

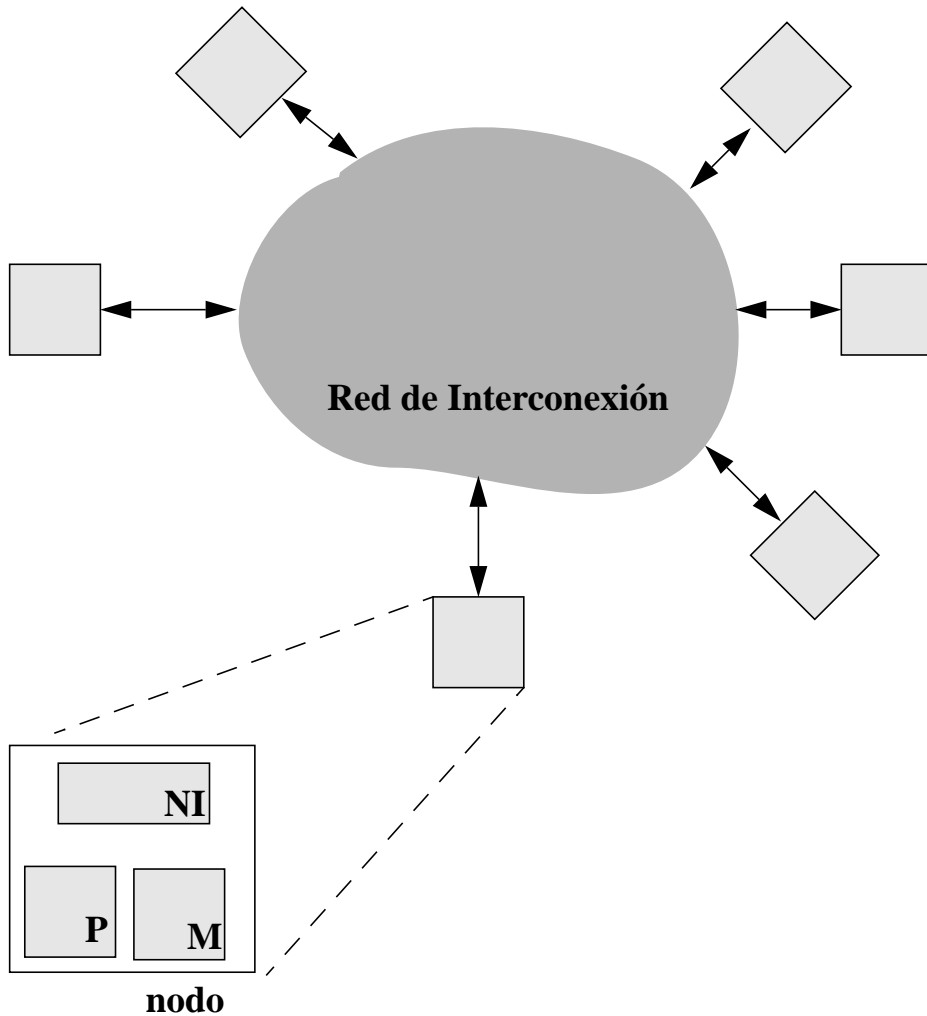
Multicomputadores

- Características Generales**
- Redes de Interconexión**
- Topologías de Interconexión**
- Estrategias de Conmutación y Control de Flujo**
- Algoritmo de Encaminamiento**
- Organización del Switch**
- Interfaz Procesador-Red**
- Networks-of-Workstations (NOW)**



