

Tema IV

PROCESAMIENTO PARALELO

4.2 Introducción

Aunque no existe un consenso sobre la definición de un multiprocesador, las siguientes propiedades corresponden a una de las primeras caracterizaciones de sistemas multiprocesador, que fue realizada por P.H. Enslow en 1977:

- Debe estar compuesto por dos o más procesadores.
 - Los procesadores deben compartir el acceso a una memoria común.
 - Los procesadores deben compartir acceso a canales de E/S, unidades de control y dispositivos.
 - El sistema es controlado por un único sistema operativo.
-

4.3 Tipos de plataforma de computación paralela

- ▶ **Organización lógica**
 - Visión que tiene el programador
 - **Estructura de control**
 - Capacidad de expresar tareas paralelas
 - **Modelo de comunicación**
 - Método de comunicación entre las tareas
- ▶ **Organización Física**
 - Estructura del hardware
 - Modelos de espacio de direcciones común
 - Memoria compartida
 - Un único sistema de memoria física
 - Memoria distribuida
 - Cada procesador tiene su propia memoria

Organización lógica

Estructura de Control
Modelo de Comunicación

4.3.1 Organización basada en la Estructura de Control

► Criterios de clasificación

- propiedades del proceso (estructura, topología y ejecución).
- propiedades de interacción.
- propiedades de los datos (división y localización).

Según los autores, existen muchos paradigmas que se utilizan en las aplicaciones paralelas. Vamos a comentar dos:

- Maestro esclavo
- SPMD

- *Descomposición iterativa*: Algunas aplicaciones están basadas en la ejecución de un lazo donde cada iteración se puede realizar de forma independiente. Esta técnica se implementa a través de una cola central de tareas ejecutables, y se corresponde con el paradigma de descomposición en grupos de tareas.
- *Paralelismo algorítmico*: el cual se centra en paralelizar el flujo de los datos de entrada.
 - *Descomposición geométrica*: El dominio del problema se divide en pequeños subdominios y cada procesador ejecuta el algoritmo en la parte del subdominio que le corresponde.
 - *Descomposición especulativa*: Se intentan N técnicas de solución simultáneamente, y $(N-1)$ de ellas se eliminan tan pronto como una de ellas devuelve una respuesta correcta.
- *Descomposición funcional*: La aplicación se divide en distintas fases, y cada fase ejecuta una parte diferente del algoritmo para resolver el problema. También puede denominarse *segmentación de datos*.
- *Maestro/Esclavo*: El proceso maestro es el responsable de descomponer el problema entre sus procesos esclavos y de, posteriormente, recoger los resultados que le envían los esclavos para ordenarlos y obtener el resultado final.
- *SPMD (Single Program Multiple Data)*: En este paradigma cada procesador ejecuta el mismo código pero sobre distintas partes de los datos.
- *Descomposición recursiva o divide y vencerás*: El problema se divide en subproblemas que se resuelven de forma independiente para, posteriormente, combinar sus resultados parciales y obtener el resultado final.

4.3.1.1 Paradigma Maestro Esclavo

- ▶ El maestro es el responsable de descomponer el proceso en pequeñas tareas, distribuirlas, recoger los resultados y ordenarlos
- ▶ Si hay un gran número de tareas puede ser un cuello de botella
 - Mas de un maestro
- ▶ Balances de carga
 - Estático
 - Se realiza al comienzo de la computación
 - Dinámico
 - Cuando el número de tareas es mayor que el de procesadores
 - Cuando el número de tareas es desconocido

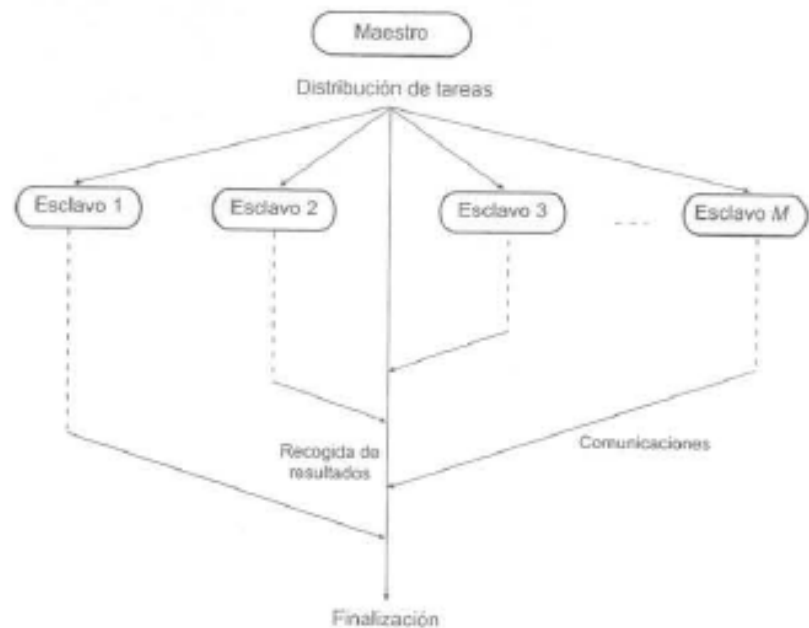


Figura 4.1: Estructura Maestro/Escravo estática.

4.3.1.2 Paradigma SPMD (*Single Program Multiple Data*)

- ▶ Cada procesador ejecuta el mismo código pero sobre distintas partes de los datos
 - La comunicación es entre esclavos
 - Eficiente si los datos están bien distribuidos y el sistema es homogéneo
 - Muy sensible a la pérdida de un procesador



Figura 4.2: Estructura básica de un programa SPMD.

4.3.2 Organización basada en el Modelo de Comunicación

- ▶ Memoria compartida, el espacio de direcciones es único
- ▶ El paso de mensaje

4.3.2.1 Espacio de Direcciones Compartidos

- ▶ Un valor escrito por un procesador, puede ser leído por cualquier otro
- ▶ La memoria es accesible por todos los procesadores a través de la “red de interconexión”
- ▶ Parámetros de la eficiencia de la red de interconexión
 - Latencia de la red
 - Tiempo que tarda en enviar un mensaje a través de la red de interconexión
 - Ancho de banda
 - Número de bits que puede enviar por unidad de tiempo
- ▶ Solución válida para un número de procesadores pequeño
 - Compartir los medios de interconexión son cuellos de botellas

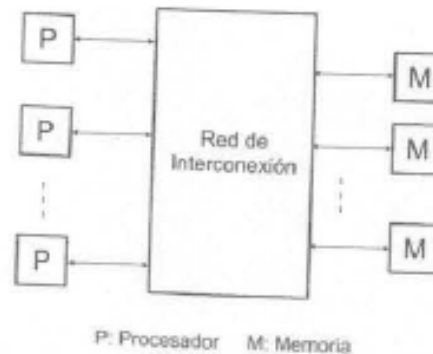


Figura 4.3: Esquema de la arquitectura de memoria compartida.

UMA – Uniform Memory Access

- ▶ El tiempo de acceso es el mismo para cualquier palabra
- ▶ Incorporan memoria cache local y global
 - Problemas de coherencia de caches
- ▶ Bus común
 - Problema de escalabilidad (máximo entre 16 y 32 procesadores)
 - **Escalabilidad:** capacidad del sistema para mejorar la potencia de cálculo cuando el número de componentes del sistema aumentan

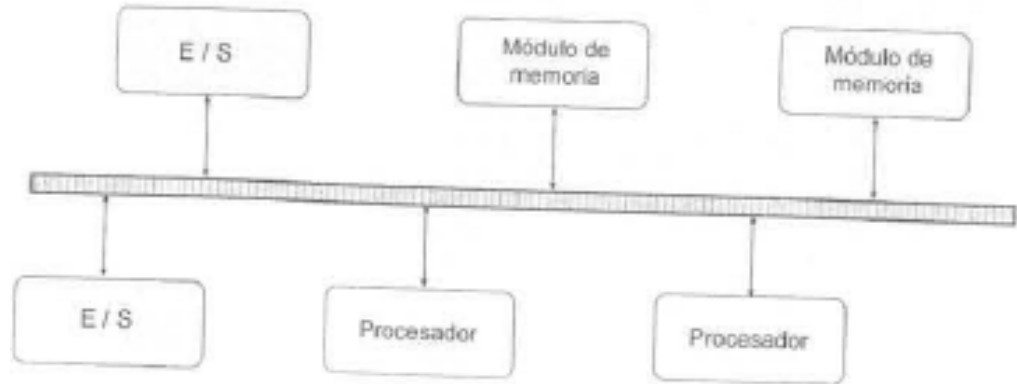


Figura 4.4: Organización multiprocesador de bus común.

NUMA (*Non Uniform Memory Access*) – DSM (*Distribute Shared Memory*)

- ▶ Mejora: dotar al procesador de memoria local, donde almacenar código y datos no compartidos
- ▶ Cada procesador con una memoria local
 - Con código y datos que no tengan que ser compartidos con otros procesadores
 - Evitan el acceso a memoria a través de la red de interconexión
- ▶ Incluye mecanismos hardware dedicado a la coherencia de caches
 - ccNUMA (caché coherent NUMA)
 - COMA (Cahé-Only Memory Access)

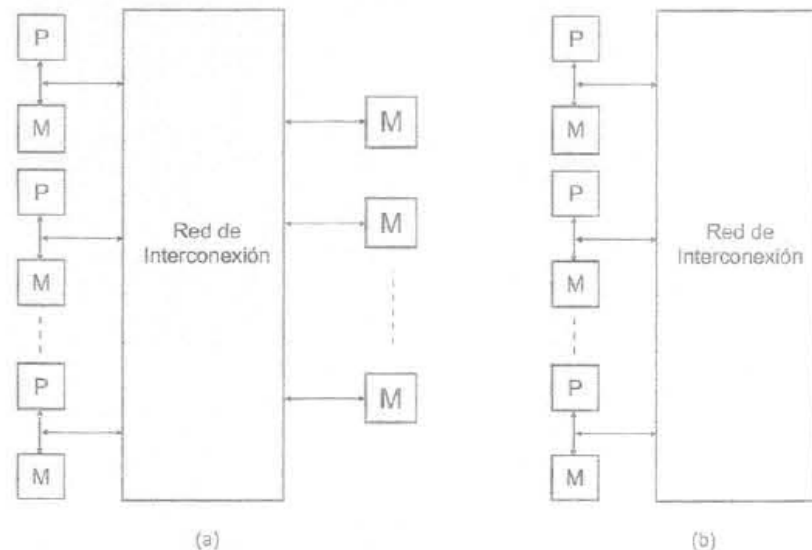
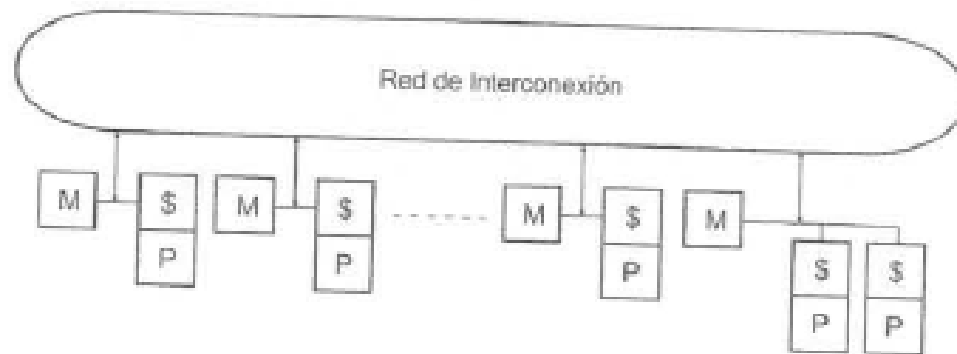


Figura 4.5: a) Arquitectura de memoria compartida con acceso no uniforme a memoria, b) memoria compartida con acceso no uniforme a memoria solo con memoria local.

ccNUMA (caché coherente NUMA)

- ▶ Cada nodo tiene una porción de la memoria total del sistema
- ▶ Las variables compartidas se reparten de manera de que solo existe una copia de cada variable
- ▶ Cada nodo consta de uno o varios procesadores con sus caches y su memoria principal
- ▶ La coherencia de caché se mantiene por
 - Escaneo de operaciones (protocolo snoopy)
 - Registro de localización de variables



\$: Protocolo de Coherencia de Cache

Figura 4.6: Arquitectura ccNUMA.

COMA (Cahé-Only Memory Access)

- ▶ Los procesadores que componen cada nodo no incluyen memoria local, solo caché
- ▶ Si tiene que acceder a una posición de memoria que se encuentra en un nodo remoto, lo copia
 - Complejidad en mantener la coherencia de las variables

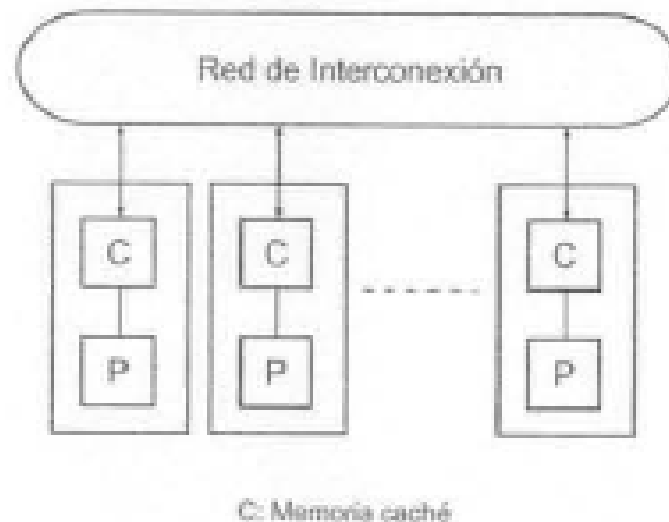


Figura 4.7: Arquitectura COMA.

