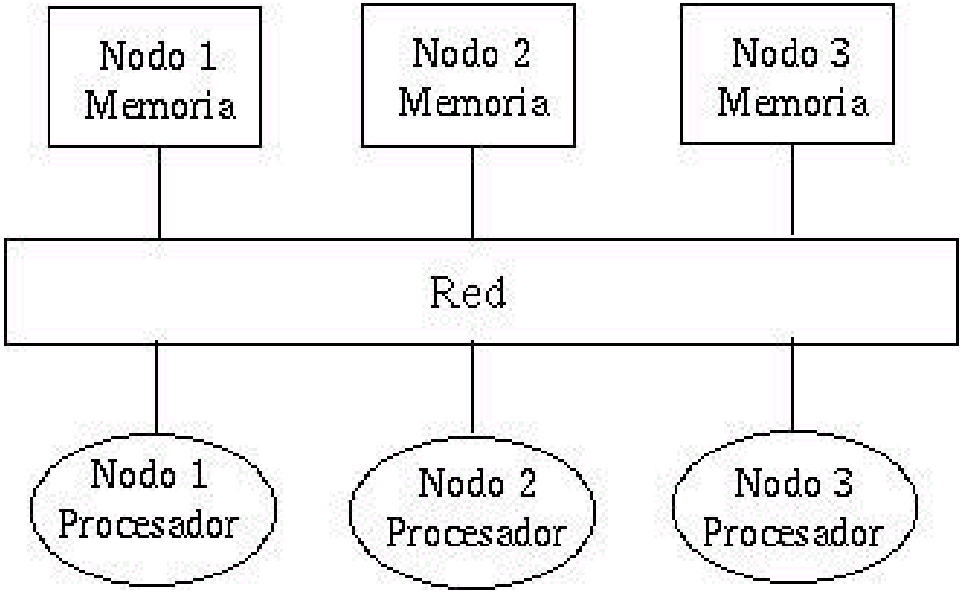
Memoria compartida distribuida

- Sistemas de memoria distribuida.



Memoria compartida distribuida

- Sistemas de memoria compartida distribuida.



Memoria compartida distribuida

- ¿Qué es la memoria compartida?

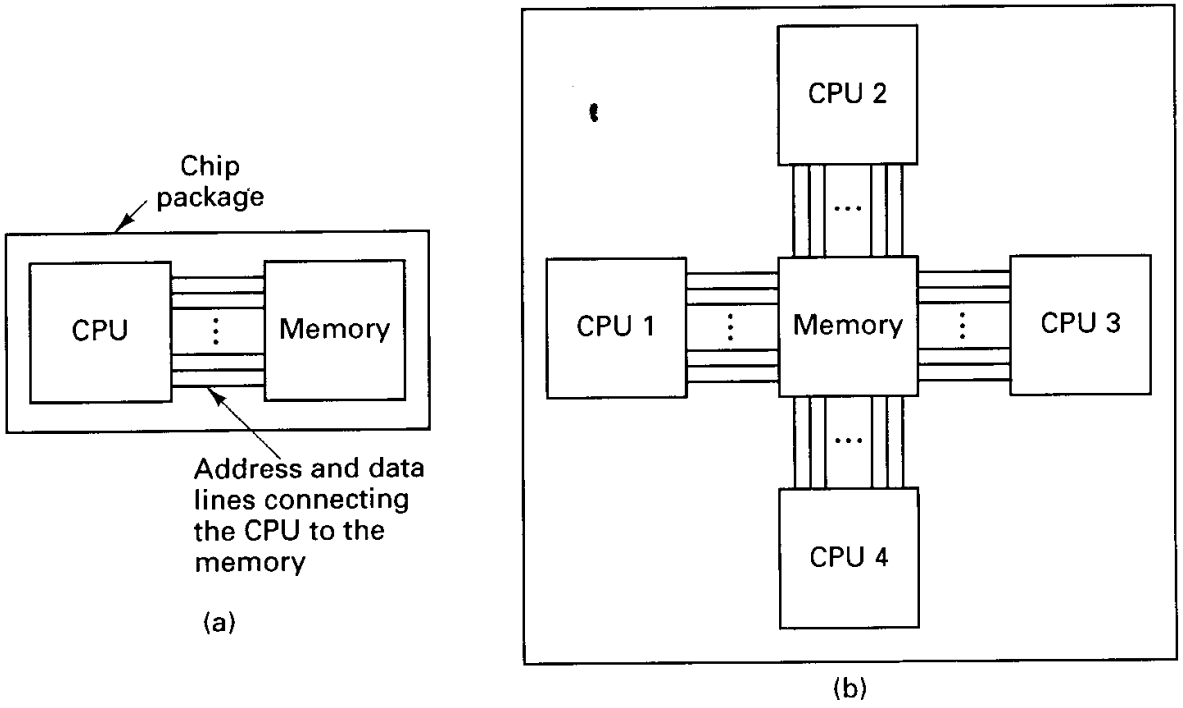


Fig. 6-1. (a) A single-chip computer. (b) A hypothetical shared-memory multiprocessor.

Memoria compartida distribuida

- ¿Qué es la memoria compartida?

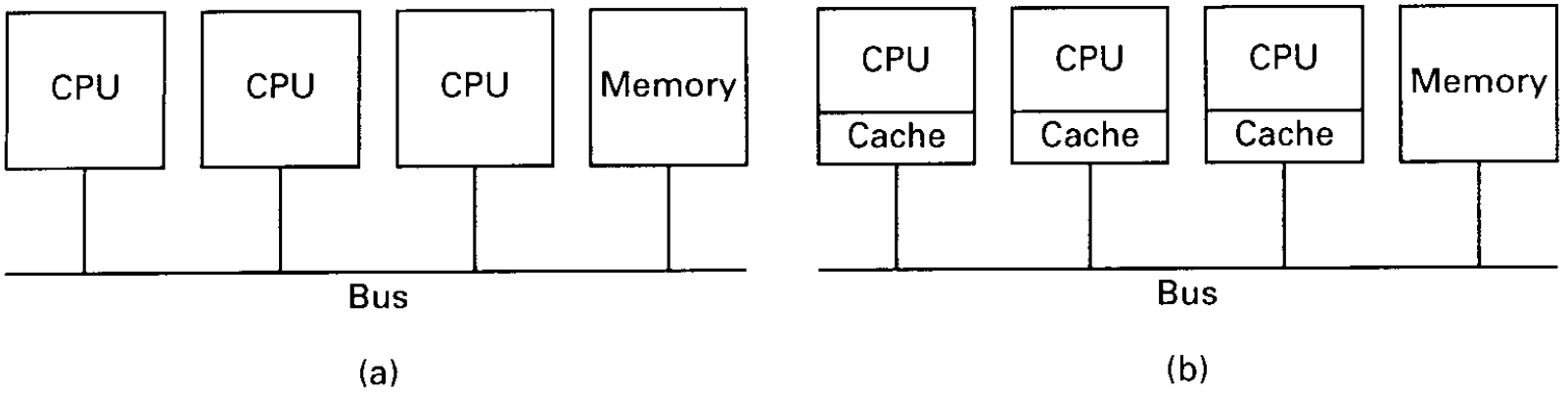


Fig. 6-2. (a) A multiprocessor. (b) A multiprocessor with caching.

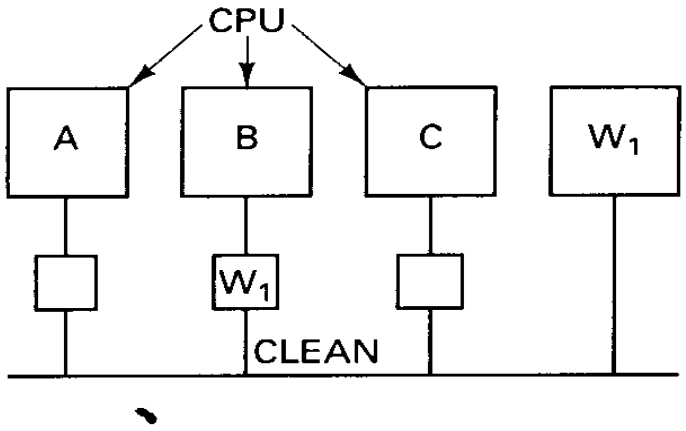
Memoria compartida distribuida

- Snoopy cache

Event	Action taken by a cache in response to its own CPU's operation	Action taken by a cache in response to a remote CPU's operation
Read miss	Fetch data from memory and store in cache	(No action)
Read hit	Fetch data from local cache	(No action)
Write miss	Update data in memory and store in cache	(No action)
Write hit	Update memory and cache	Invalidate cache entry

Memoria compartida distribuida

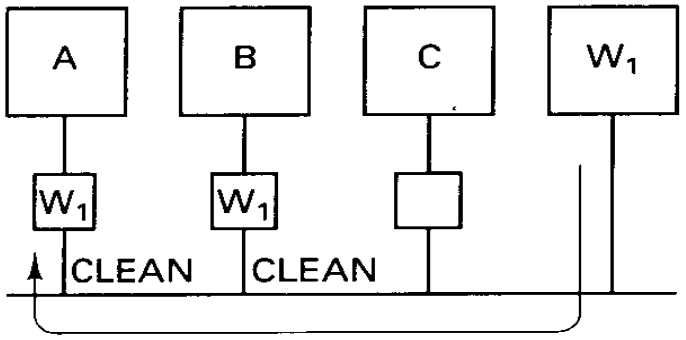
- Necesitamos un protocolo de manejo de bloques de cache.
  - Ideas.
- Piense en los siguientes estados:
  - Inválido.
    - Datos inválidos en cache.
  - Limpio.
    - La memoria esta actualizada.
  - Sucio.
    - La memoria es incorrecta.



Memory is correct

Initial state.
Word W containing value W_1 is in memory and is also cached by B.

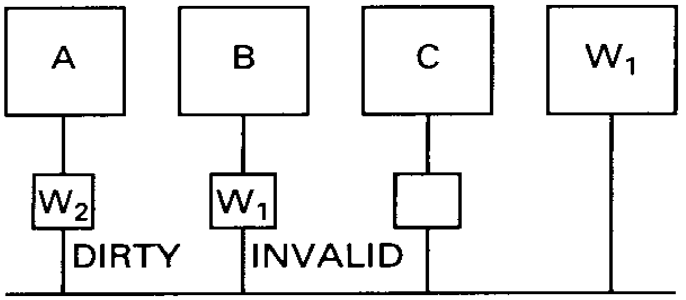
(a)



Memory is correct

A reads word W and gets W_1. B does not respond to the read, but the memory does.