Características Generales

□ Organización

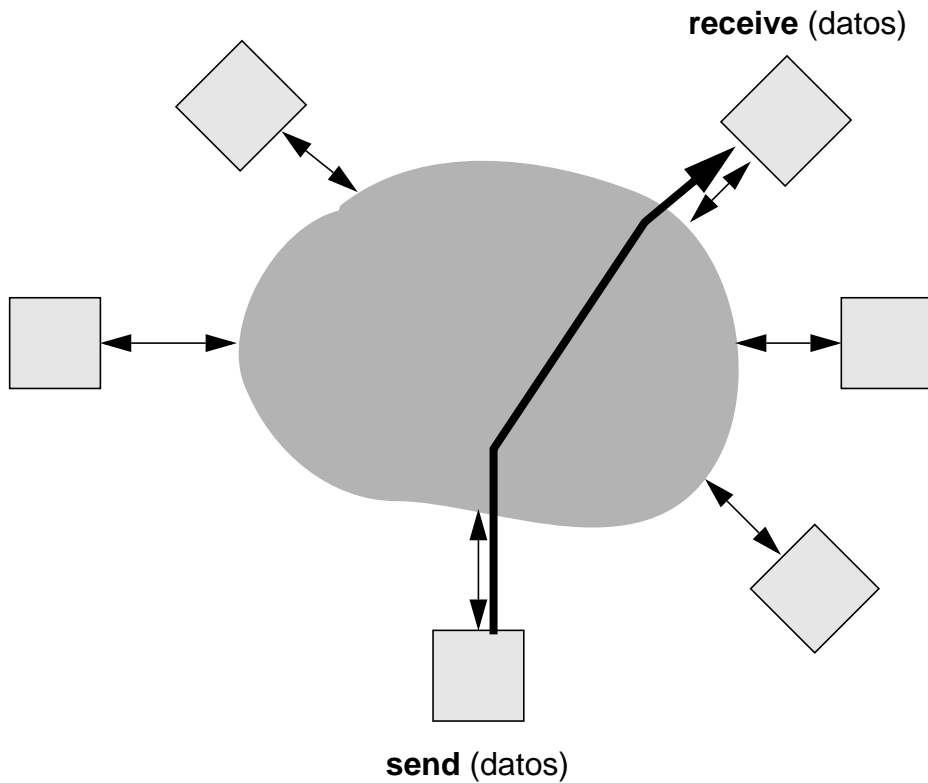


Principio básico

- ✓ Muchos procesadores baratos mejor que pocos procesadores caros (SMP)

Características Generales

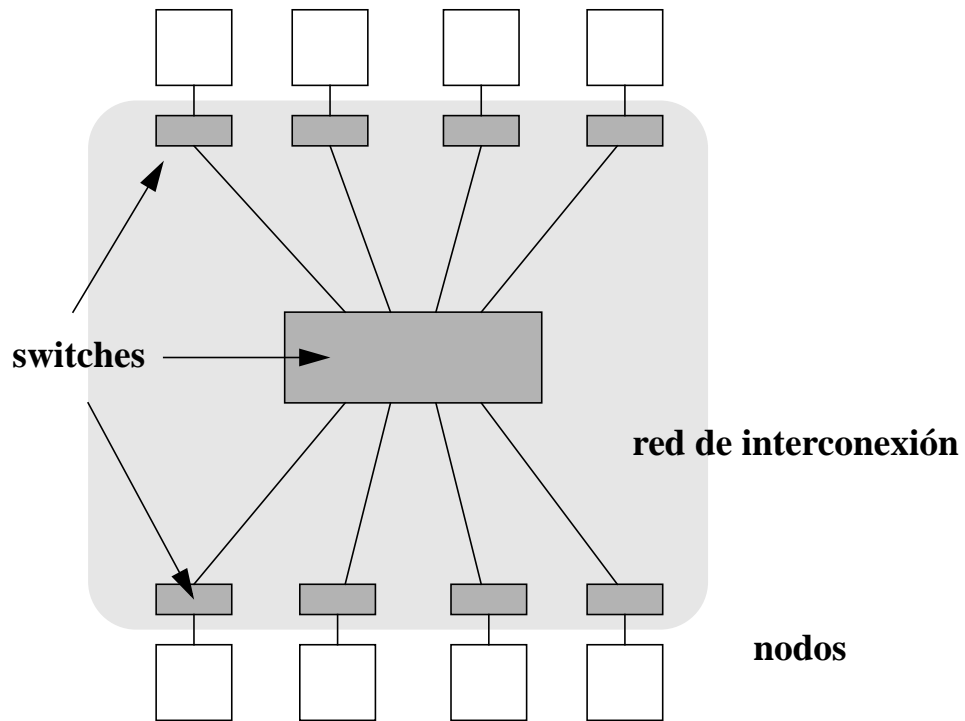
□ Paso de mensajes



- ✓ Está generalmente aceptado que “paso de mensajes” es más complejo que “variables compartidas”
- ✓ Estandar para paso de mensajes: MPI