4.3.2.2 Paso de Mensajes

- ▶ Intercambio de información, en forma de mensajes, entre los diferentes procesadores que forman el sistema
- ▶ Elementos necesarios
 - Emisor, receptor, canal de comunicación, y el mensaje
- ▶ Operaciones básicas necesarias

- Envío: Usada para enviar el mensaje.

- Recepción: Usada para recibir el mensaje.

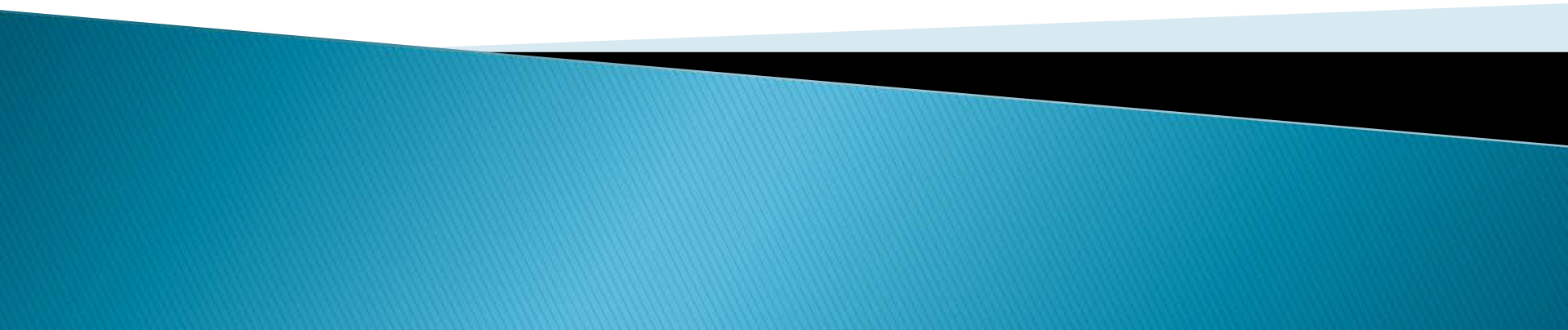
- Identificación: Usada para identificar cada uno de los procesadores del sistema, mediante un identificador único. Este identificador debe acompañar a las operaciones de envío y recepción para poder identificar al emisor y el receptor.

- Número de participantes: Usada para identificar el número de procesadores que participan en la comunicación.

Organización física

Memoria Compartida

Memoria Distribuida



4.4 Sistemas de Memoria Compartida

▶ Aspectos de diseño

- Organización de la memoria
 - Ya comentado: espacio único o compartido
- El diseño del protocolo de coherencia de la caché
- El diseño de la red de interconexión
 - Usada para:
 - Para acceder a la memoria remota
 - Intercambiar mensajes entre procesadores
- Topologías de redes de interconexión
 - Estáticas
 - Definidas durante la construcción de la máquina
 - Dinámica
 - Que puede adaptarse a los requisitos de comunicación de los programas que se ejecuten

4.4.1.1 Redes Estáticas

► Redes unidimensionales

- Conectar cada procesador con dos procesadores vecinos
- Lineal
 - Puede enviar un mensaje simultáneamente a su izquierda y a su derecha
- Anillo
 - Enlazando los extremos
 - Los mensajes se pueden enviar por la izquierda o derecha, según sea el camino mas óptimo

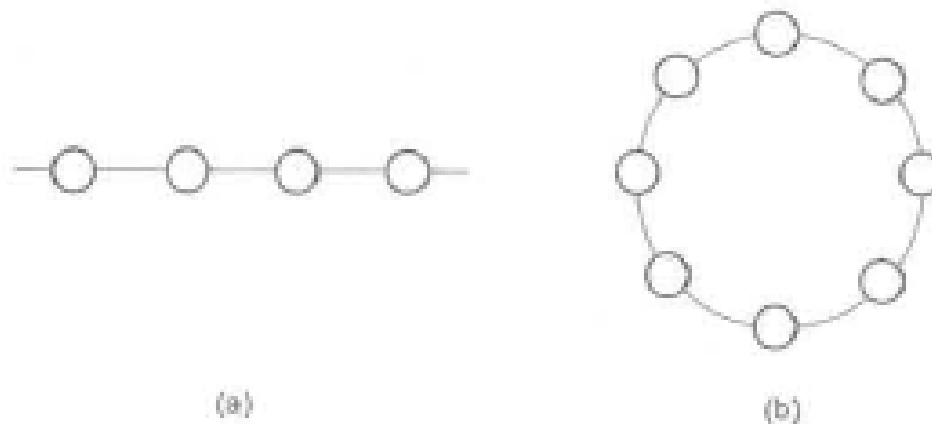


Figura 4.8: Topologías de red unidimensionales: a) lineal, b) anillo.

Redes bidimensionales

- ▶ Anillo cordal
 - Incrementando el número de enlace por nodo
- ▶ Malla
- ▶ Red sistólica o array sistólico
 - Red en malla con conexión en diagonal
- ▶ Red completamente conectada
- ▶ Red en estrella
 - Nodo central de comunicación

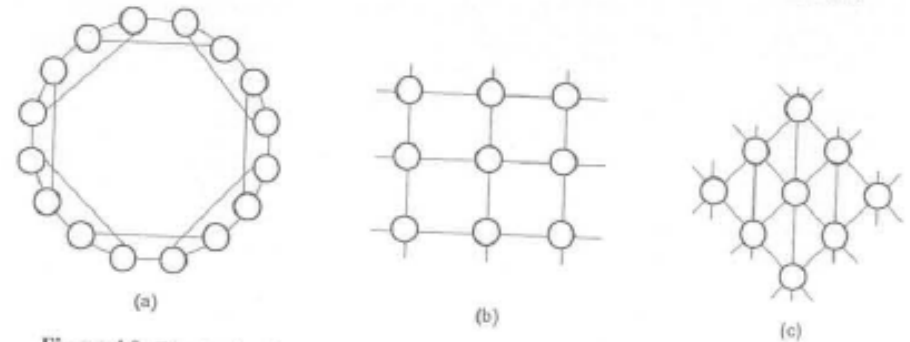


Figura 4.9: Topologías de red bidimensionales: a) anillo cordal, b) malla, c) red sistólica.

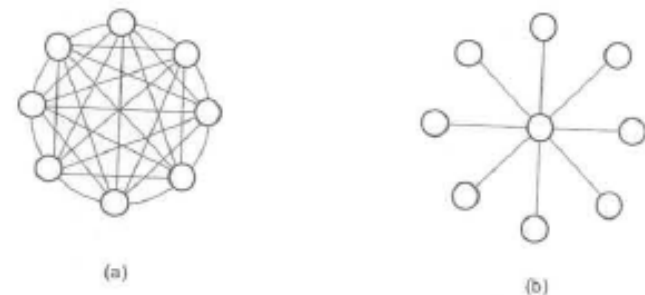


Figura 4.10: Topologías de red bidimensionales: a) red completamente conectada, b) red en estrella.

Redes en árbol

- ▶ Un procesador en cada nodo
- ▶ Solo un camino entre cualquier par de procesadores
- ▶ Las comunicaciones pueden verse comprometidas si el número de procesadores es alto y se realizan comunicaciones con procesadores situados en niveles superiores

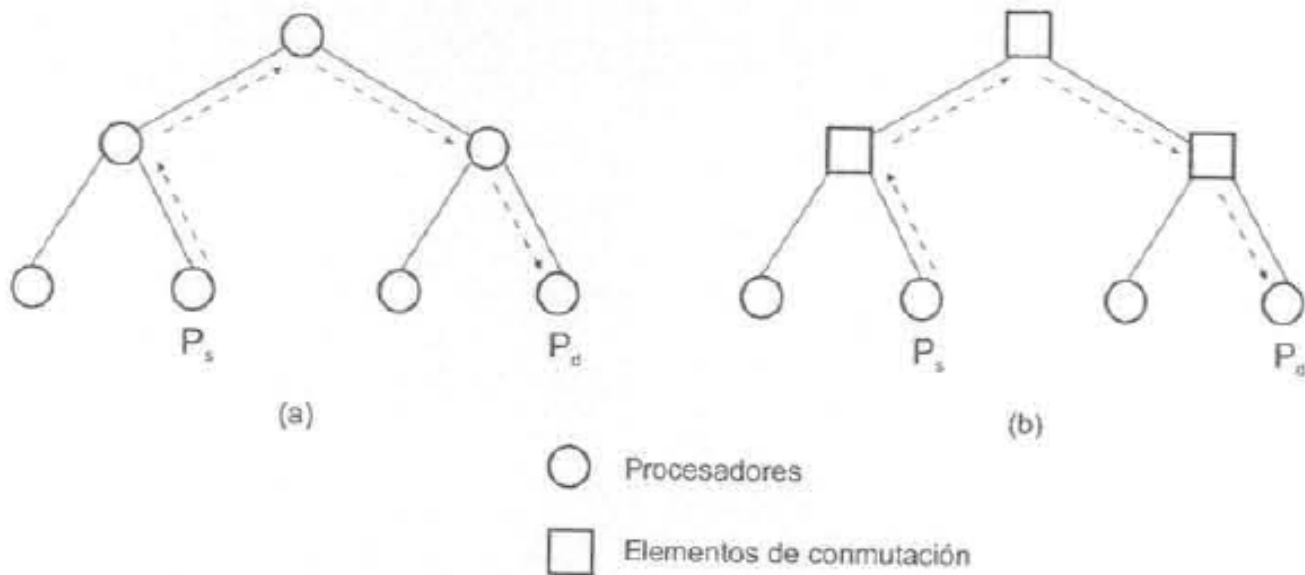


Figura 4.11: Topología de red bidimensional: red de árboles binarios.

Red en árbol grueso

- Consiste en aumentar el número de conexiones con los procesadores de menor nivel

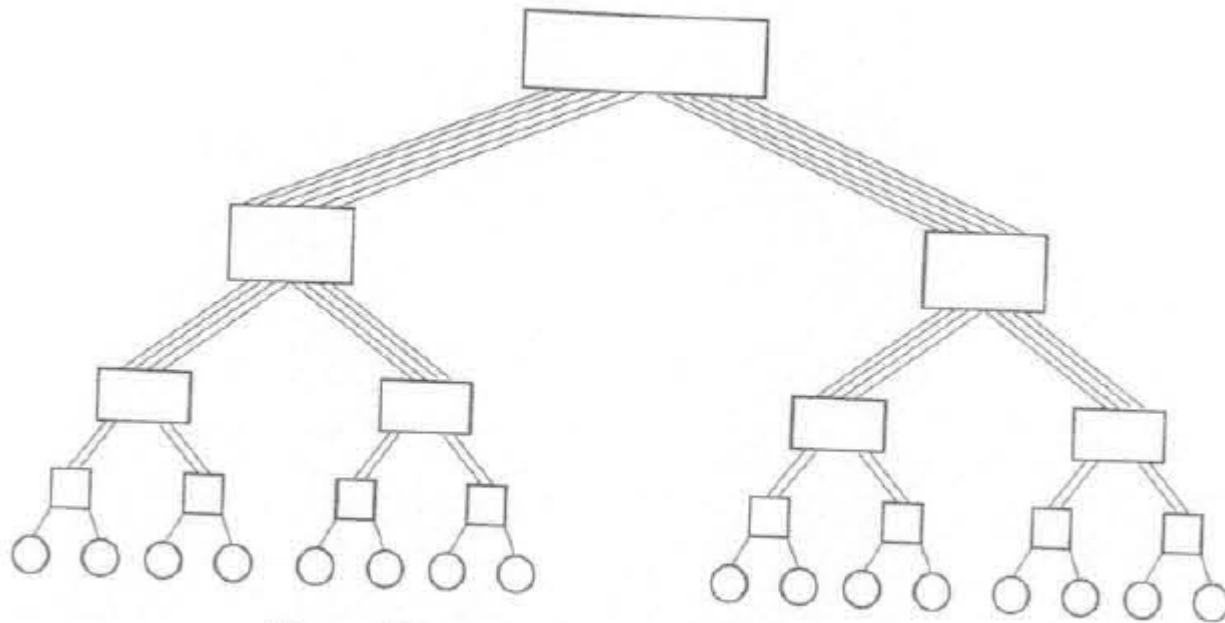


Figura 4.12: Topología de red bidimensional: fat tree.