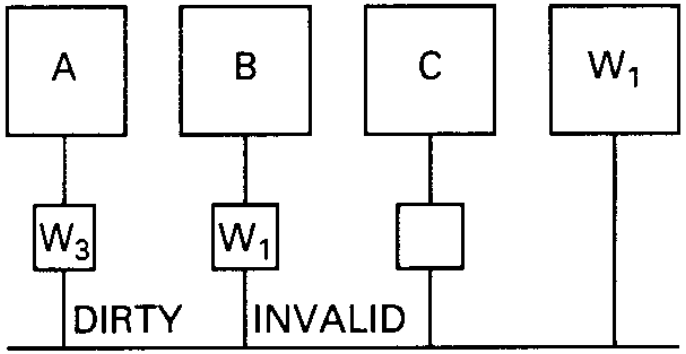
(b)



Memory is correct

A writes a value W_2. B snoops on the bus, sees the write, and invalidates its entry. A's copy is marked DIRTY.

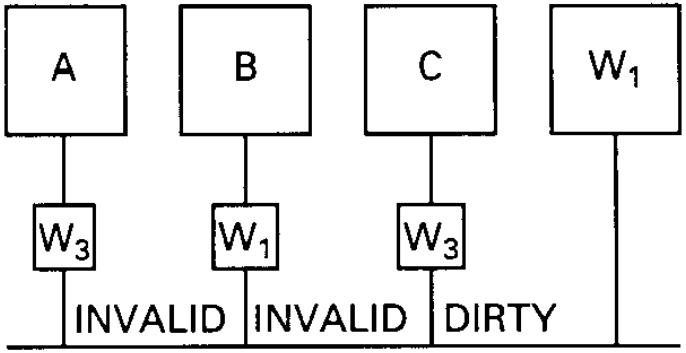
(c)



Memory is correct

A writes W again. This and subsequent writes by A are done locally, without any bus traffic.

(d)



Memory is correct

C reads or writes W. A sees the request by snooping on the bus, provides the value, and invalidates its own entry. C now has the only valid copy.

(e)

Memoria compartida distribuida

- Protocolo de manejo de bloques de cache.
  - Propiedades:
    1. La consistencia se logra haciendo que todos los cachés husmen el bus.
    2. El protocolo se integra dentro de la unidad de administración de memoria.
    3. Todo el algoritmo se realiza en un ciclo de memoria.

Modelos de consistencia

- ¿Cómo se trabaja con la memoria en un sistema de memoria compartida distribuida?
  - Copias de lectura.
  - Copias de escritura.
  - Consideraciones.

Modelos de consistencia

- “Un modelo de consistencia es en esencia un contrato entre el software y la memoria” (Adve y Hill, 1990).
- ¿Qué ocurre si se obedecen las reglas?
- ¿Qué ocurre si se violan las reglas?

Modelos de consistencia

- Estricta.
- Secuencial.
- Causal.
- PRAM.
- Débil.
- Liberación.
- Entrada.

Consistencia estricta

- Cualquier lectura a una localidad de memoria *x* regresa el valor guardado por la operación de escritura más reciente en *x*.
- Consideraciones.

Consistencia estricta

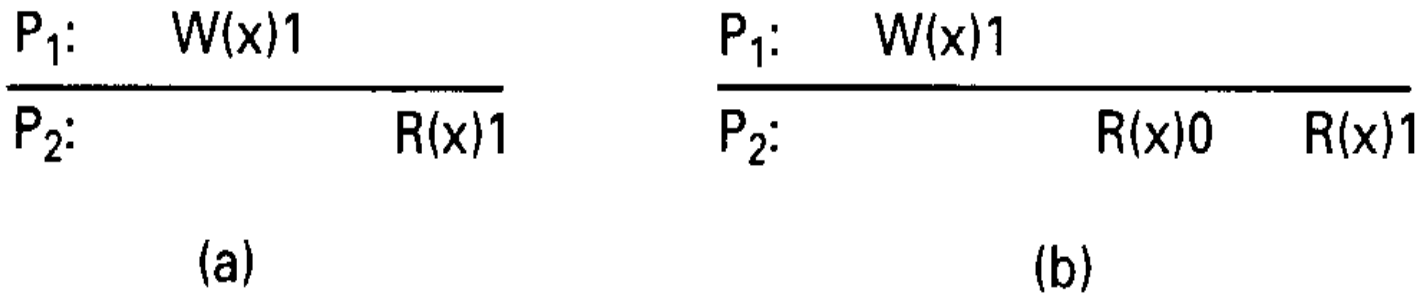


Fig. 6-12. Behavior of two processes. The horizontal axis is time. (a) Strictly consistent memory. (b) Memory that is not strictly consistent.

Consistencia secuencial

- “El resultado de cualquier ejecución es el mismo que si las operaciones de todos los procesos fueran ejecutadas en algún orden secuencial, y las operaciones de cada proceso individual aparecen en el orden especificado por su programa” (Lamport 1979).
- Cuando los procesos se ejecutan en paralelo en diferentes máquinas, cualquier intercalado valido es aceptable, pero los procesos deben ver las mismas referencias a memoria en el mismo orden.

Consistencia secuencial

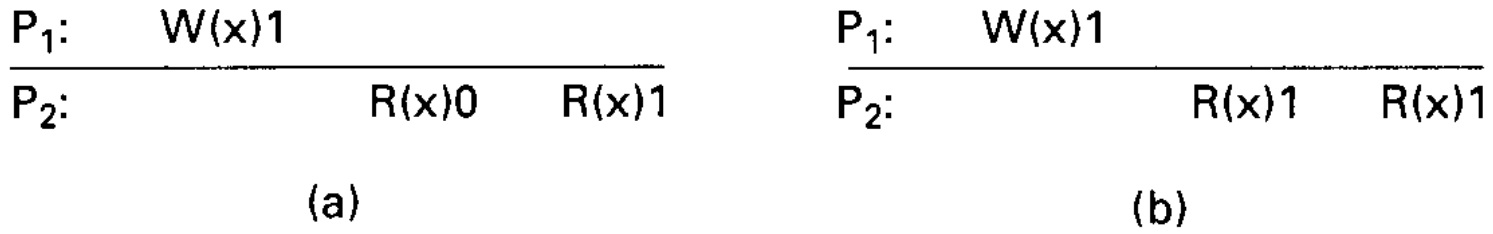


Fig. 6-13. Two possible results of running the same program.

```
a = 1;                b = 1;                c = 1;
print (b, c);         print (a, c);         print (a, b);
```

(a) (b) (c)

Fig. 6-14. Three parallel processes.

Consistencia secuencial

```
a = 1;
print (b, c);
b = 1;
print (a, c);
c = 1;
print (a, c);1
```

Prints: 001011

Signature: 00101

(a)

```
a = 1;
b = 1;
print (a, c);
print (b, c);
c = 1;
print (a, b);
```

Prints: 101011

Signature: 101011

(b)

```
b = 1;
c = 1;
print (a, b);
print (a, c);
a = 1;
print (b, c);
```

Prints: 010111

Signature: 110101

(c)

```
b = 1;
a = 1;
c = 1;
print (a, c);
print (b, c);
print (a, b);
```

Prints: 111111

Signature: 111111

(d)

Fig. 6-15. Four valid execution sequences for the program of Fig. 6-14. The vertical axis is time, increasing downward.

Consistencia secuencial

- Método formal para expresar la consistencia secuencial. (Ahamad et al., 1993).
  - La serie de operaciones de lectura y escritura de un proceso i se denotan por H_i.
  - Para obtener el orden relativo se fusionan las cadenas de operación de cada H_i en una nueva cada S.
  - En S se deben cumplir dos condiciones:
    - Mantenerse el orden del programa.
    - Debe ser respetada la coherencia en la memoria.

Consistencia causal

- “Las escrituras potencialmente relacionadas de forma causal son vistas por todos los procesos en el mismo orden. Las escrituras concurrentes pueden ser vistas en un orden diferente en máquinas diferentes” (Hutto y Ahamad, 1990).

Consistencia causal

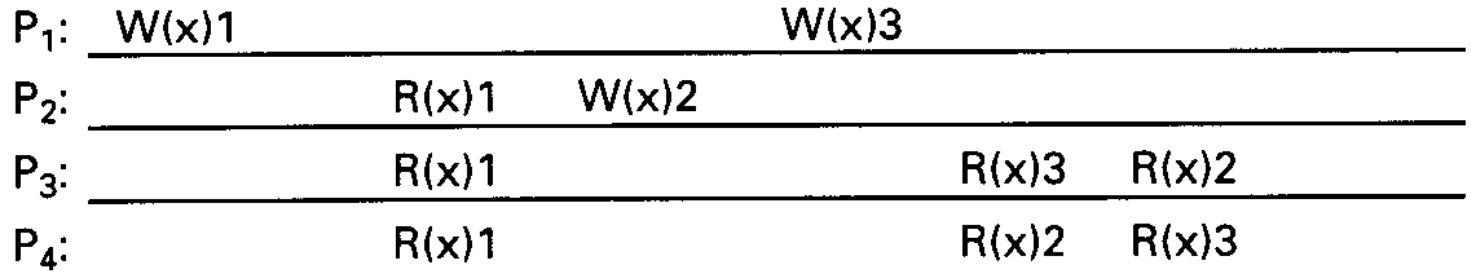
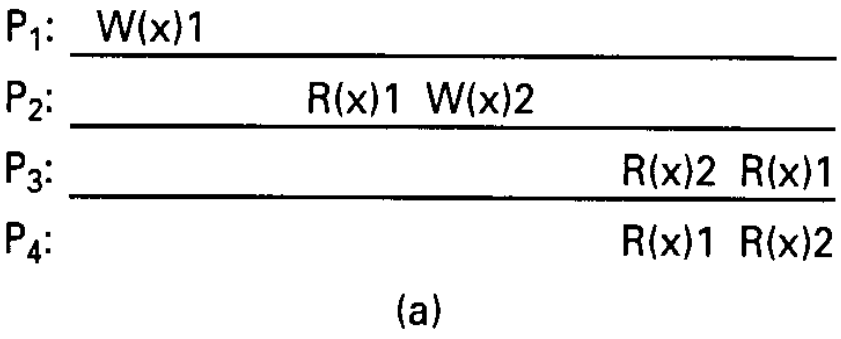


Fig. 6-16. This sequence is allowed with causally consistent memory, but not with sequentially consistent memory or strictly consistent memory.