Redes de Interconexión

❑ Redes Indirectas

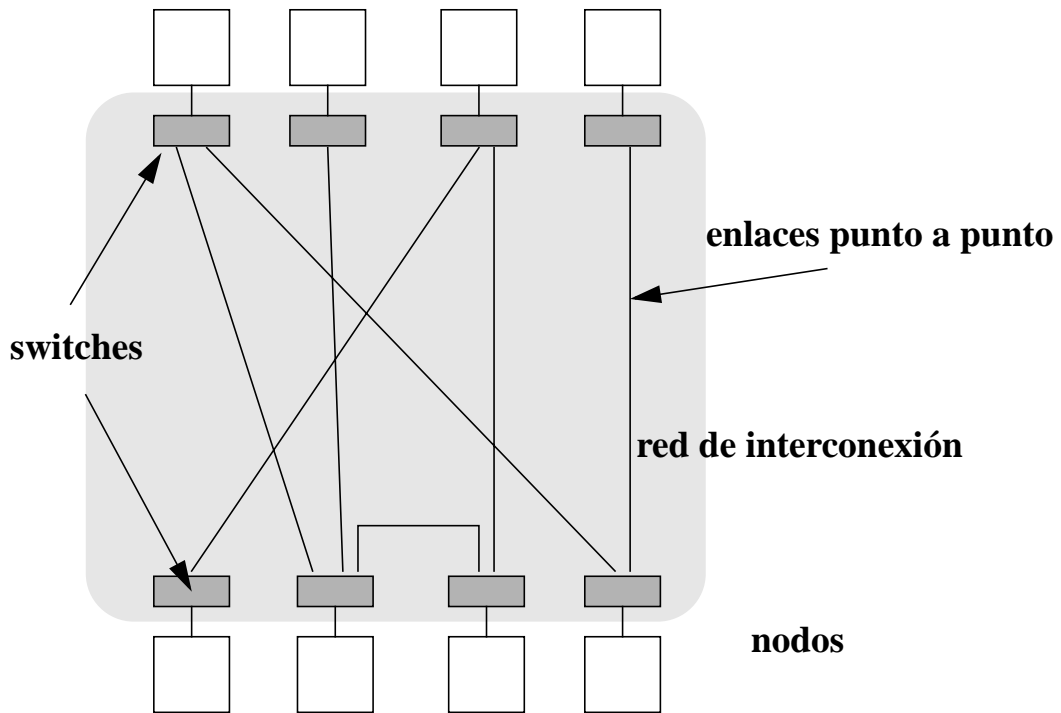


✓ Buena conectividad

✓ Escalabilidad limitada

Redes de Interconexión

❑ Redes Directas



- ✓ Buena escalabilidad
- ✓ Conectividad limitada

Redes de Interconexión

❑ Aspectos Esenciales



Topología

Patrón de interconexión de los nodos/switches.



Algoritmo de Encaminamiento

Estrategia para seleccionar el camino entre el nodo fuente y el nodo destino.