Redes tipo Mesh

- ▶ Cada procesador se conecta con cuatro procesadores salvo los extremos
 - Mesh cuadrado
 - Mesh rectangular
 - Mesh cerrada o toro
 - Mesh tridimensional

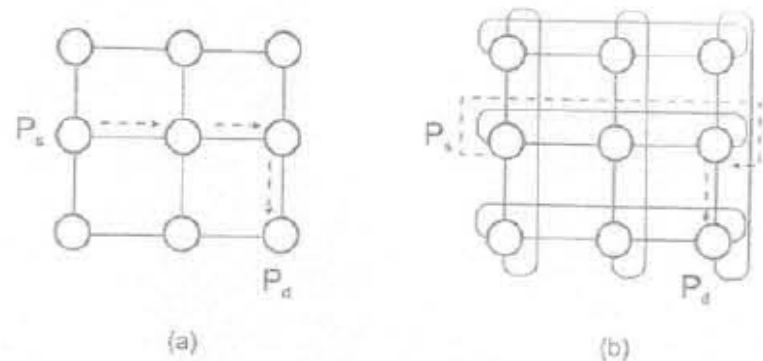


Figura 4.13: Topologías de red bidimensionales: a) mesh cuadrada, b) mesh cerrada o toro.

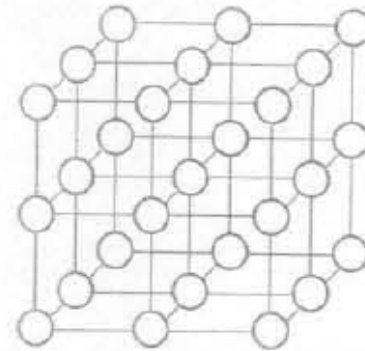


Figura 4.14: Topología de red tridimensional: mesh tridimensional.

Redes Hipercubos

- ▶ Redes multidimensionales con dos procesadores por dimensión
- ▶ Un hipercubo de dimensión d tiene $P=2^d$ procesadores
- ▶ Dos procesadores se conectan entre sí, si tienen un bit distinto en un aposición determinada
- ▶ Un procesador de un hipercubo de dimensión d , se conecta directamente con d procesadores
- ▶ Todo hipercubo se puede dividir en dos de dimensión $d-1$.
 - Se selecciona la posición de un bit y se agrupan todos los que tengan un cero en esa posición. El resto forma la segunda partición
- ▶ Distancia de hamming
 - Número total de posiciones de bits para los que las etiquetas de los procesadores son diferentes

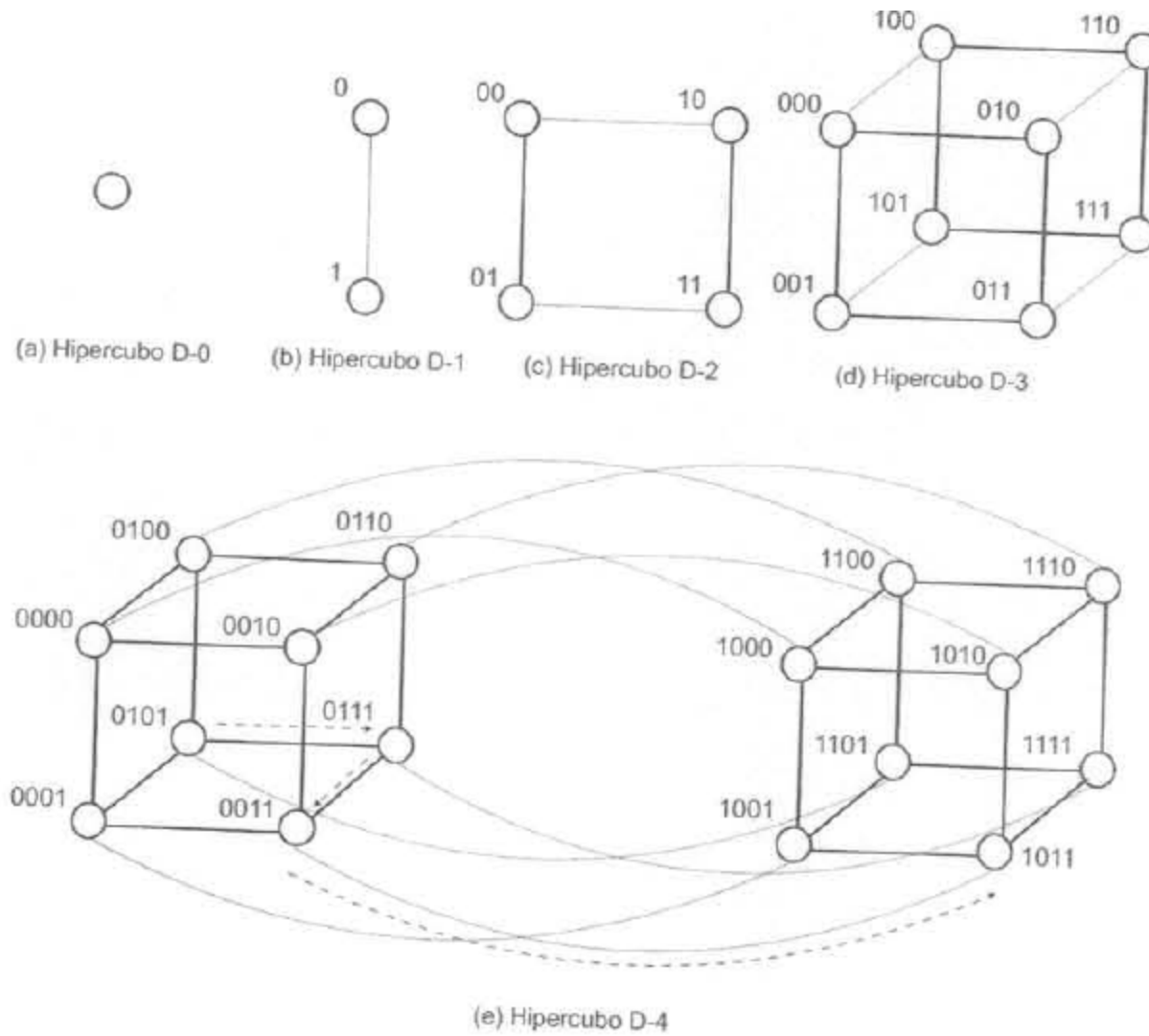


Figura 4.15: Hipercubos de dimensión: a) cero, b) uno, c) dos, d) tres, e) cuatro.

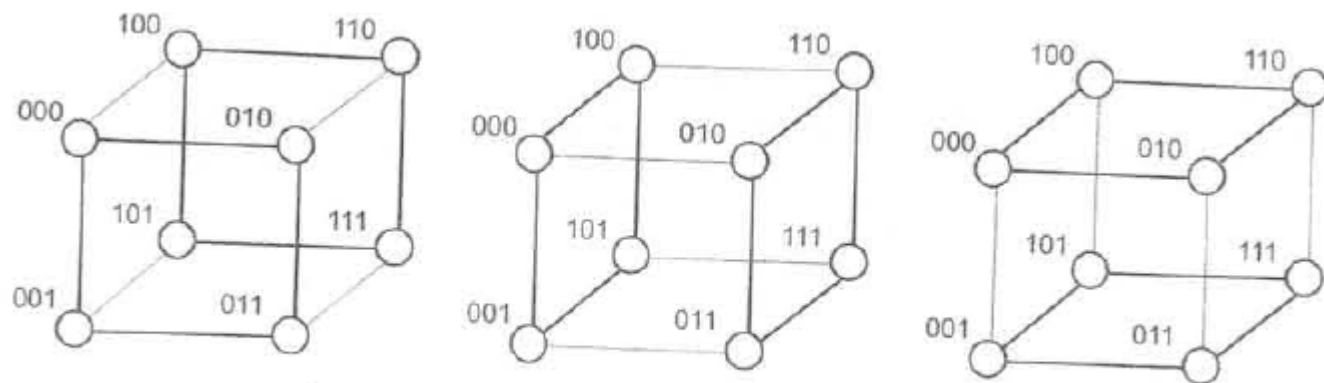
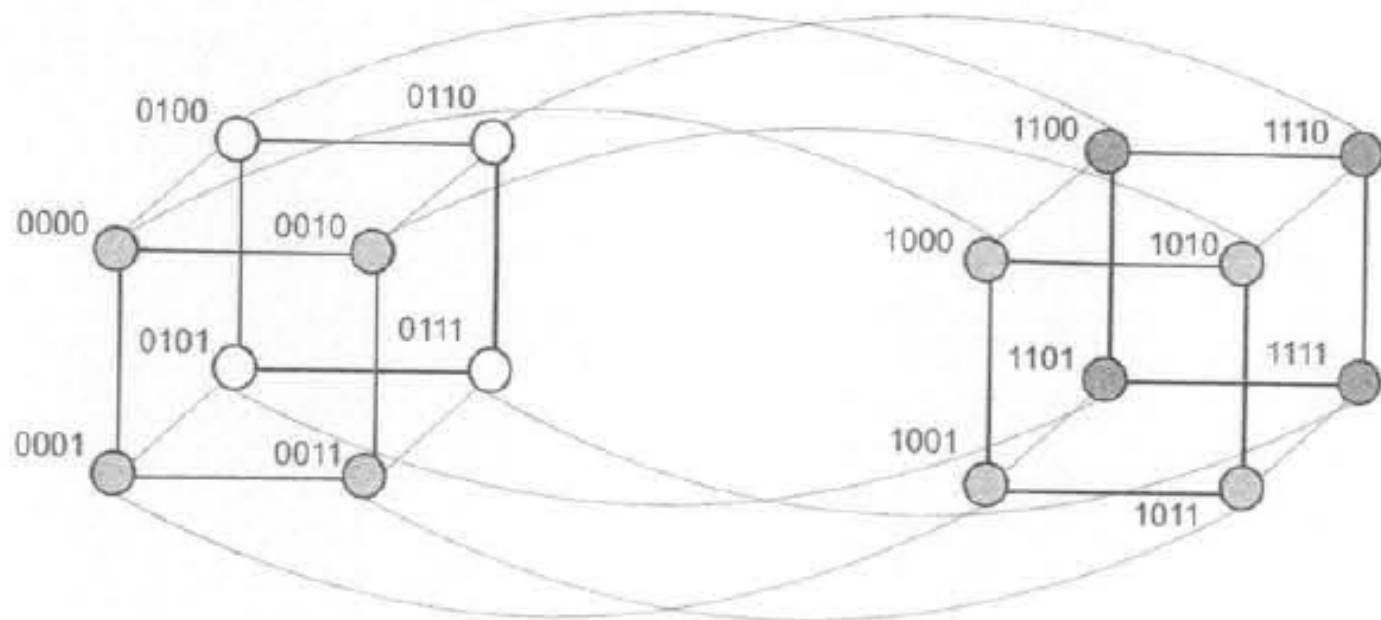
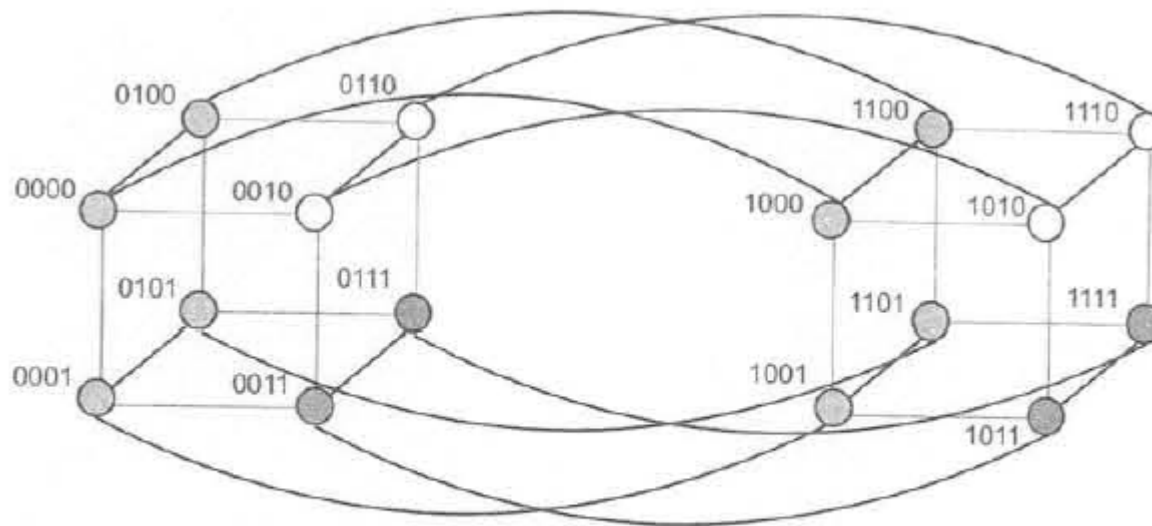


Figura 4.16: Particiones de un hipercubo tridimensional.



(a)



(b)

Figura 4.17: Divisiones de la red hipercubo.

4.4.1.2 Caracterización de redes Estáticas

- ▶ **Diámetro**
 - Máxima distancia entre dos procesadores cualesquiera
- ▶ **Conectividad**
 - Medida de la multiplicidad de caminos
- ▶ **Conectividad de arco**
 - El menor número de arcos que debe eliminarse para obtener dos redes disjuntas
- ▶ **Ancho de canal**
 - El número de bits que puede transmitirse simultáneamente
- ▶ **Velocidad de canal**
 - Velocidad máxima que se puede emitir por cada cable físico
- ▶ **Ancho de banda**
 - Velocidad de canal x ancho de canal
- ▶ **Ancho de bisección**
 - Mínimo número de enlaces que debe eliminarse para que la red se divida en dos iguales
- ▶ **Ancho de banda de disección**
 - El menor volumen de comunicaciones entre dos mitades cualesquiera con igual número de procesadores
- ▶ **Coste**
 - Número de enlaces

Tabla 4.1: Características de las redes estáticas.