Consistencia causal



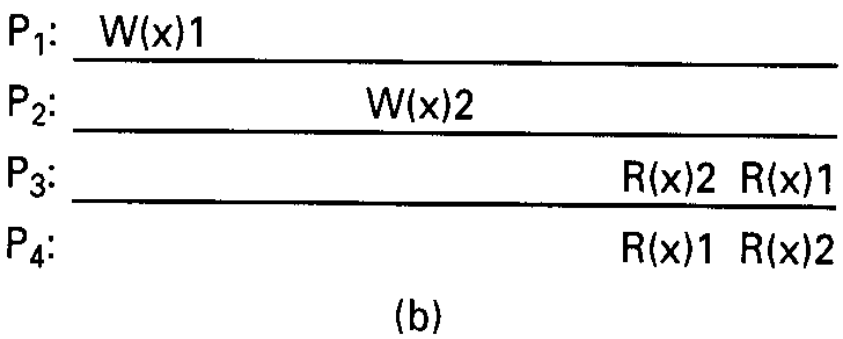


Fig. 6-17. (a) A violation of causal memory. (b) A correct sequence of events in causal memory.

Consistencia PRAM

- PRAM (Pipelined RAM).
  - “Las escrituras realizadas por un proceso son recibidas por los otros procesos en el orden en que son realizadas, pero las escrituras de procesos diferentes pueden verse en un orden diferente por procesos diferentes” (Lipton y Sandberg, 1988).

Consistencia PRAM

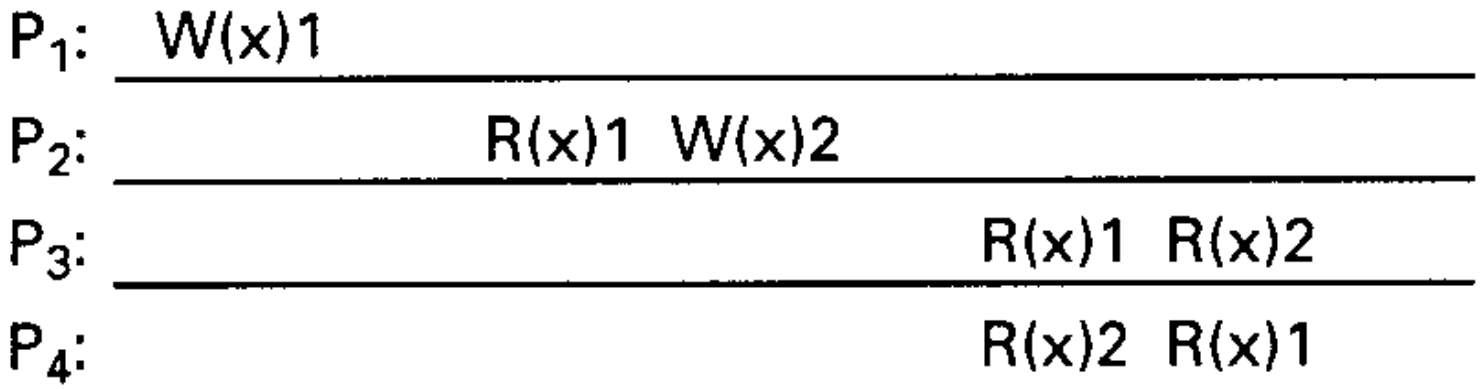


Fig. 6-18. A valid sequence of events for PRAM consistency.

Consistencia débil

- Según Dubois et al., 1986:
  1. Los accesos a las variables de sincronización son secuencialmente consistentes.
  2. No se permite realizar un acceso a una variable de sincronización hasta que las escrituras hayan terminado en todas partes.
  3. No se permite realizar un acceso a los datos (lectura o escritura) hasta realizar todos los accesos anteriores a las variables de sincronización.

Consistencia débil

- No es necesario mostrar inmediatamente a otros procesos cambios en la memoria hechos por cada operación de escritura. Los resultados de varias operaciones de escritura pueden ser combinados y enviados a otros procesos sólo cuando ellos lo necesitan.
- Implementación.
  - Se usa una variable de sincronización.

Consistencia débil

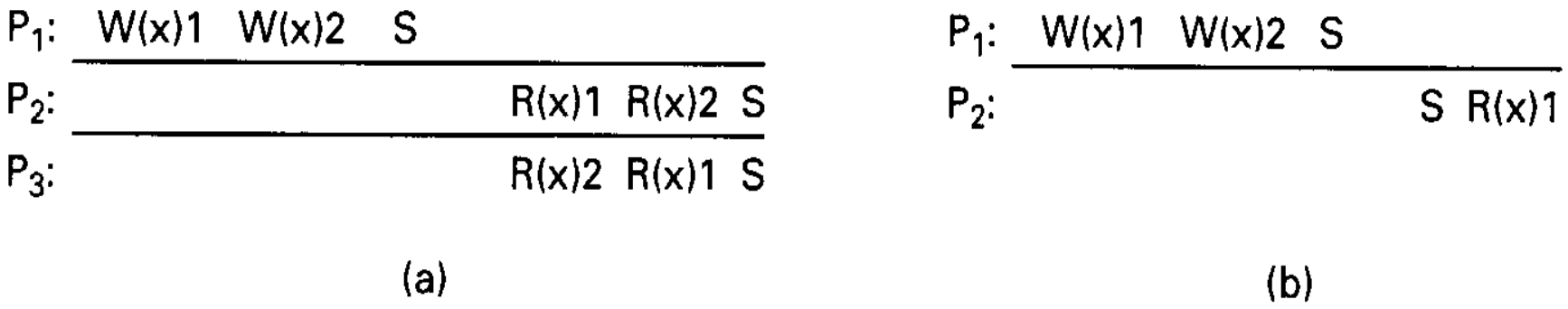


Fig. 6-22. (a) A valid sequence of events for weak consistency. (b) An invalid sequence for weak consistency.

Consistencia de liberación

- Gharachorlo et al., 1990.
- Distingue dos tipos de acciones de sincronización:
  - Las que preceden a la entrada en secciones críticas.
  - Las que se desarrollan después de completarse una sección crítica.
  - Estas acciones de sincronización son conocidas comúnmente como:
    - acquire access (acceso de adquisición).
    - release access (acceso de liberación).

Consistencia de liberación

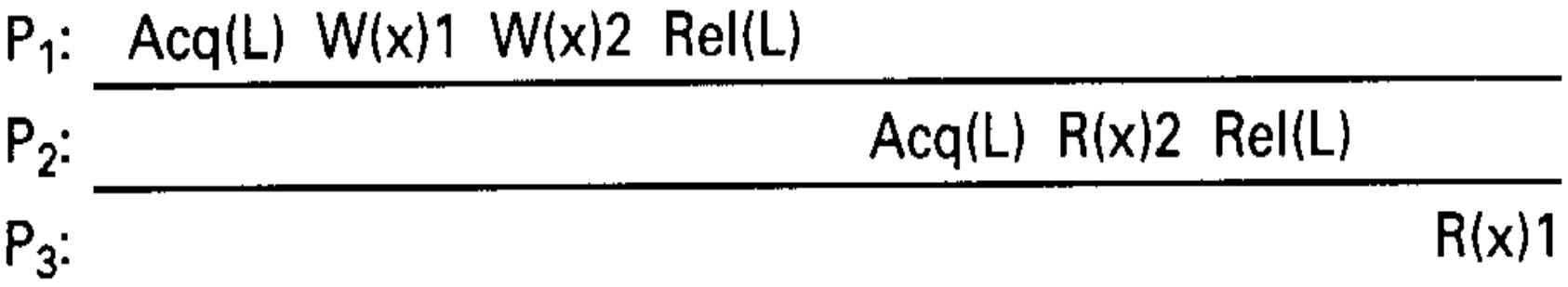


Fig. 6-23. A valid event sequence for release consistency.

Consistencia de entrada

- Mejora de la consistencia de liberación.
  - Ideas.
- Las variables compartidas que han cambiado ya no se determinan de manera empírica.
- Variables de sincronización independientes.

Consistencia de entrada

- Bershad y Zekauskas, 1991.
  1. No se permite realizar un acceso de adquisición a una variable de sincronización con respecto a un proceso hasta que se realicen todas las actualizaciones de los datos compartidos protegidos con respecto a ese proceso.
  2. Antes de permitir la realización de un acceso en modo exclusivo a una variables de sincronización por un proceso, ningún otro proceso debe poseer la variable de sincronización, ni siquiera en modo no exclusivo.

Consistencia de entrada

- Bershad y Zekauskas, 1991.
  3. Después de realizar un acceso en modo exclusivo a una variable de sincronización, no se puede realizar el siguiente acceso en modo no exclusivo de otro proceso a esa variable de sincronización hasta haber sido realizado con respecto del propietario de esa variable.

Consistency	Description
Strict	Absolute time ordering of all shared accesses matters
Sequential	All processes see all shared accesses in the same order
Causal	All processes see all casually-related shared accesses in the same order
Processor	PRAM consistency + memory coherence
PRAM	All processes see writes from each processor in the order they were issued. Writes from different processors may not always be seen in the same order

(a)

Weak	Shared data can only be counted on to be consistent after a synchronization is done
Release	Shared data are made consistent when a critical region is exited
Entry	Shared data pertaining to a critical region are made consistent when a critical region is entered

(b)

Fig. 6-24. (a) Consistency models not using synchronization operations. (b) Models with synchronization operations.

Memoria compartida distribuida

- DSM (Distributed Shared Memory).
  - DSM basada en páginas.
  - DSM basada en variables compartidas.
  - DSM basada en objetos.

DSM basada en páginas

- Memoria distribuida compartida clásica.
  - Li y Hudack, 1989.
    - IVY.

DSM basada en páginas

- Diseño básico.
- Replica.
- Granularidad.
- Obtención de la consistencia secuencial.
- Búsqueda de propietario.
- Búsqueda de las copias.
- Reemplazo de páginas.