RED	DIÁMETRO	CONECTIVIDAD DE ARCO	ANCHO DE BISECCIÓN	COSTE = NÚMERO DE ENLACES
Completamente conectada	1	$p - 1$	$\frac{p^2}{4}$	$\frac{p(p-1)}{2}$
Estrella	2	1	1	$p - 1$
Árbol binario	$2 \log \frac{p+1}{2}$	1	1	$p - 1$
Array lineal	$p - 1$	1	1	$p - 1$
Anillo	$\lfloor \frac{p}{2} \rfloor$	2	2	p
Mesh bidimensional	$2(\sqrt{p} - 1)$	2	$\sqrt{p}$	$2(p - \sqrt{p})$
Mesh bidimensional cerrado	$2\lfloor \frac{\sqrt{p}}{2} \rfloor$	4	$2\sqrt{p}$	$2p$
Hipercubo	$\log p$	$\log p$	$\frac{p}{2}$	$\frac{p \log p}{2}$
k-aria d-cubo cerrada	$d\lfloor \frac{k}{2} \rfloor$	$2d$	dp	dp

4.4.1.3 Redes Dinámicas

Redes basadas en bus
Redes crossbar (matriciales)
Redes multietapas

Redes basadas en Bus

- ▶ Todos los nodos comparten un único medio de comunicación
- ▶ Solo un procesador puede transmitir información al bus
- ▶ Lógica de arbitraje
 - FIFO, Round Robin, LRU
- ▶ Ancho de banda
 - Frecuencia del reloj X número de líneas existente

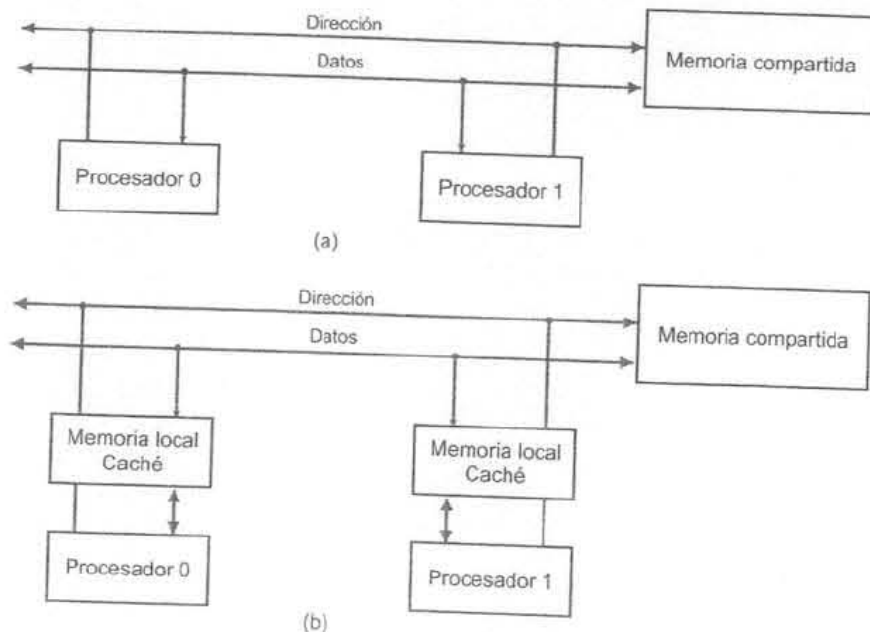


Figura 4.18: Sistema multiprocesador con conexión mediante bus.

Redes Crossbar

- ▶ Permite conectar P procesadores con q elementos mediante conmutadores
- ▶ La complejidad y el coste aumenta del orden de p^2
- ▶ Son no-bloqueantes

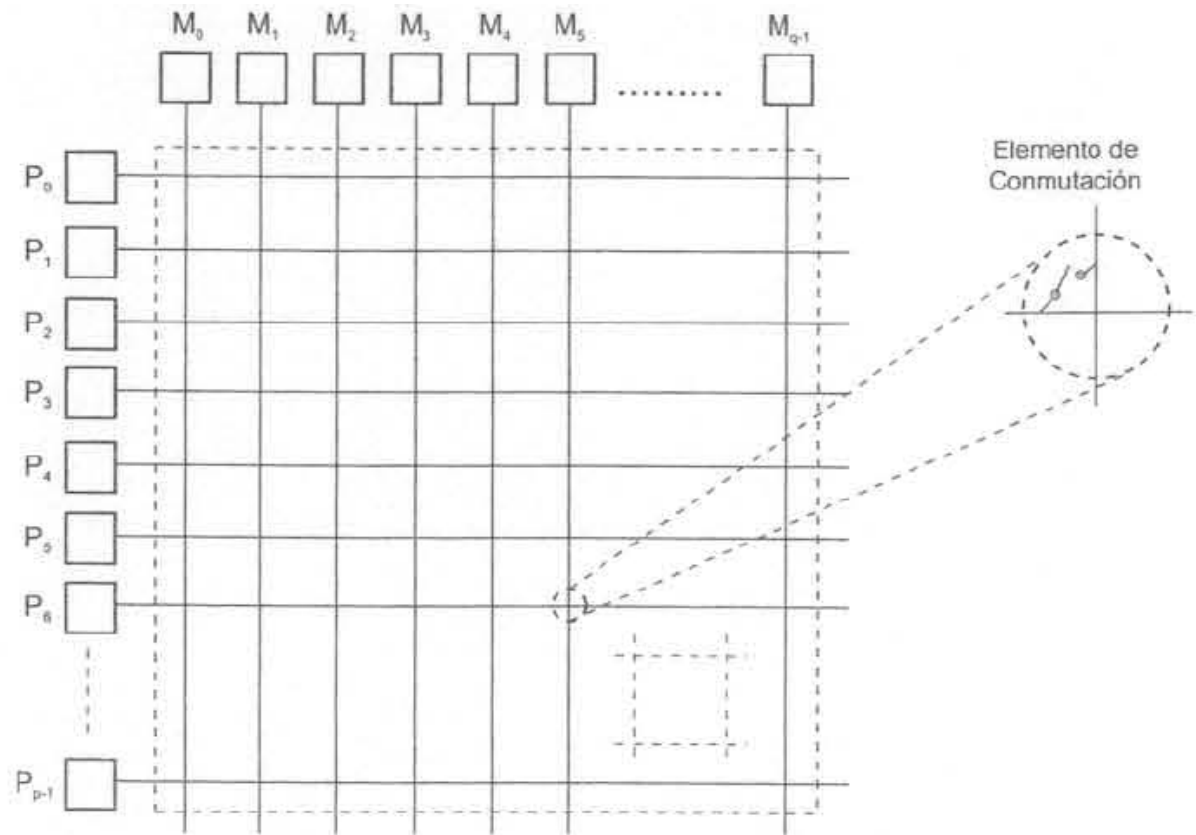


Figura 4.19: Red crossbar que conecta p procesadores con q elementos de memoria.

Redes Multietapas

- ▶ Se componen de una serie de etapas G_i compuesta por conmutadores conectados a las etapas adyacentes por conexiones estáticas C_j
- ▶ Los conmutadores con igual número de entrada y de salida se dice que tiene un orden $= a = b$. ($2 \times 2 =$ orden 2)
- ▶ Son redes bloqueantes

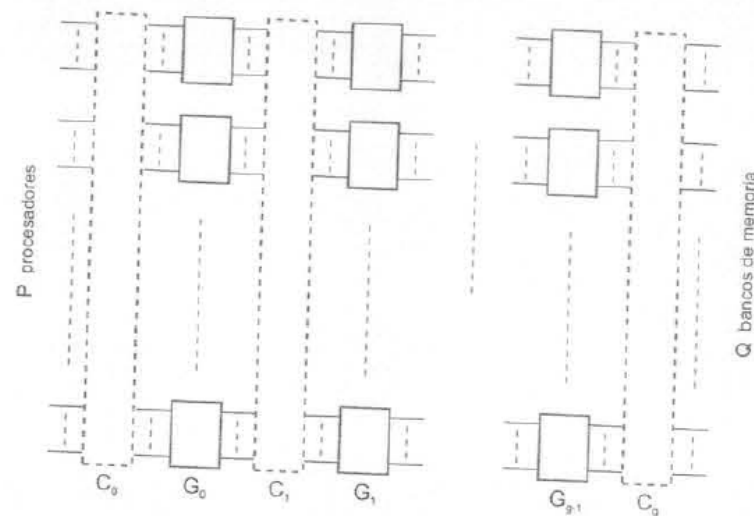


Figura 4.20: Red multietapa generalizada formada por g etapas y $g + 1$ conexiones.

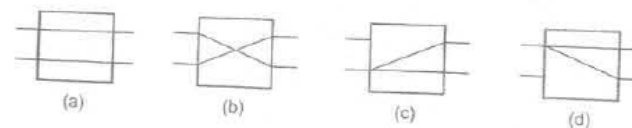


Figura 4.21: Modos de un conmutador binario: a) directo, b) cruzado, c) difusión superior, d) difusión inferior.

Red omega

- ▶ Utiliza la permutación por barajamiento perfecto (*perfect shuffle*)
 - Desplazamiento a la izquierda de los bits que representan el número
- ▶ Para p nodos de entrada y p nodos de salida
 - Etapas = $\log_2 P$
 - Número de conmutadores necesarios = $P/2 (\log_2 P)$

La permutación por barajamiento perfecto (*perfect shuffle*), σ^k , cuyo dominio es el conjunto de enteros $[0, n - 1]$, se define como

$$\sigma^k(x_{m-1}x_{m-2} \dots x_1x_0) = x_{m-2} \dots x_1x_0x_{m-1}$$

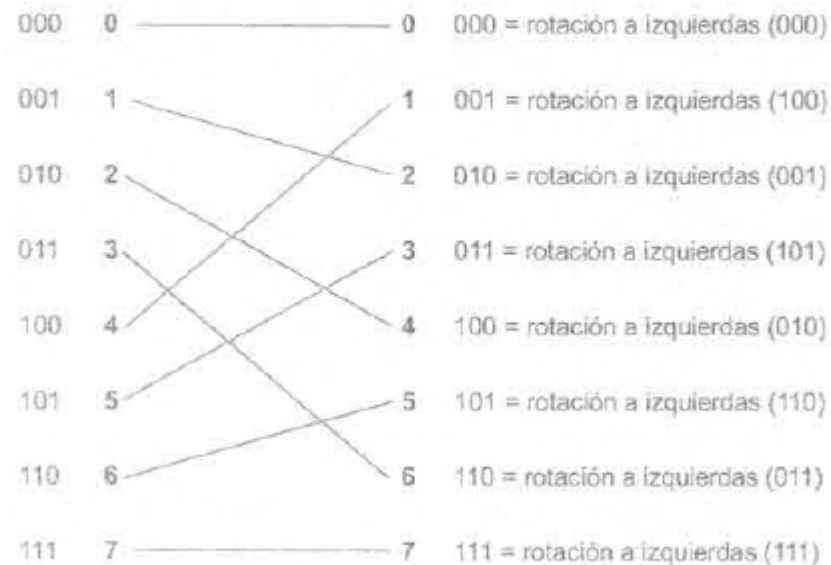


Figura 4.22: Barajamiento perfecto en una conexión de ocho entradas y ocho salidas.

Encaminamiento

- ▶ De la dirección de destino: cada etapa el bit correspondiente a la etapa si es 1 por la inferior, si en cero, por la superior
- ▶ Ejemplo :el procesador 110 al 100

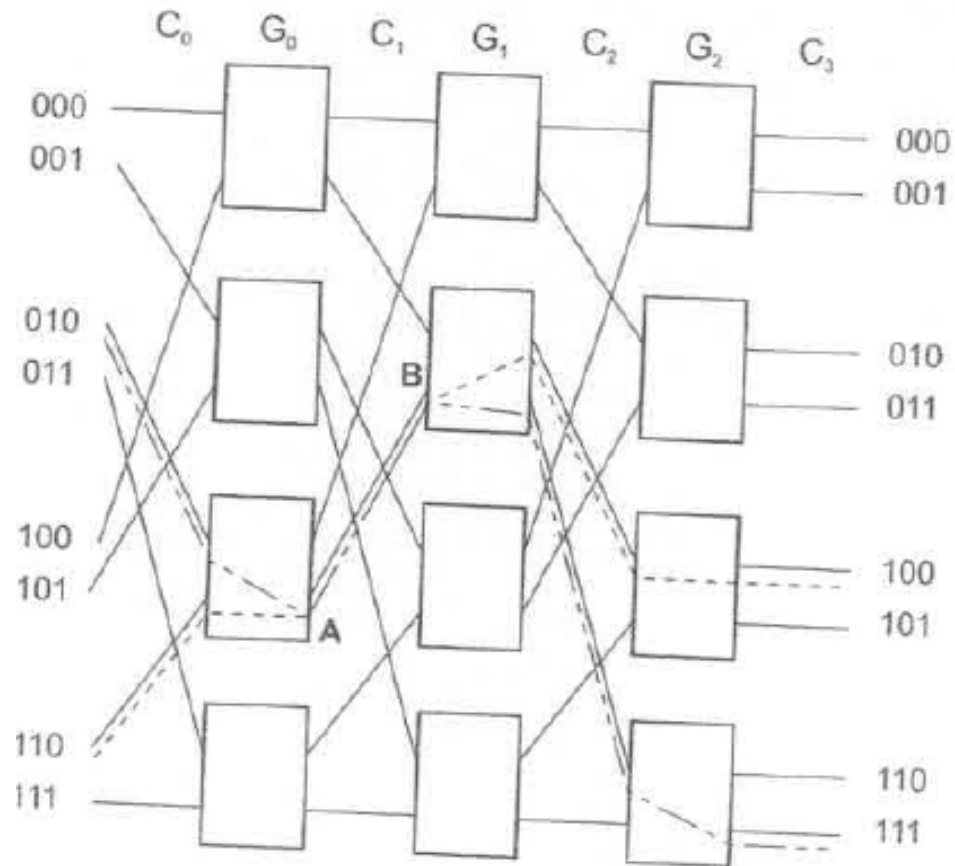


Figura 4.23: Red omega de 8 entradas y 8 salidas.

Red Baseline

- ▶ Se construyen conectando los conmutadores de la etapa i con los sub-bloques de la etapa $i + 1$
- ▶ El direccionamiento igual que la red omega

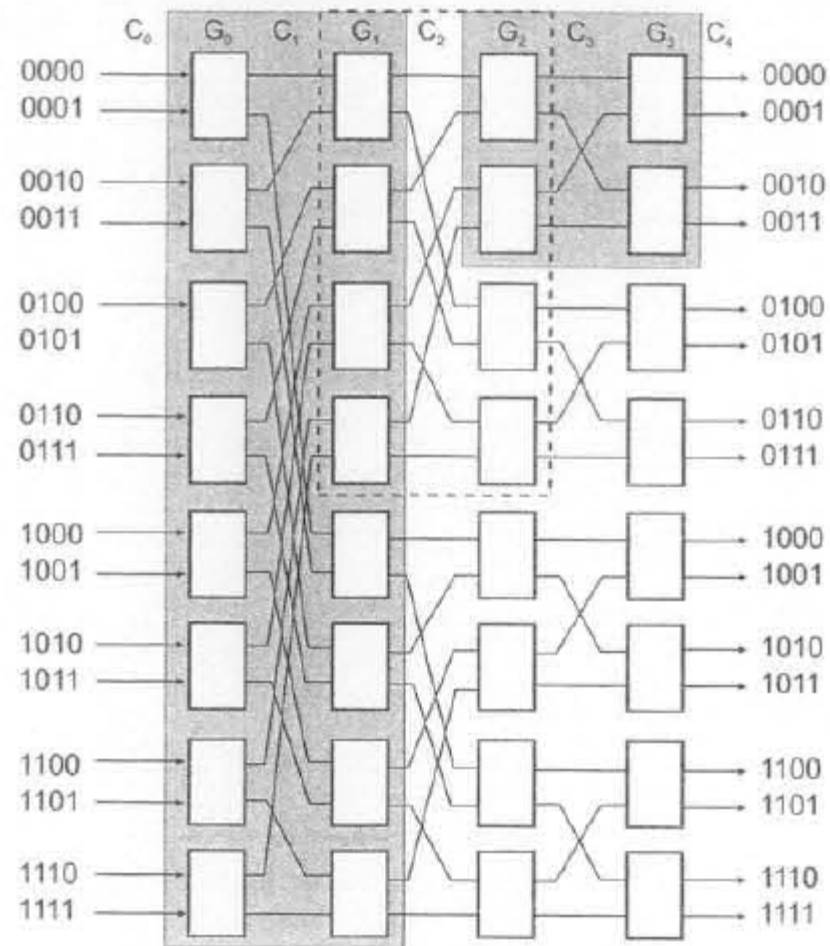


Figura 4.24: Red baseline de 16 entradas y 16 salidas.

Red butterfly

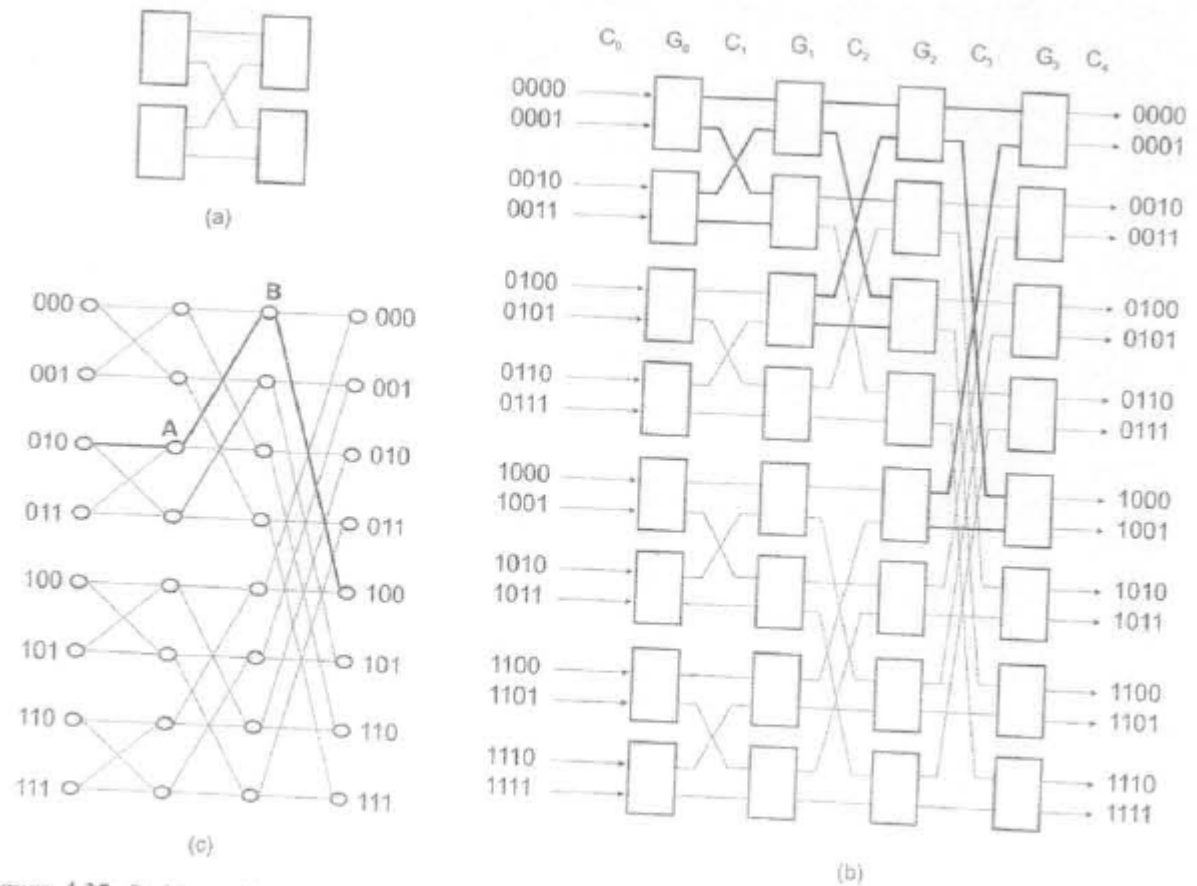


Figura 4.25: Red butterfly: a) modelo básico de permutación, b) red butterfly de 16 entradas y 16 salidas, c) diagrama de conmutadores de la red butterfly.

4.4.1.4 Comparación de rendimientos

- ▶ Las multi-etapas son una solución aceptable teniendo en cuenta precio/prestaciones

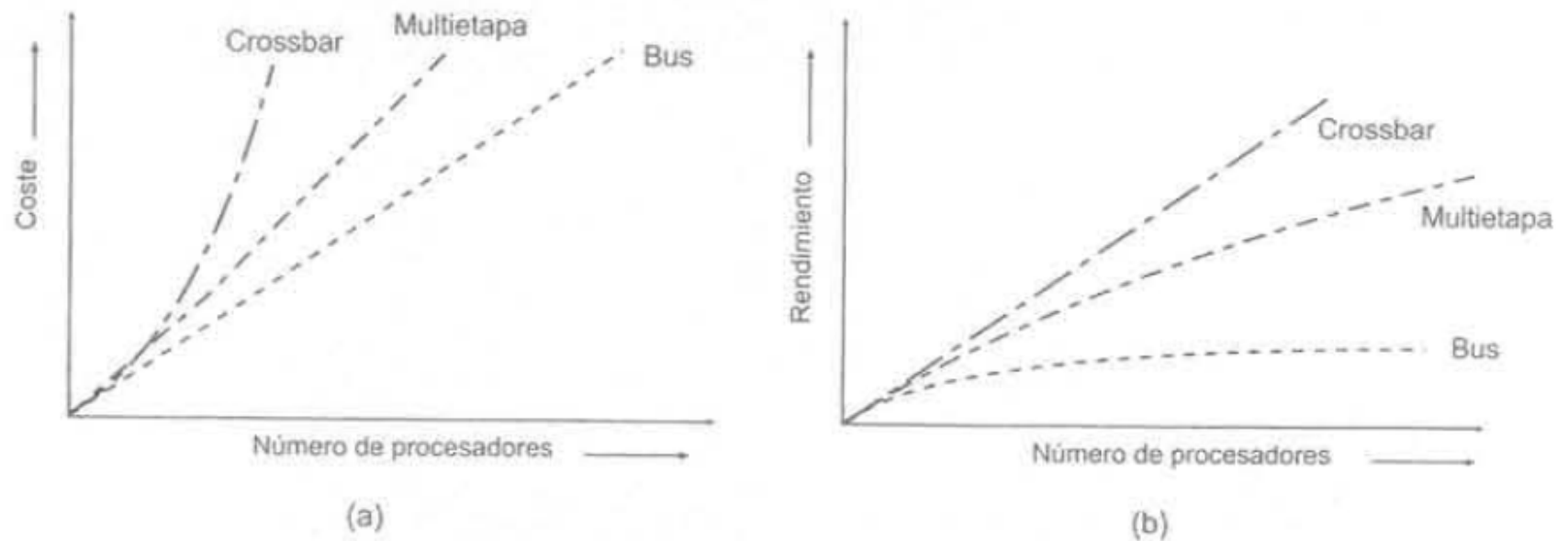


Figura 4.26: Comparativa de las arquitecturas crossbar, bus y multi-etapa: a) Coste, b) Prestaciones.

4.4.2 Protocolo de coherencia de cache

► Coherencia de cache

- Un sistema de memoria es coherente si el valor devuelto por una operación de lectura sobre una dirección de memoria es siempre el mismo valor que el almacenado en la última operación de escritura realizada sobre esa misma dirección, independientemente de qué procesador realice las operaciones
- Soluciones
 - Solo permitir acceso locales o remotos
- Causas
 - Modificar datos compartidos
 - Migración de procesos
 - Por uso de E/S mediante DMA
- Soluciones
 - Invalidar
 - Actualizar

► Invalidar

- Consiste en invalidar las copias en las caches de los datos modificados

► Actualizar

- Consiste en actualizar todas las copias en las caches del dato que se acaba de modificar