Diseño básico

- Idea.
  - Intentar emular el caché de un multiprocesador mediante MMU y el software del sistema operativo.
- ¿Cómo se vería esta idea?
  - Ideas.
- Manejo de accesos locales vs. accesos remotos.

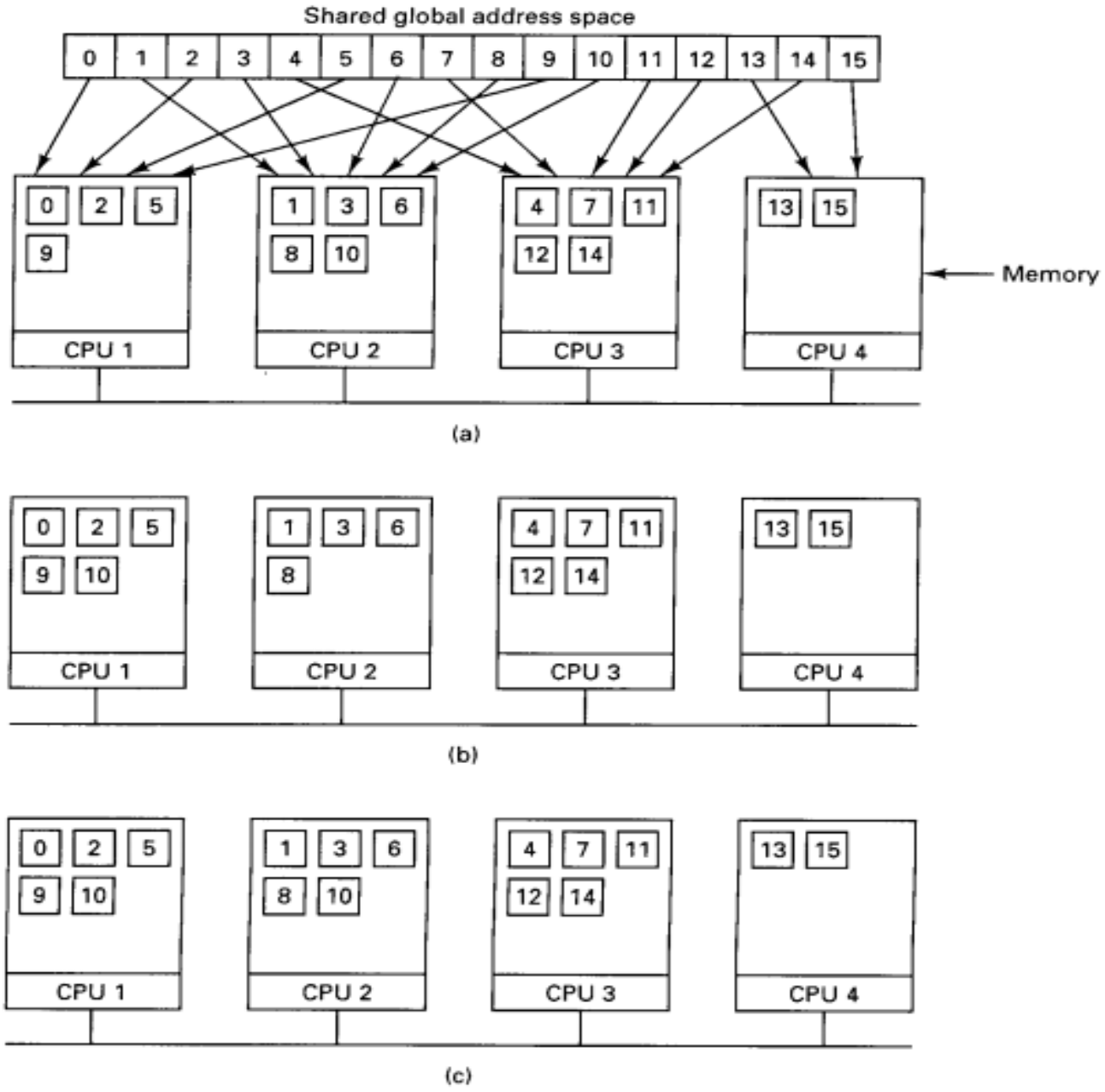


Fig. 6-25. (a) Chunks of address space distributed among four machines. (b) Situation after CPU 1 references chunk 10. (c) Situation if chunk 10 is read only and replication is used.

Réplica

- Incrementa el rendimiento.
- Réplica de pedazos de solo lectura.
- Réplica de pedazos de lectura-escritura.
- Inconsistencia.

Granularidad

- Tamaño del pedazo de memoria que se replica.
- Fallos de página.
- Traer página completa vs. Traer varias páginas.
- Compartición falsa.
- Compiladores inteligentes.

Granularidad

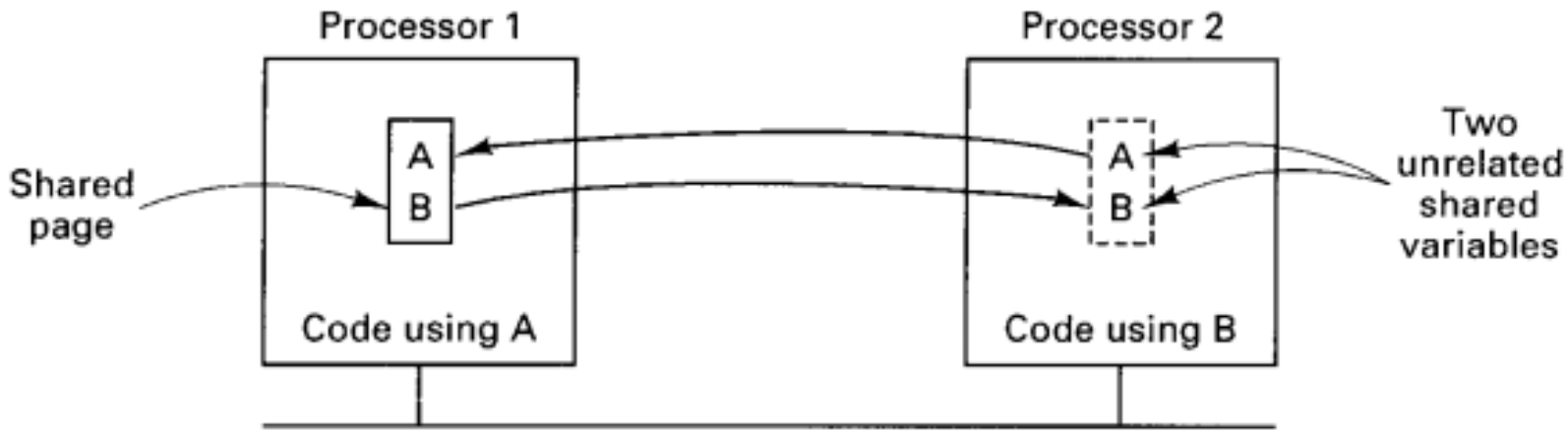
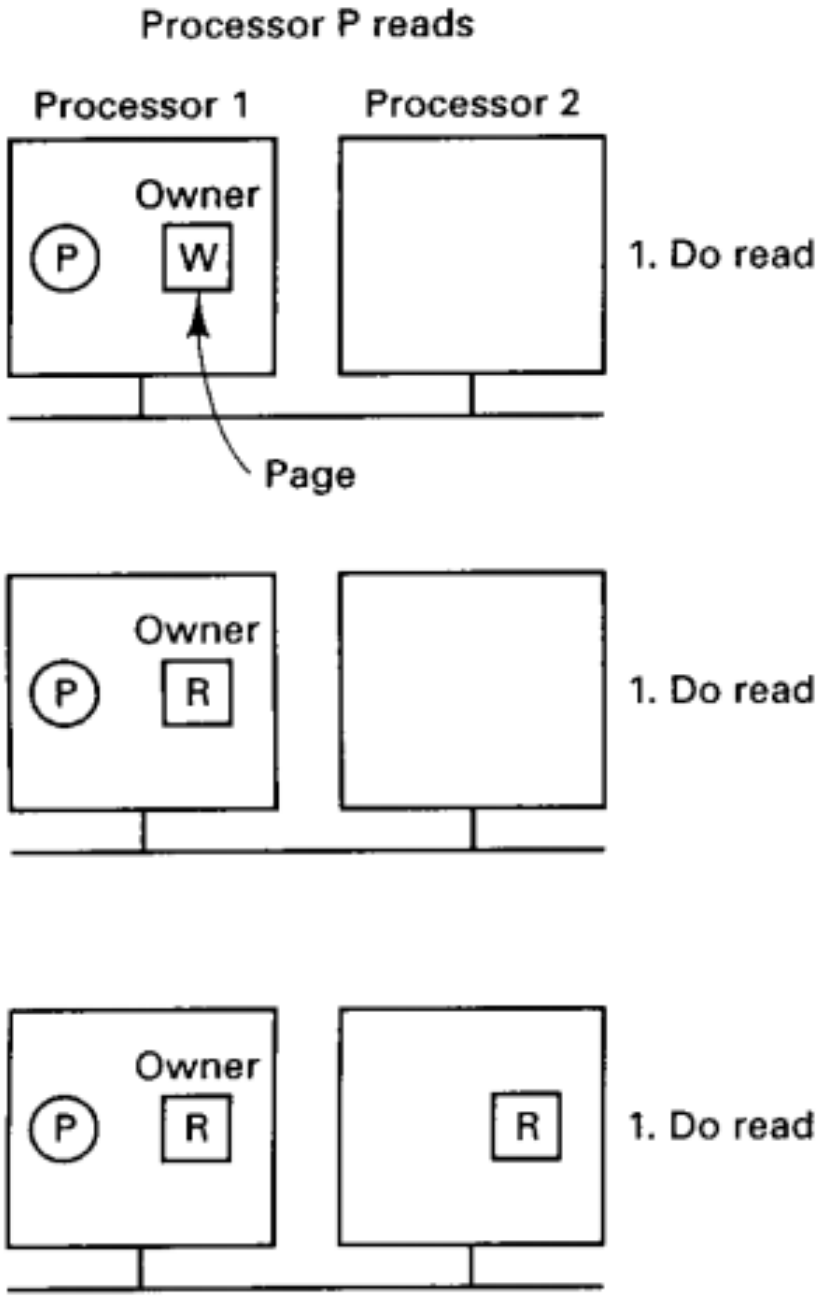
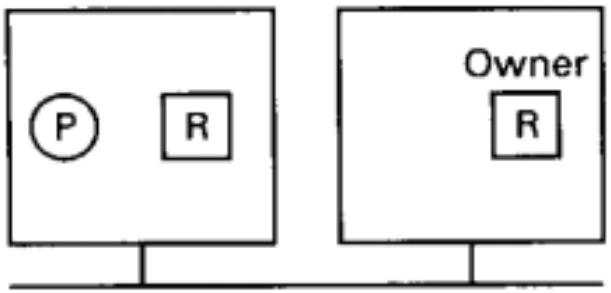


Fig. 6-26. False sharing of a page containing two unrelated variables.