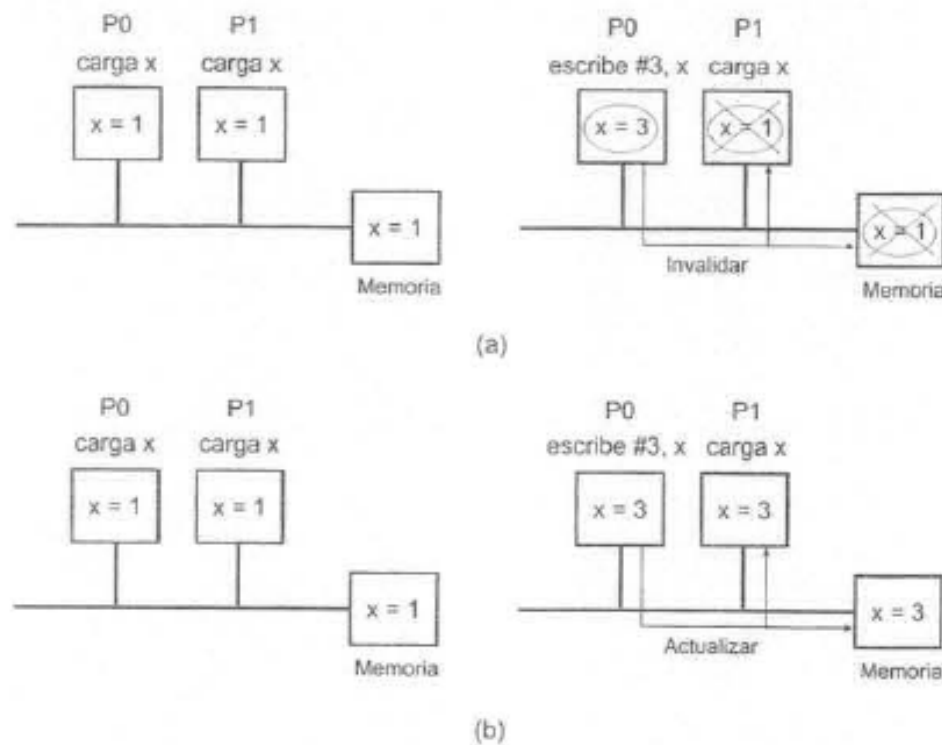
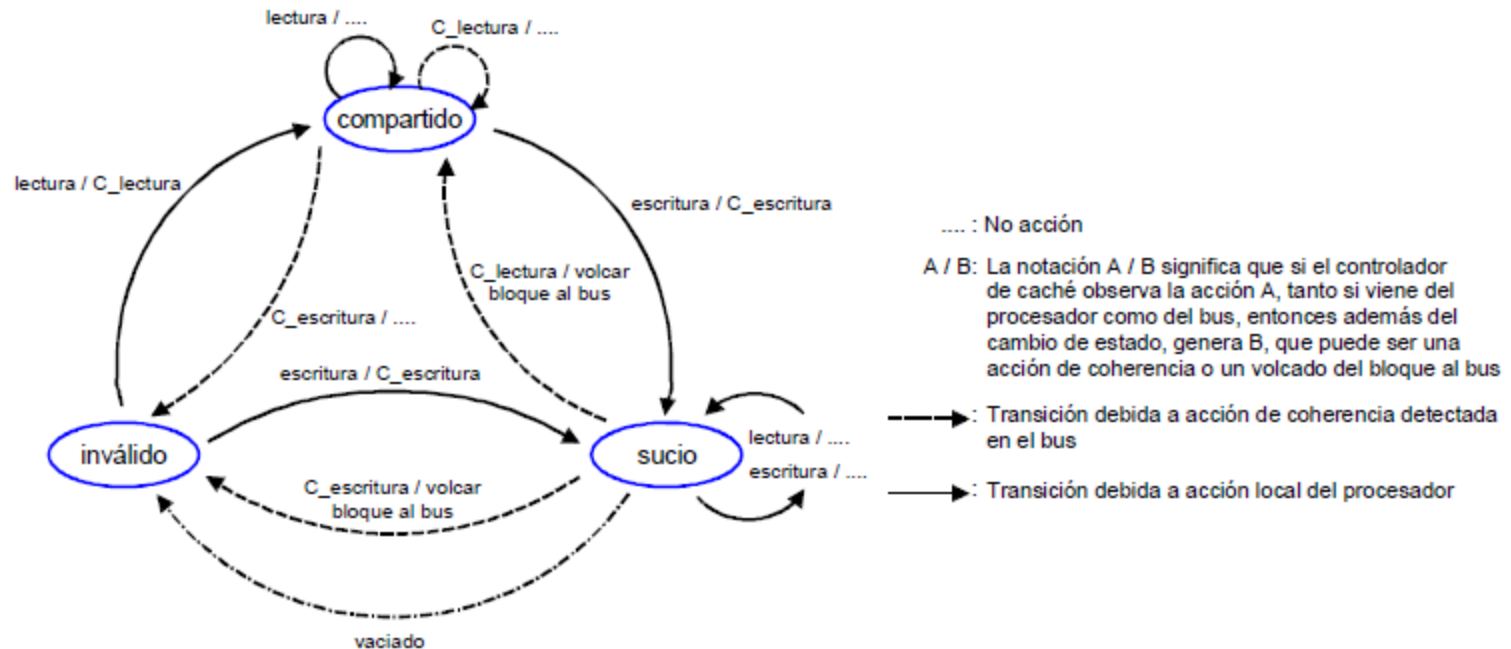


Figura 4.27: Protocolos de coherencia de caché en sistemas multiprocesador: a) invalidación, b) actualización.

Diagrama de estados



Sistema snoopy o de vigilancia de bus

- ▶ Cada procesador monitoriza el trafico de la red
- ▶ La caché de cada procesador tiene asociada unas etiquetas que utiliza para determinar el estado de su bloque
- ▶ Limitación: el ancho del bus

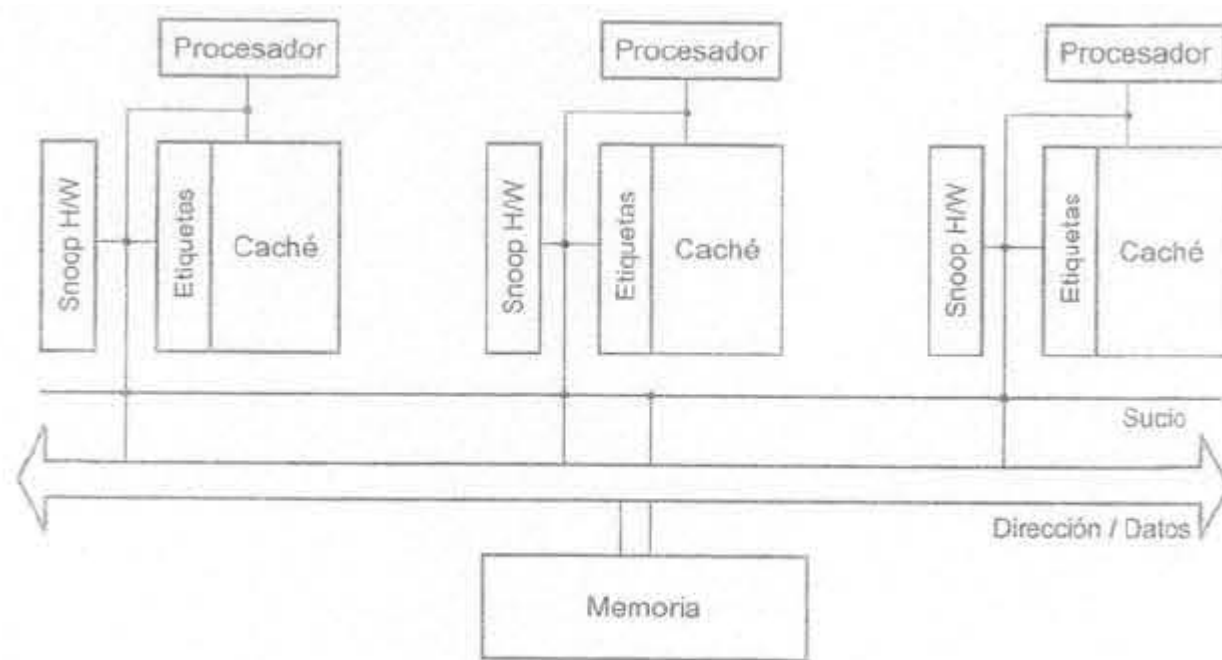


Figura 4.29: Sistema snoopy.

Sistema basado en directorios

- ▶ Usa un sistema de mapa de bits para almacenar la localización de las copias de caché que tiene cada procesador
- ▶ DIRECTORIO CENTRALIZADO
 - EL MAPA DE BITS ESTÁ EN MEMORIA PRINCIPAL
- ▶ DIRECTORIO DISTRIBUIDO
 - El mapa de bits está en las memorias locales

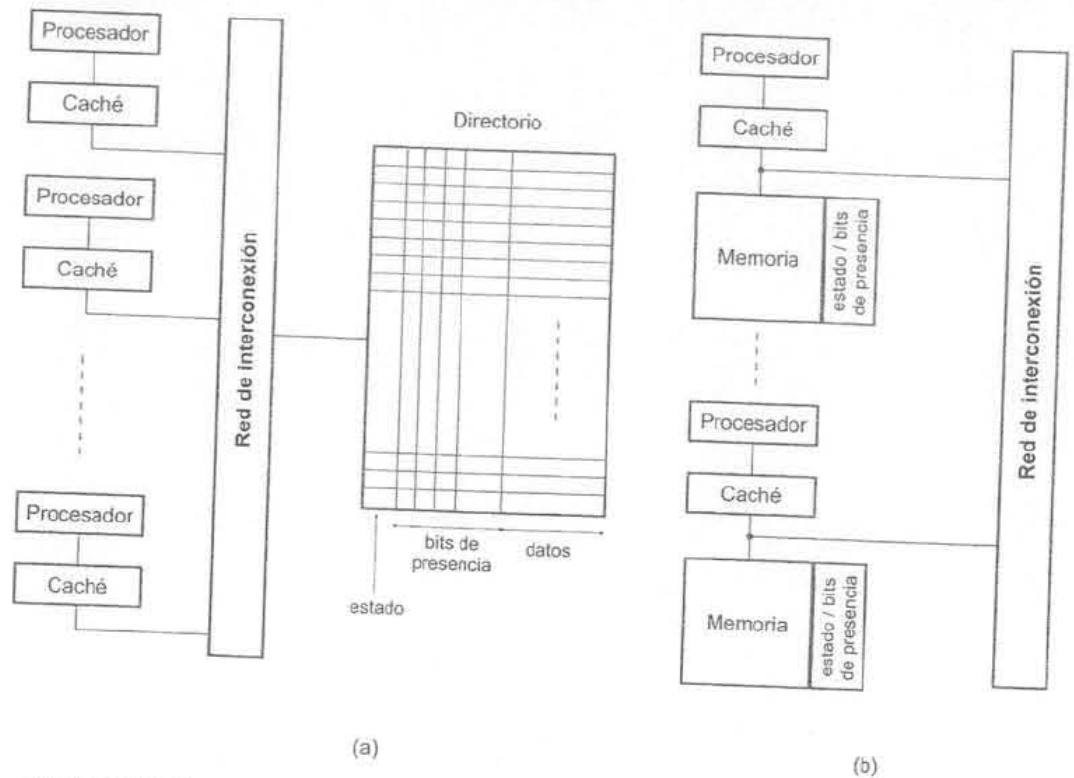


Figura 4.30: Sistema basado en directorios: a) directorio centralizado, b) directorio distribuido.

4.5 Sistema de memoria distribuida

- ▶ Cada procesador tiene su propia memoria local o privada solo accesible por su procesador
- ▶ La comunicación se realiza por paso de mensajes
- ▶ MPP (*Massively Parallel Processor*)
 - Múltiples CPUs comunicadas por un bus de datos
- ▶ Cluster
 - Múltiples ordenadores enlazadas por una red de interconexión
 - Cluster no dedicado NOW
 - Estaciones de trabajo con switch
 - Cluster dedicado o Boewulf
 - Conjunto minimalista de nodos conectados por un medio de comunicación barato (rj-45)
 - Fuertemente dependiente de la arquitectura