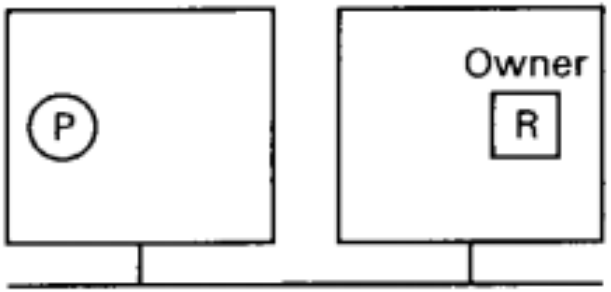
Obtención de la consistencia secuencial

- Réplicas de páginas de lectura-escritura.
- Averiguar palabra a escribir y su valor.
- Actualizaciones simultaneas.
- Esquema de invalidación vs. actualización.
- Protocolo de invalidación.
  - Se garantiza consistencia.

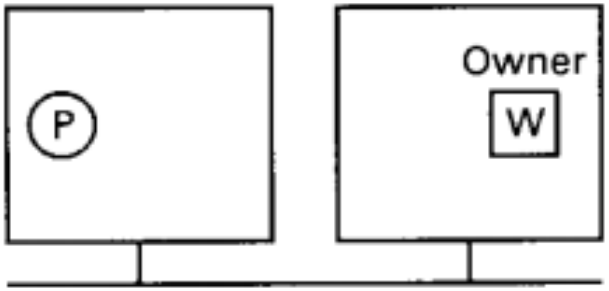

Processor P reads
Processor 1
Processor 2
Owner
P
W
1. Do read
Page
Owner
P
R
1. Do read
Owner
P
R
R
1. Do read



1. Do read

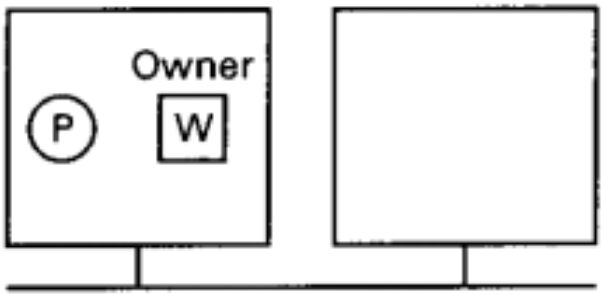


1. Ask for copy
2. Mark page as R
3. Do read

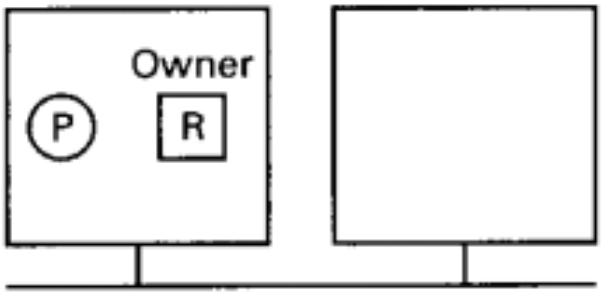


1. Ask for degradation
2. Ask for copy
3. Mark page as R
4. Do read

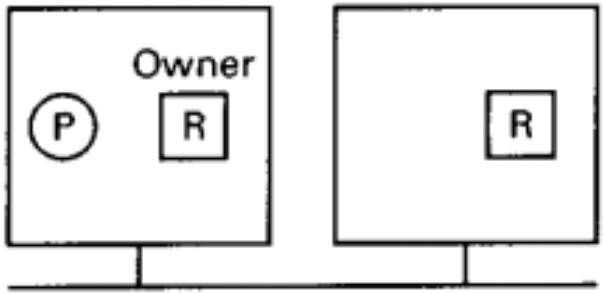
Processor P writes



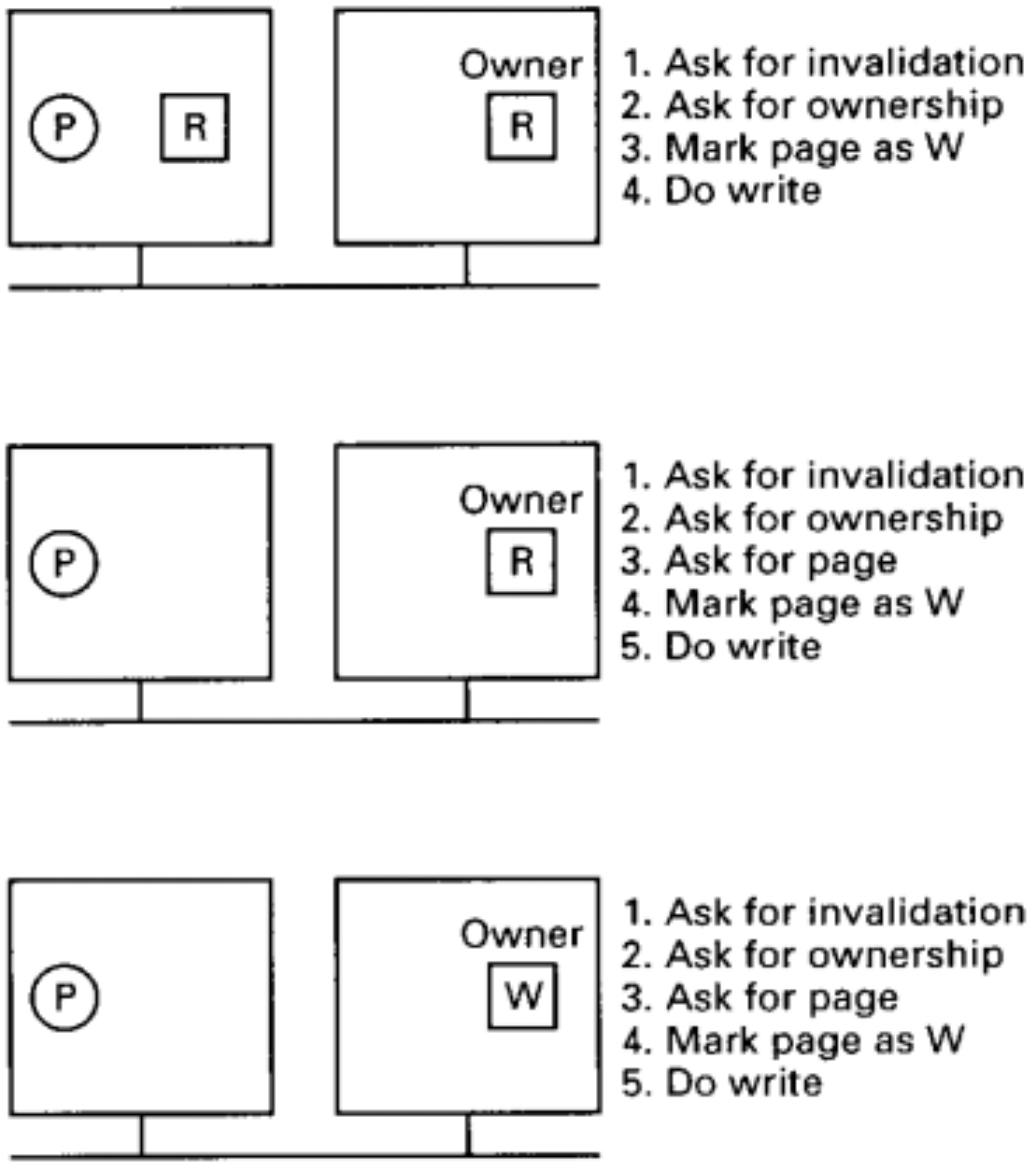
1. Do write



1. Mark page as W
2. Do write



1. Invalidate copies
2. Mark page as W
3. Do write


P
R
Owner
R
1. Ask for invalidation
2. Ask for ownership
3. Mark page as W
4. Do write
P
Owner
R
1. Ask for invalidation
2. Ask for ownership
3. Ask for page
4. Mark page as W
5. Do write
P
Owner
W
1. Ask for invalidation
2. Ask for ownership
3. Ask for page
4. Mark page as W
5. Do write

Búsqueda del propietario

- Buscar directamente al propietario.
- Usar controlador de páginas.
- Múltiples controladores de páginas.
- Registro de probables propietarios.
- ¿Ideas del funcionamiento de cada uno?

Búsqueda del propietario

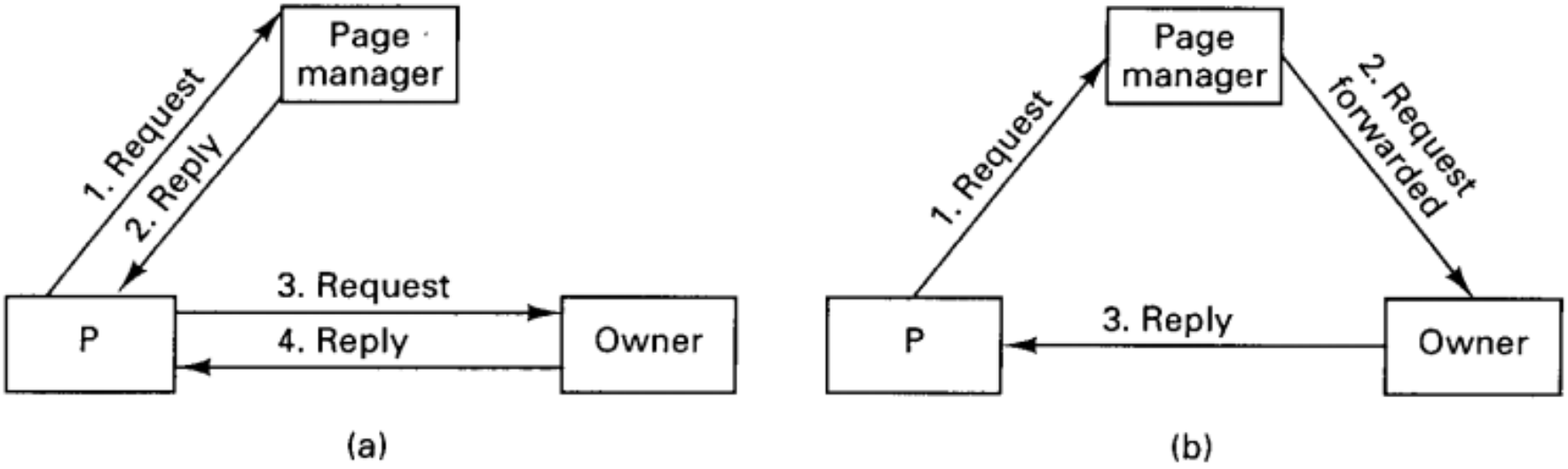


Fig. 6-28. Ownership location using a central manager. (a) Four-message protocol. (b) Three-message protocol.

Búsqueda de copias

- Ideas.
- Medio de transmisión no-confiable.
- Lista del conjunto de copias.
- Protocolo de invalidación.

Búsqueda de copias

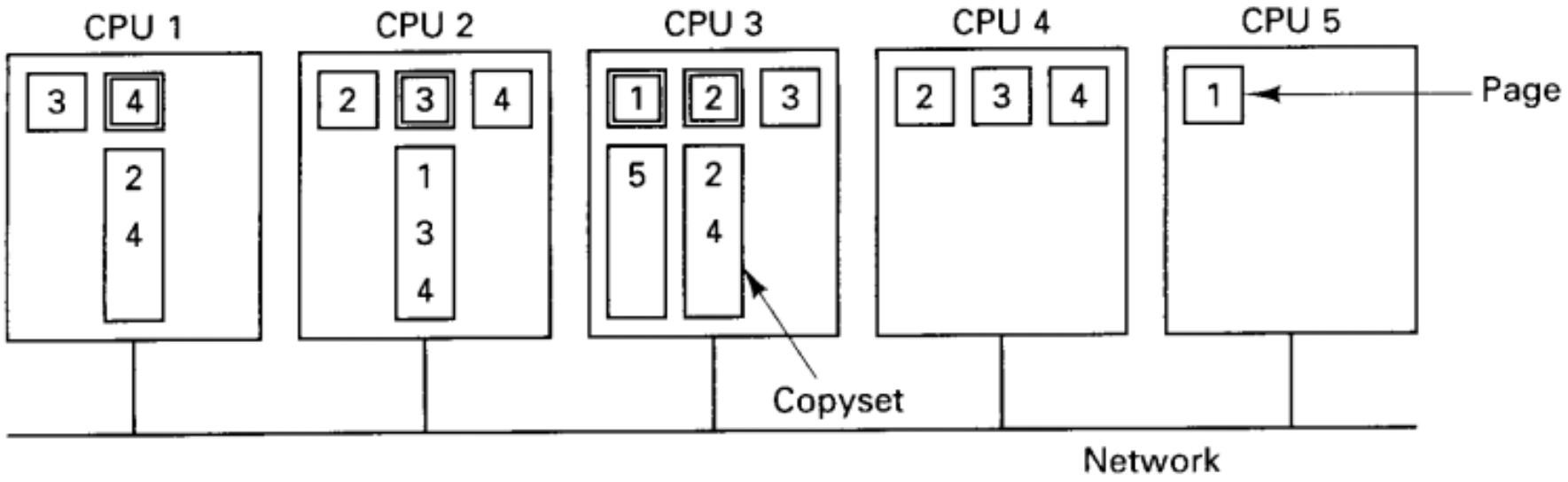


Fig. 6-29. The owner of each page maintains a copyset telling which other CPUs are sharing that page. Page ownership is indicated by the double boxes.

Reemplazo de páginas

- Buscar página para sacar de memoria.
- Página poseída por otro proceso.
- Página duplicada del proceso saliente.
- Página no duplicada.
- Transmitir número de marcos libres.
- Problema de compartición activa.
  - ΔT.

Sistemas distribuido de archivos

- Ideas.
- Hay que diferenciar entre:
  - Servicio de archivos.
    - Especificaciones.
    - Primitivas, parámetros y acciones.
  - Servidor de archivos.
    - Proceso que se ejecuta en alguna máquina.
    - Ayuda a implantar el servicio de archivo.

Sistemas distribuido de archivos

- Dos componentes básicos:
  - Servicio de archivos.
    - Operaciones en archivos individuales:
      - Lectura, escritura, adicción.
  - Servicio de directorios.
    - Crear y administrar directorios.
    - Añadir y eliminar archivos del directorio.

Interfaz del servicio de archivos

- Pregunta fundamental.
  - ¿Qué es una archivo?
    - Características.
    - Atributos.
      - Propietario.
      - Tamaño.
      - Permisos de acceso.
      - Fecha de creación.

Interfaz del servicio de archivos

- El servicio de archivo puede dividirse en dos tipos:
  - Modelo carga/descarga.
  - Modelo de acceso remoto.

Interfaz del servicio de archivos

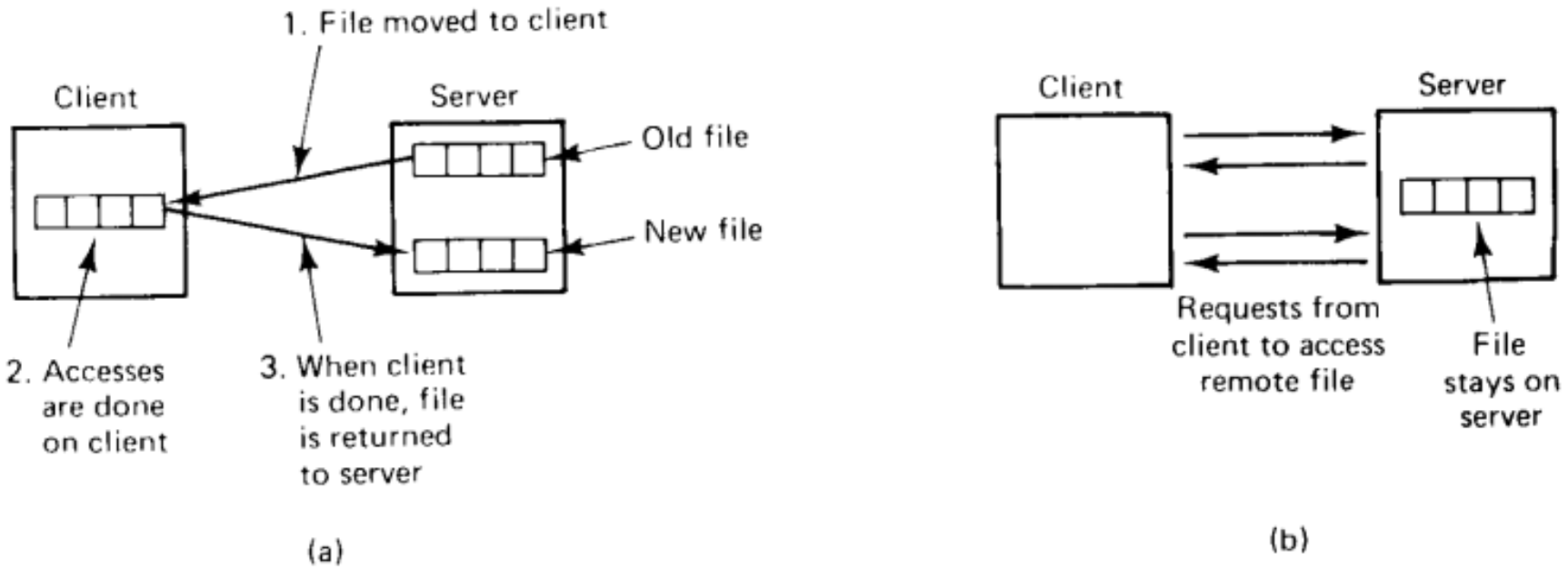


Fig. 5-1. (a) The upload/download model. (b) The remote access model.

Interfaz del servidor de directorios

- Define un alfabeto y una sintaxis para formar los nombres de:
  - Archivos.
  - Directorios.