Ejemplo de arquitectura de memoria distribuida

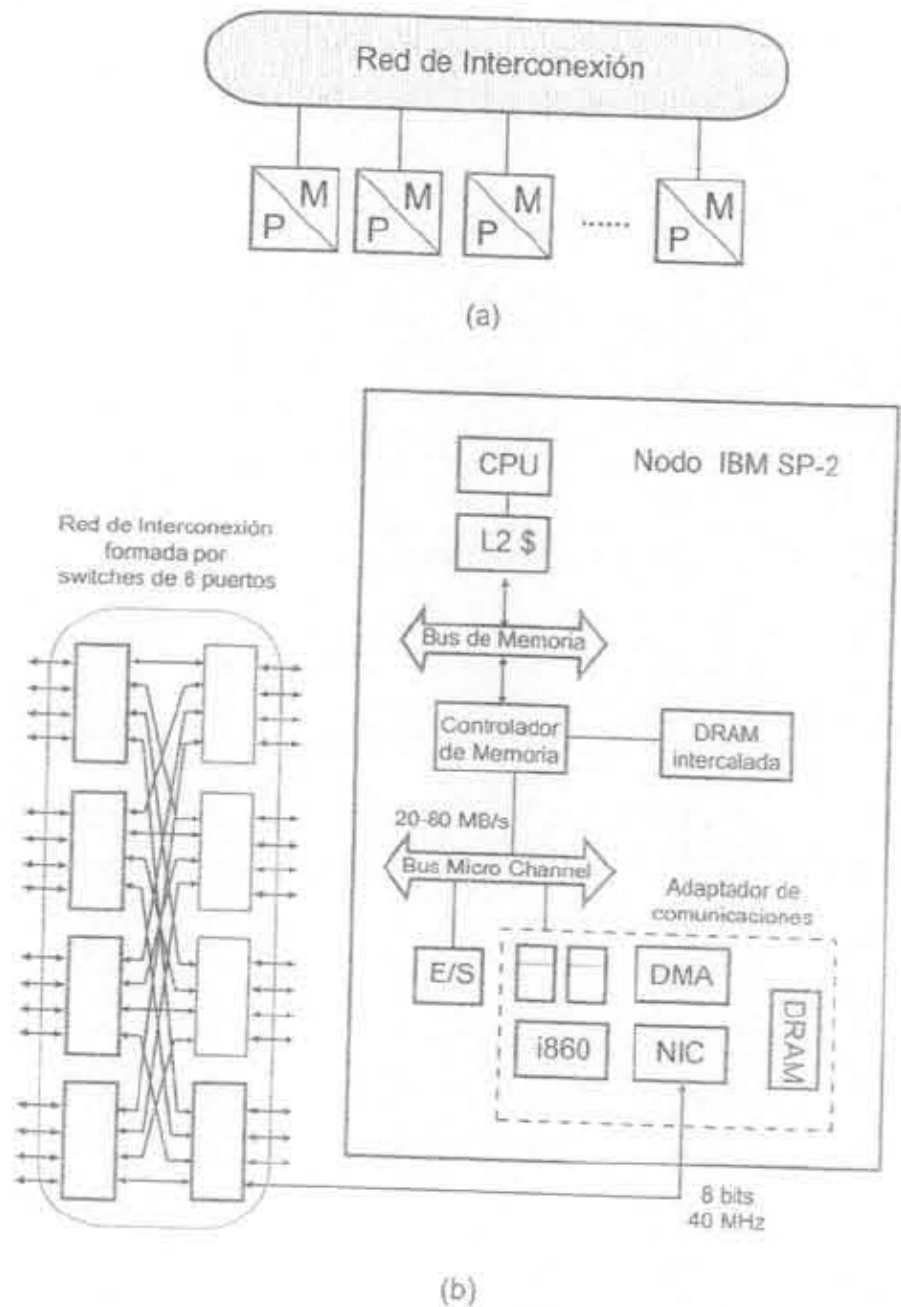


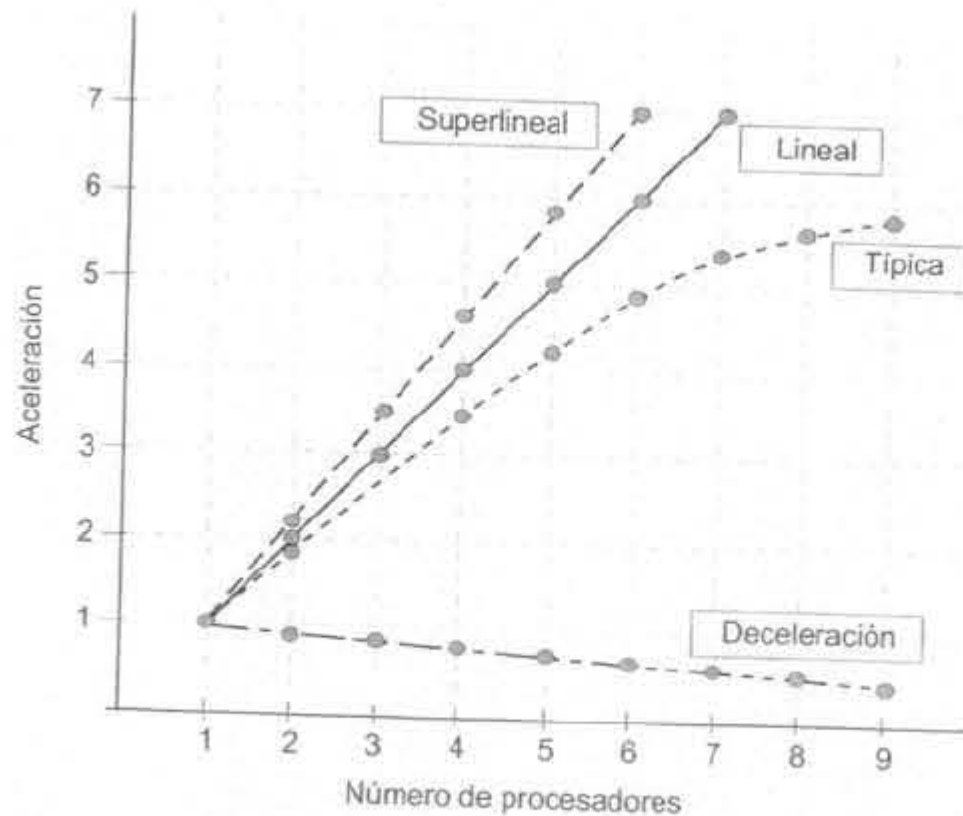
Figura 4.31: a) Arquitectura de memoria distribuida, b) IBM SP-2.

4.5.1 consideraciones generales sobre los cluster

- ▶ Pueden presentar distintas topologías
 - Lineales, anillos, estrella
- ▶ Pueden utilizar un switch o un hub
- ▶ ¿Porqué cluster?
 - Mas económicos que los ordenadores vectoriales y MPP
 - Mejora y abaratamiento de los componentes de redes
 - Escalables, bajo coste, hardware convencional, software de libre distribución

4.5.3 ¿Cuándo y cómo utilizar un cluster

- ▶ Memoria compartida: alto acoplamiento. (semáforos, monitores..)
- ▶ Cluster: control mas sencillo, pero no soporta un alto grado de acoplamiento = muchos mensajes



: Curva de aceleración frente al número de procesadores para distintos casos de aplicaciones paralelas.

4.6 Rendimiento y coste en sistemas paralelos

- ▶ Granularidad: tamaño de los procesos

$$\frac{\text{tiempo de computación}}{\text{tiempo de comunicación}} = \frac{t_{comp}}{t_{com}}$$

- ▶ Factor de aceleración (speedup)

Una medida del rendimiento relativo entre un sistema multiprocesador y un sistema con un único procesador es el factor de aceleración denominado *speedup*, $S(M)$, que se define como:

$$S(M) = \frac{t_s}{t_p}$$

donde t_s es el tiempo de ejecución de un programa en un único procesador y t_p es el tiempo de ejecución en un sistema paralelo con M procesadores. El parámetro $S(M)$ da el incremento de velocidad que se obtiene cuando se utiliza un sistema con M procesadores en lugar de un único procesador.

$$S(M) = \frac{\text{Número de pasos computacionales utilizando un procesador}}{\text{Número de pasos computacionales paralelos con } M \text{ procesadores}}$$

Factores

- ▶ Periodos en los que no todos los procesadores están realizando trabajo útil
- ▶ Los cálculos adicionales
- ▶ El tiempo de comunicación para enviar mensajes

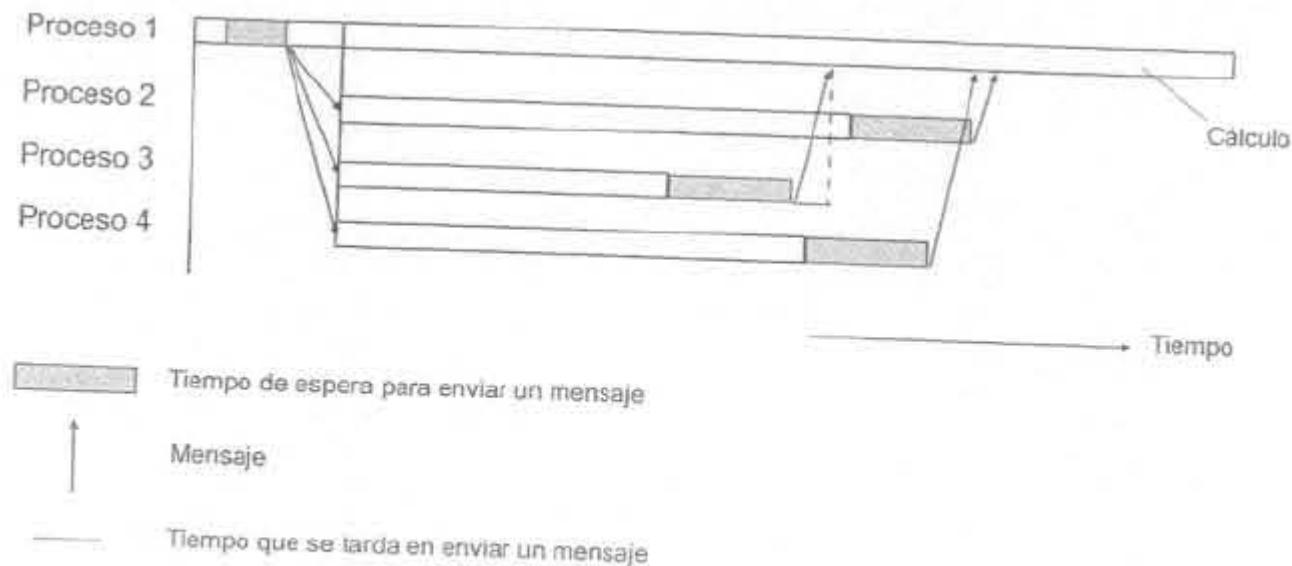


Figura 4.33: Diagrama espacio-temporal de un programa de paso de mensajes.

4.6.1.3 Ley de Amdahl

Si se llama f a la fracción del programa que no se puede dividir en tareas paralelas, $0 \leq f \leq 1$, y se considera que no hay sobrecarga cuando el programa se divide en tareas paralelas, el tiempo de computación necesario para ejecutar el programa en M procesadores ($t_p(M)$) vendrá dado por la siguiente expresión:

$$t_p(M) = f \cdot t_s + \frac{(1-f) \cdot t_s}{M}$$

En la Figura 4.34 se muestra gráficamente la paralelización en M procesadores de un programa

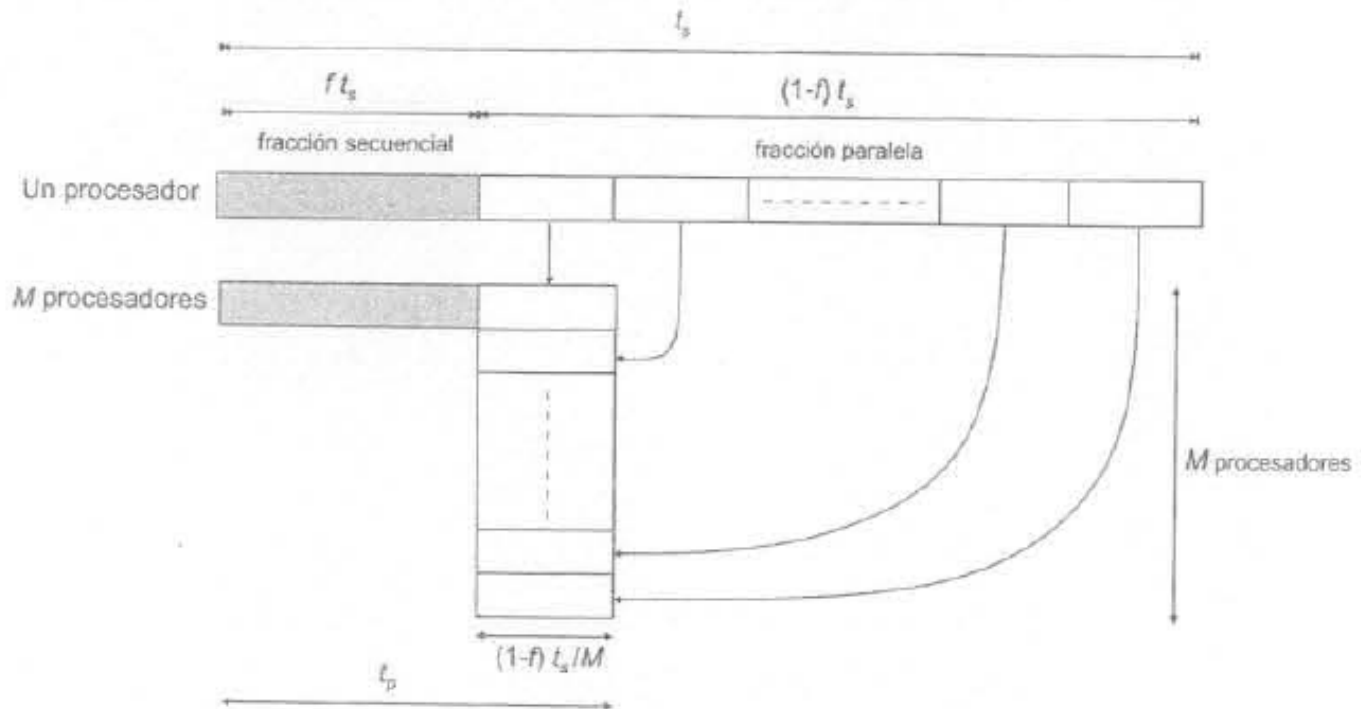


Figura 4.34: Programa secuencial paralelizado. Ley de Amdahl

► Límite de Amdahl $1/f$

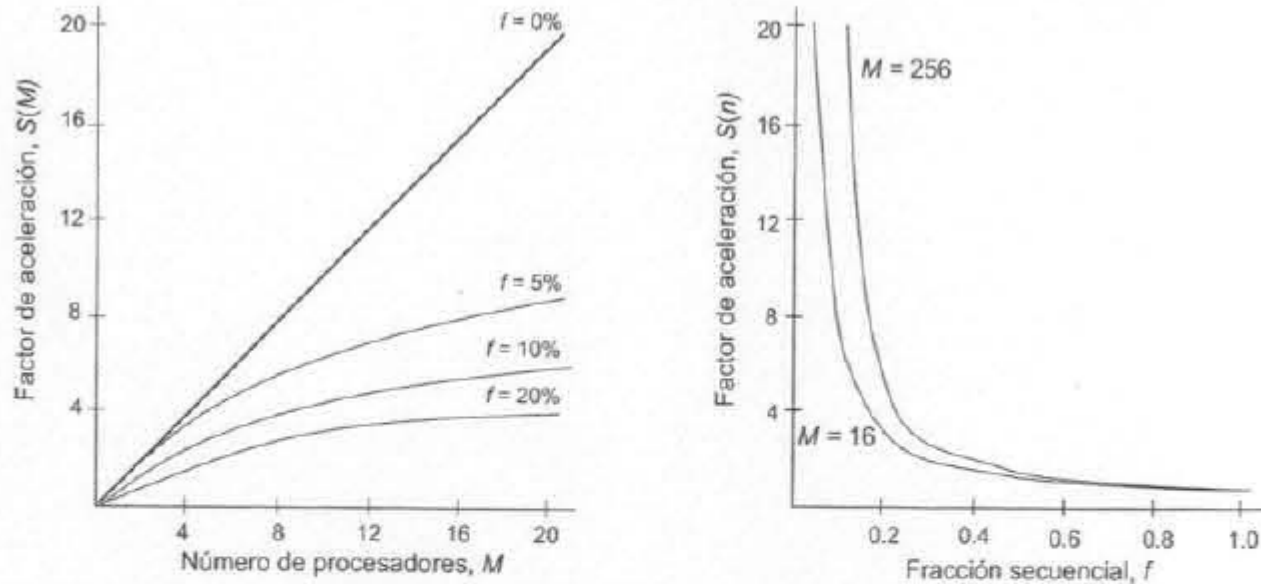


Figura 4.35: Gráficas del factor de aceleración, $S(M)$.