Interfaz del servidor de directorios

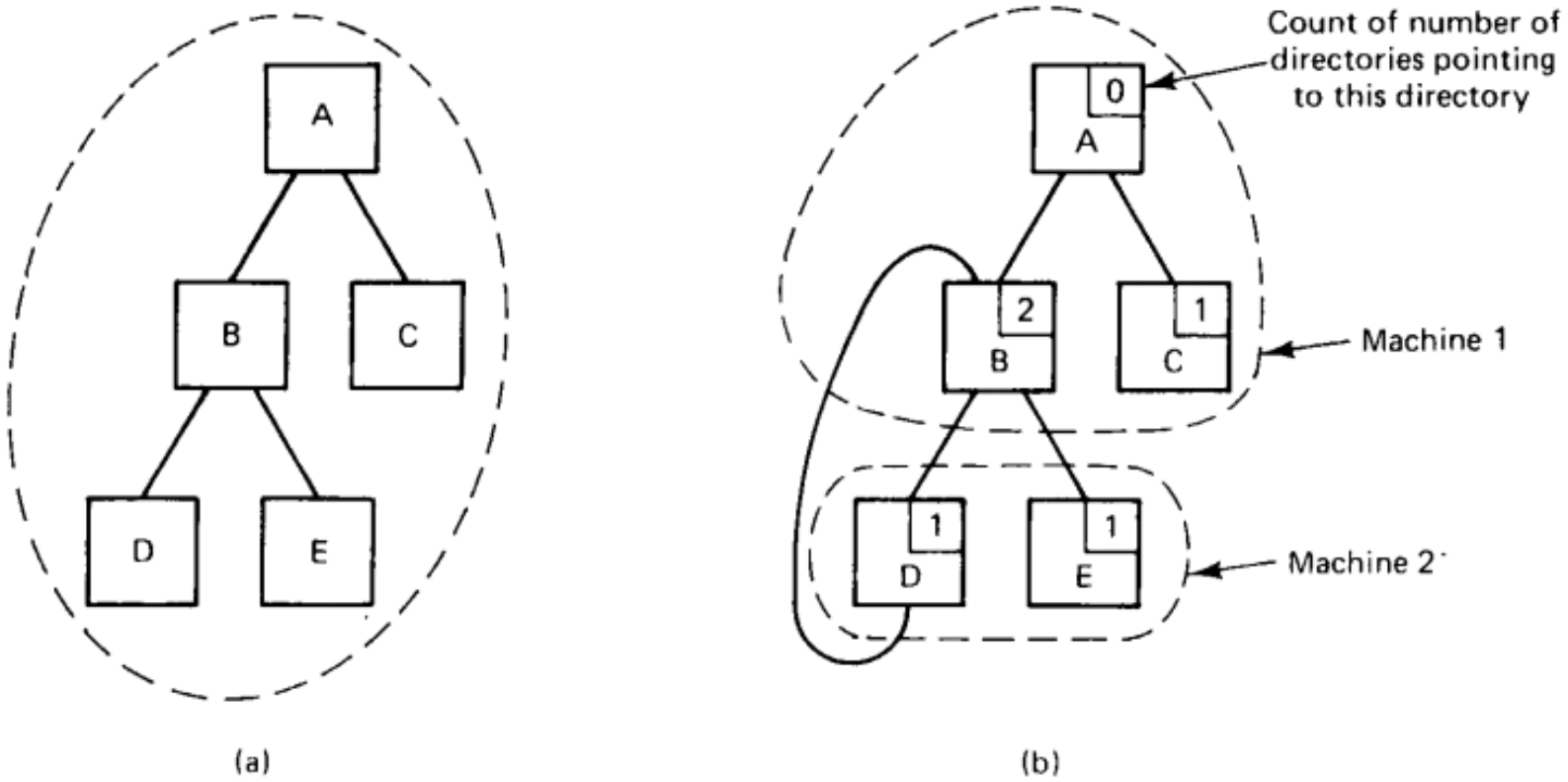


Fig. 5-2. (a) A directory tree contained on one machine. (b) A directory graph on two machines.

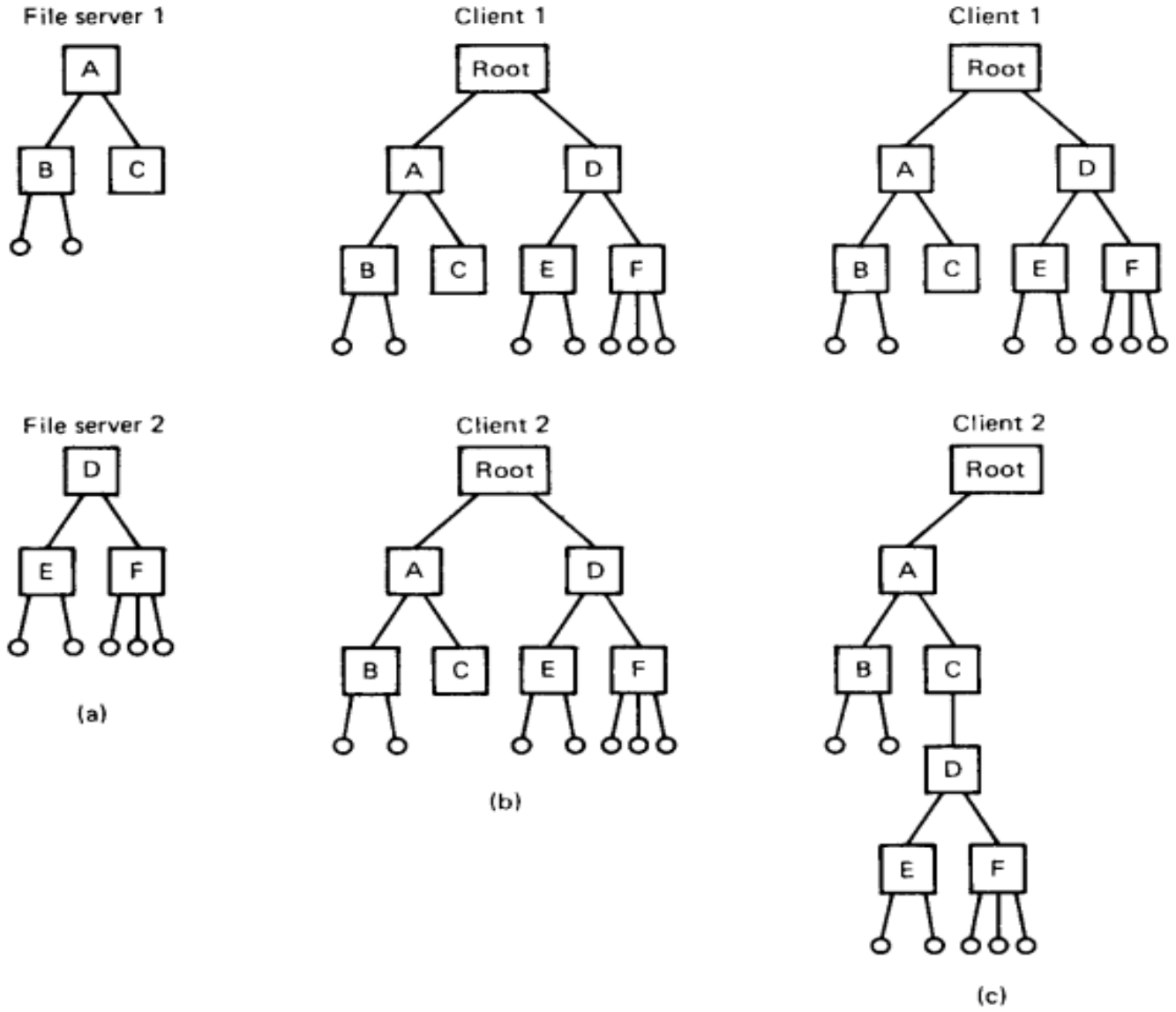


Fig. 5-3. (a) Two file servers. The squares are directories and the circles are files. (b) A system in which all clients have the same view of the file system. (c) A system in which different clients may have different views of the file system.

Transparencia de nombres

- Dos tipos de transparencia:
  - Transparencia con respecto a la posición.
    - /servidor1/dir1/dir2/x
  - Independencia con respecto a la posición.
    - /servidor1/dir1/dir2/x a /servidor2/dir1/dir2/x

Transparencia de nombres

- Tres métodos usuales para nombrar los archivos y directorios en un sistema distribuido:
  - Nombre máquina + ruta de acceso.
    - /maquina/ruta o maquina:ruta
  - Montaje de sistemas de archivos remotos en la jerarquía local de archivos.
  - Un espacio de nombres que tenga la misma apariencia en todas las máquinas.

Réplicas

- Razones para la existencia de este servicio:
  1. Aumentar la confiabilidad al disponer de respaldos independientes.
  2. Permitir el acceso al archivo aunque falle el servidor de archivos.
  3. Repartir la carga de trabajo entre varios servidores.

Métodos de replicación

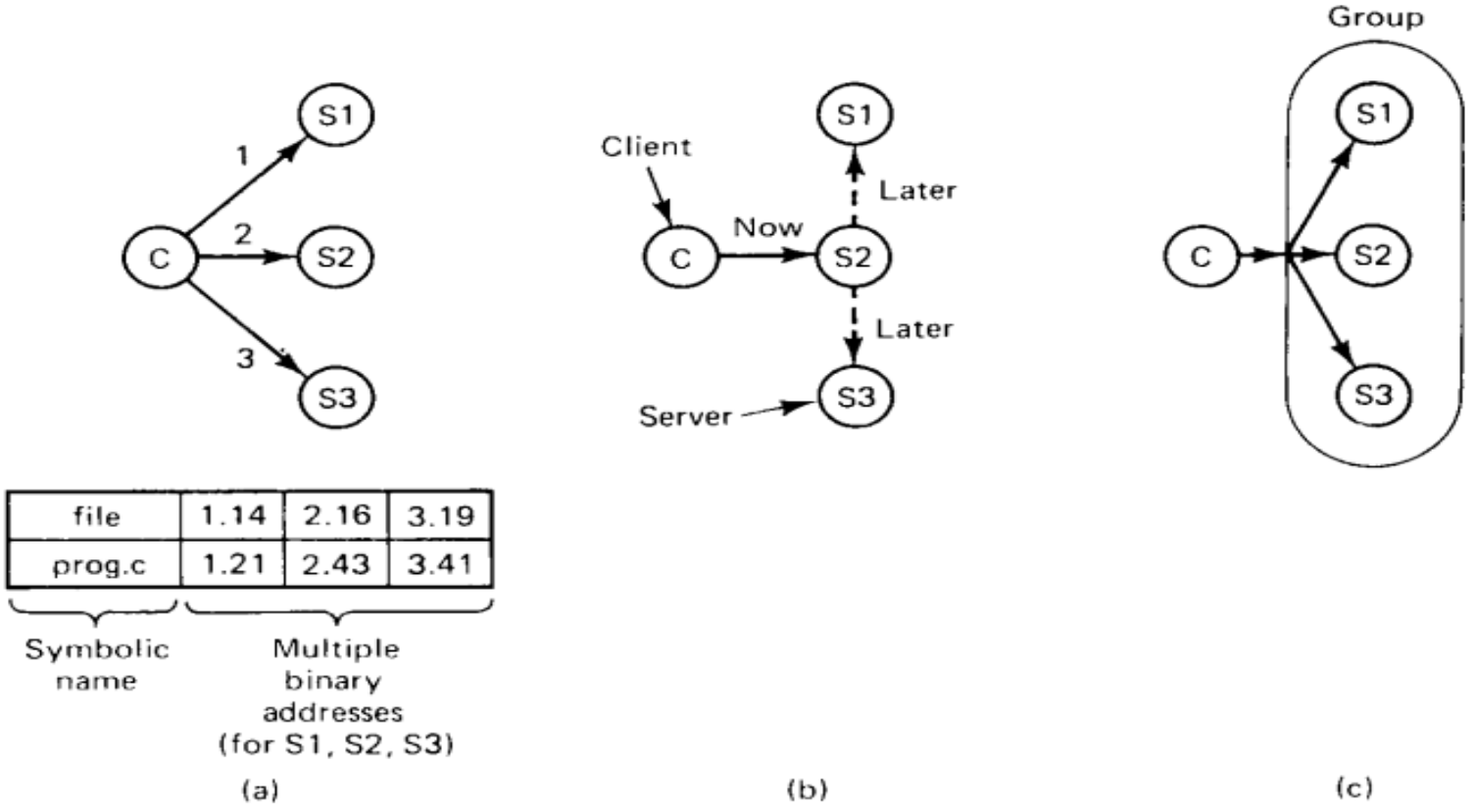


Fig. 5-12. (a) Explicit file replication. (b) Lazy file replication. (c) File replication using a group.

Semántica de los archivos compartidos

- Semántica (Consistencia).
  - Concepto.
  - Uso.
  - Problemas.

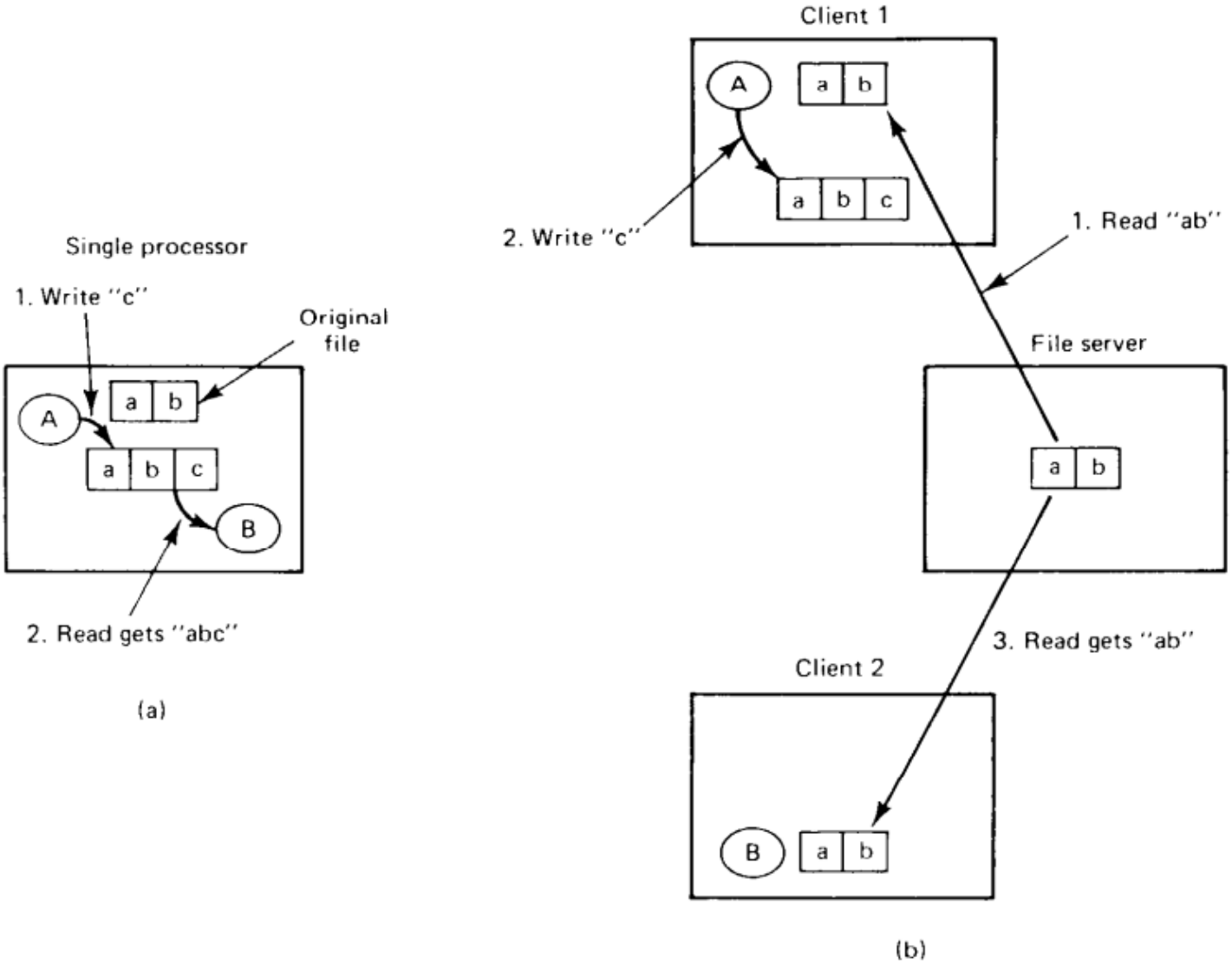


Fig. 5-4. (a) On a single processor, when a READ follows a WRITE, the value returned by the READ is the value just written. (b) In a distributed system with caching, obsolete values may be returned.

Semántica de los archivos compartidos

Método	Comentario
Semántica de Unix	Cada operación en un archivo es visible a todos los procesos de manera instantánea
Semántica de Sesión	Ningún cambio es visible a otros procesos hasta que el archivo se cierre
Archivos Inmutables	No existen actualizaciones; es más fácil compartir y replicar
Transacciones	Todos los cambios tiene la propiedad del todo o nada

Protocolos de actualización

- Dos métodos para la actualización:
  - Réplica de la copia primaria.
  - Algoritmo del voto.
    - Gifford, 1979.
    - Algoritmo del voto con fantasma.
  - Ideas del funcionamiento de cada uno.

Réplica de la copia primaria

- Funcionamiento.
  - Servidor primario.
- ¿Qué pasa si el servidor primario falla?.
  - Soluciones.

Algoritmo del voto

- La idea fundamental es exigir a los clientes que soliciten y adquieran el permiso de varios servidores antes de leer o escribir un archivo replicado.

Algoritmo del voto

- El esquema de Gifford.
  - Dado un archivo con N réplicas.
  - Un cliente necesita:
    - Quórum de lectura → N_r
    - Quórum de escritura → N_w
  - Los valores de $N_r + N_w > N$

Protocolos de actualización

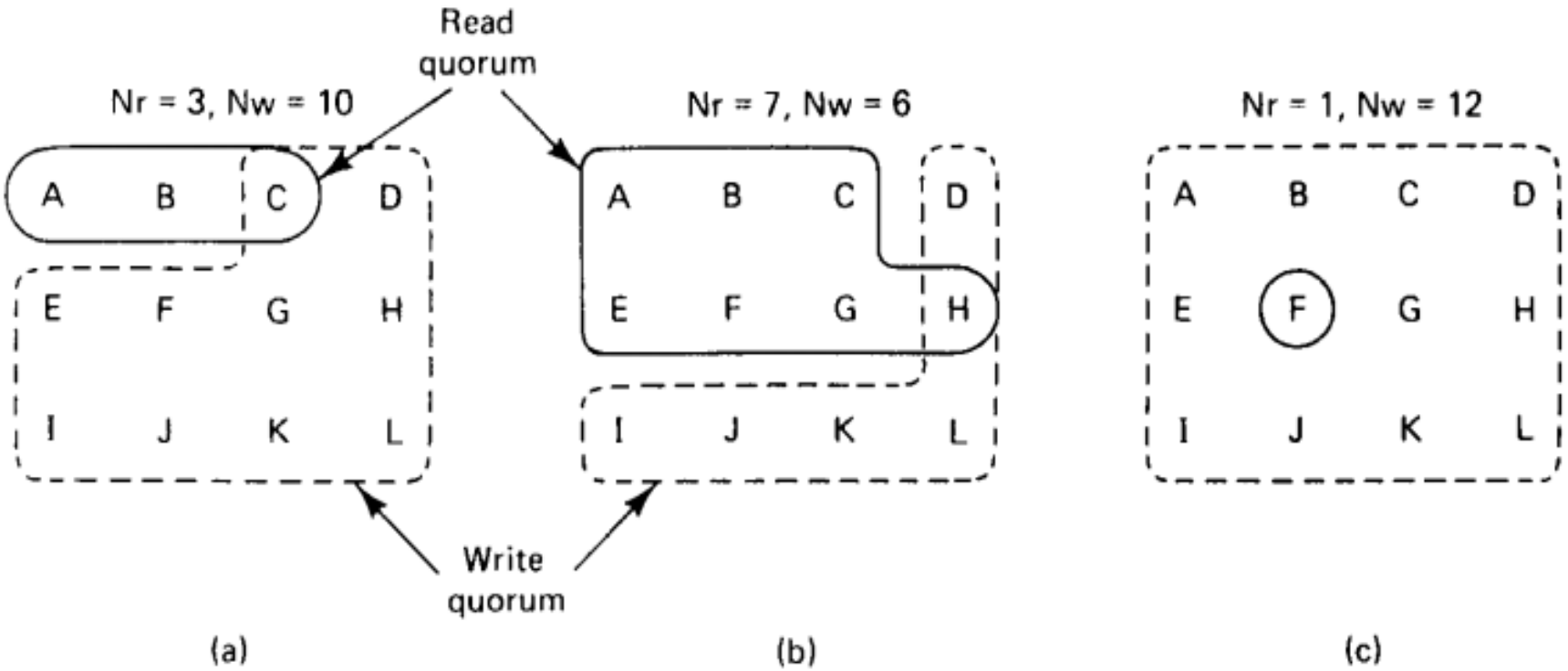


Fig. 5-13. Three examples of the voting algorithm.

Algoritmo del voto con fantasma

- Variante del algoritmo con voto.
  - Ideas.
- En la mayoría de las aplicaciones.
  - Lecturas son más frecuentes que las escrituras.
- ¿Qué ocurre sin fallan unos cuantos servidores?

Algoritmo del voto con fantasma

- Se crea un servidor fantasma.
  - ¿Qué hace?
  - Por cada servidor real fallido.
  - No se permite en el quórum de lectura, pero si en el de escritura.
    - ¿Por qué?
  - Funcionamiento.
    - ¿Qué hace con las escrituras?

Algoritmo del voto con fantasma

- Al arrancar de nuevo un servidor fallido.
  - Se debe obtener un quórum de lectura.
    - ¿Por qué?
- ¿Por qué funciona el algoritmo?
  - Posee la misma propiedad del esquema básico de votación.
    - N_r y N_w se eligen de modo que sea imposible obtener un quórum de lectura y otro de escritura al mismo tiempo.