Del estudio de la ley de Amdahl, se podría definir el límite de Amdahl ($\frac{1}{f}$) como el mayor factor de aceleración posible cuando el número de procesadores disponibles tiende a infinito. Por ejemplo, si un programa contiene un 10 % de código no paralelizable ($f = 0,1$), el límite de Amdahl sería 10 ($\frac{1}{0,1}$), independientemente del número de procesadores del sistema.

4.6.1.4 Eficiencia

- ▶ La eficiencia da la fracción de tiempo que se utilizan los procesadores durante la computación

Si se normaliza la aceleración de un programa paralelo dividiéndola entre el número de procesadores se obtiene la eficiencia E de un sistema:

$$E = \frac{\text{Tiempo de ejecución utilizando un único procesador}}{\text{Tiempo de ejecución utilizando un multiprocesador} \times \text{número de procesadores}} = \frac{t_s}{t_p \times M}$$

Si se tiene en cuenta la ecuación anterior se puede expresar la eficiencia E en función del factor de aceleración $S(M)$ mediante la siguiente expresión:

$$E = \frac{S(M)}{M} \times 100 \% = \frac{1}{1 + (M - 1) \cdot f} \times 100 \%$$

4.6.1.5 Coste

El coste C (o trabajo) de una computación, se define como:

$$C = \text{Tiempo de ejecución} \times \text{Número de procesadores utilizados}$$

El coste de una computación secuencial es simplemente su tiempo de ejecución t_s , y el de una computación paralela es $t_p \times M$. El tiempo de ejecución paralelo, t_p , viene dado por $\frac{t_s}{S(M)}$. Por lo tanto, el coste de una computación paralela se puede expresar como:

$$C = \frac{t_s \cdot M}{S(M)} = \frac{t_s}{E}$$

Un coste óptimo de un algoritmo paralelo es aquel que es proporcional al coste (tiempo de ejecución) que tiene en un sistema con un único procesador.

4.6.1.6 Escalabilidad

- ▶ Un sistema es escalable si permite ampliar su tamaño para obtener una mejora de rendimiento
- ▶ La escalabilidad puede ser de:
 - hardware o
 - algorítmica
- ▶ Un sistema es escalable si el rendimiento del mismo se incrementa linealmente con relación al número de procesadores usados en cierta aplicación
- ▶ Parámetros
 - Tamaño del sistema (nº de procesadores)
 - Frecuencia del reloj
 - Tamaño del problema
 - Tiempo de CPU
 - Capacidad de memoria
 - Pérdidas (*overhead*) de comunicación
 - Coste del sistema

4.6.1.7 Balance de carga

- ▶ Distribuir equitativamente la carga computacional entre todos los procesadores disponibles
- ▶ Balance de carga estático
 - El balance se realiza antes de la ejecución
 - Es muy difícil estimar de forma precisa el tiempo de ejecución antes de ejecutarlo
 - Puede haber retrasos en las comunicaciones que pueden variar en distintas circunstancias
 - El problema puede necesitar distinto número de pasos computacionales (algoritmos de búsquedas)
- ▶ Balance de carga dinámico
 - El balance se realiza durante la ejecución
 - Balance de carga centralizado
 - Si no hay muchos esclavos y son intensiva computacionalmente
 - Balance de carga distribuido
 - Se utiliza varios maestros

Balance de carga

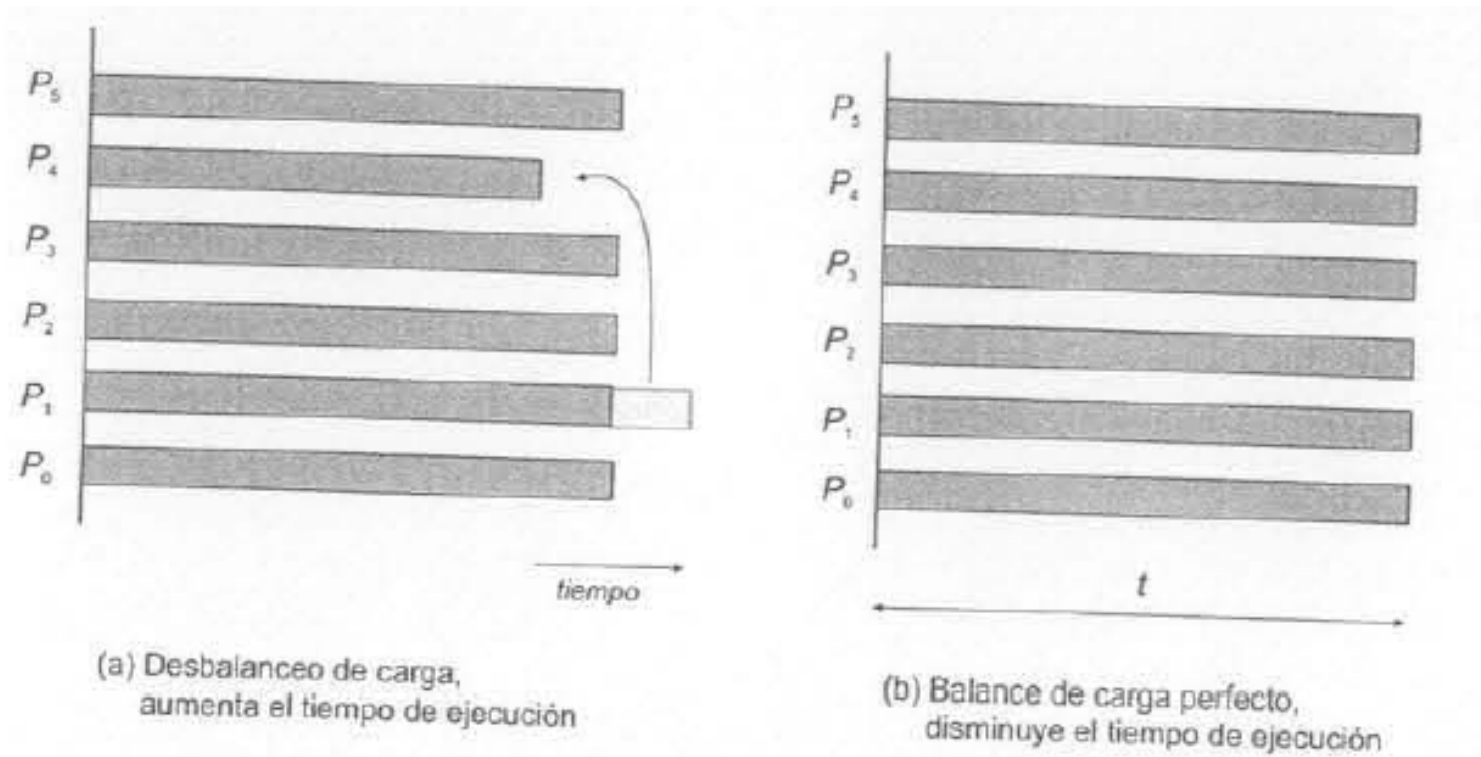


Figura 4.36: Balance de carga.

Balance de carga centralizado/distribuido

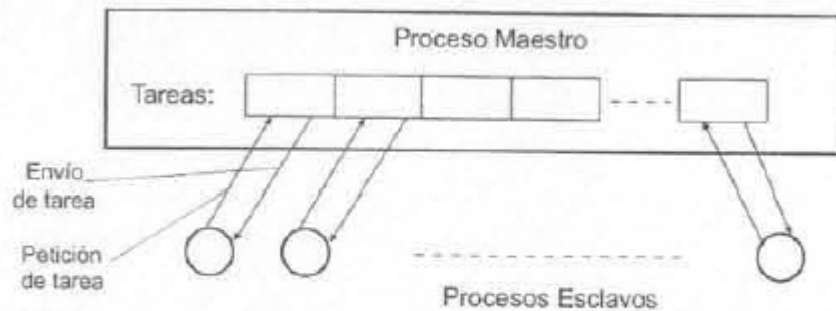


Figura 4.37: Balance de carga dinámico centralizado.

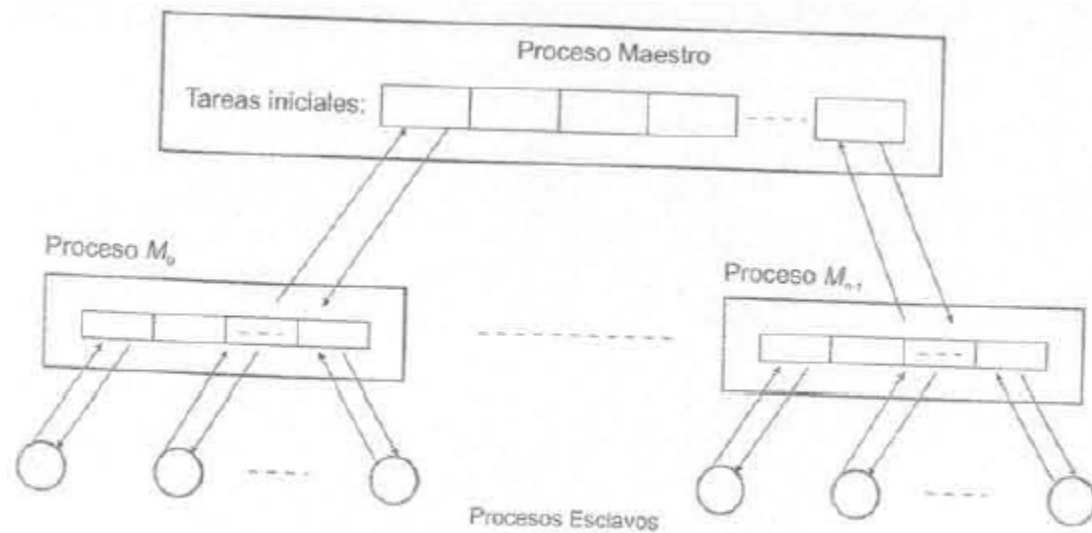


Figura 4.38: Balance de carga dinámico distribuido.

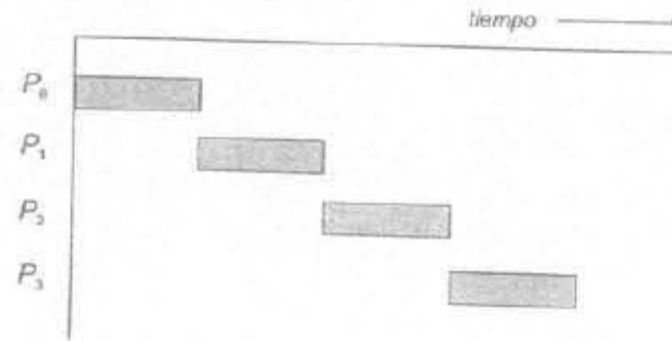
4.6.2 Coste de la comunicación mediante paso de mensaje

- ▶ Latencia en comunicación
 - Tiempo de iniciación (t_s)
 - Tiempo de salto (T_h)
 - Tiempo de transferencia por palabra (t_w)
- ▶ Almacenamiento y reenvío (*store-and-forward*)
 - Cada nodo lo reenvía únicamente cuando lo ha recibido y almacenado completamente

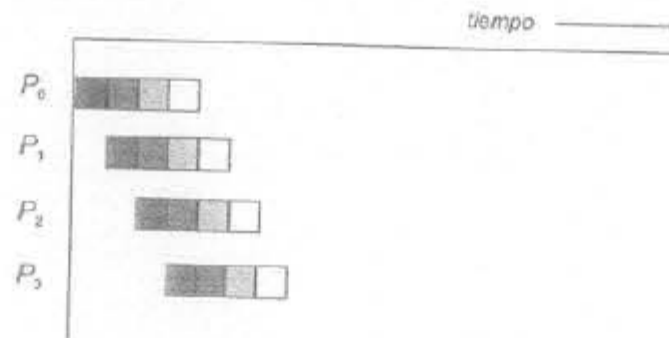
$$t_s + (mt_w + t_h)l \quad \Rightarrow \quad t_{com} = t_s + mlt_w$$

- ▶ Corte y continuación (*cut-through*)
 - Divide cada mensaje en un número fijo de unidades llamados dígitos de control de flujo (*flow control digits o flits*)
 - Antes de enviar el primer *flits* se establece la ruta mediante un paquete denominado *tracer*

$$t_{com} = t_s + lt_h + mlt_w$$



(a)



(b)

Figura 4.39: Transferencia de un mensaje: a) enrutamiento mediante almacenamiento y reenvío (*store-and-forward*), b) enrutamiento mediante *cut-through*.