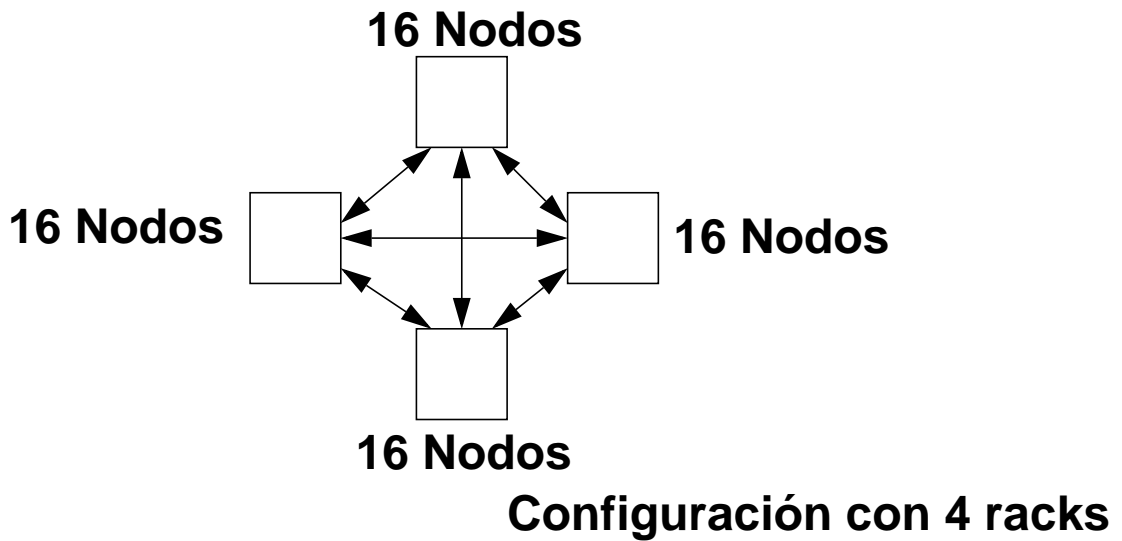
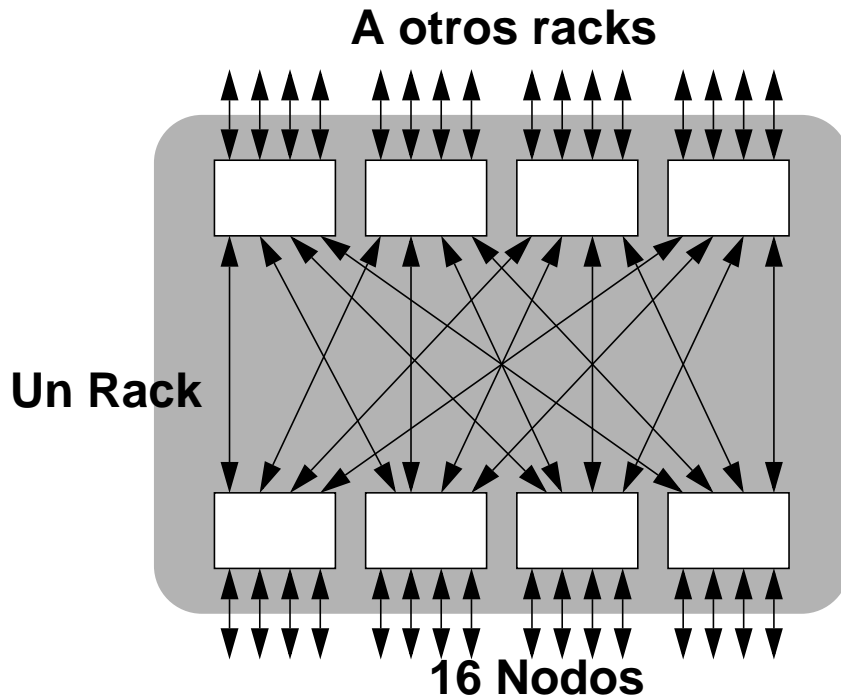
Mecanismo de conmutación y control de flujo

Cómo y cuándo se mueven los datos a través del camino.



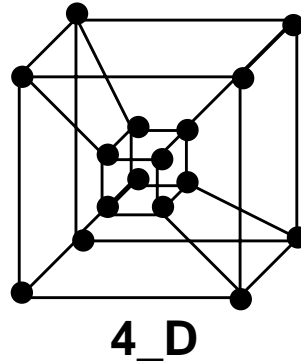
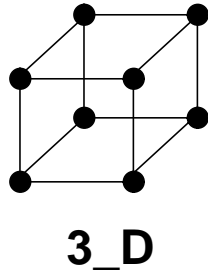
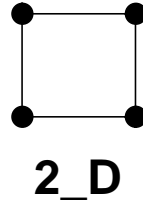
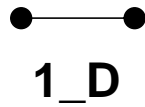
Topologías de Interconexión

❑ Multietapa (IBM SP2)



Topologías de Interconexión

□ Hipercono

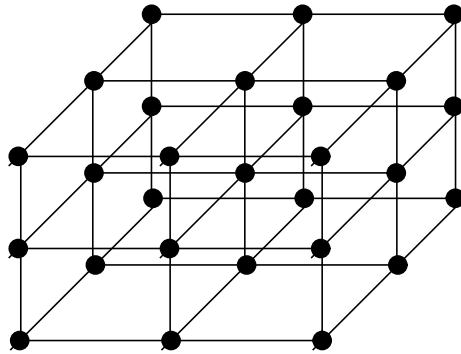


✓ Muy populares hace 10 años.

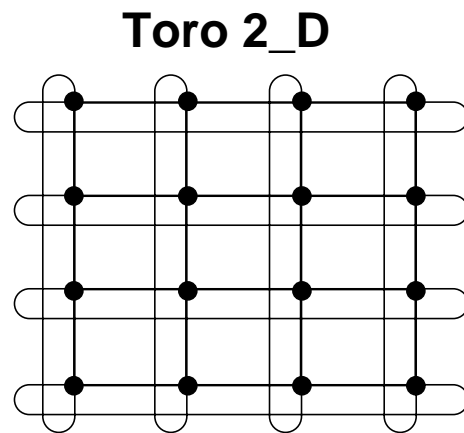
✓ Escalabilidad limitada. No se consideran apropiadas en la actualidad para un elevado número de procesadores.

Topologías de Interconexión

□ Mallas y toros de K dimensiones



Malla 3_D

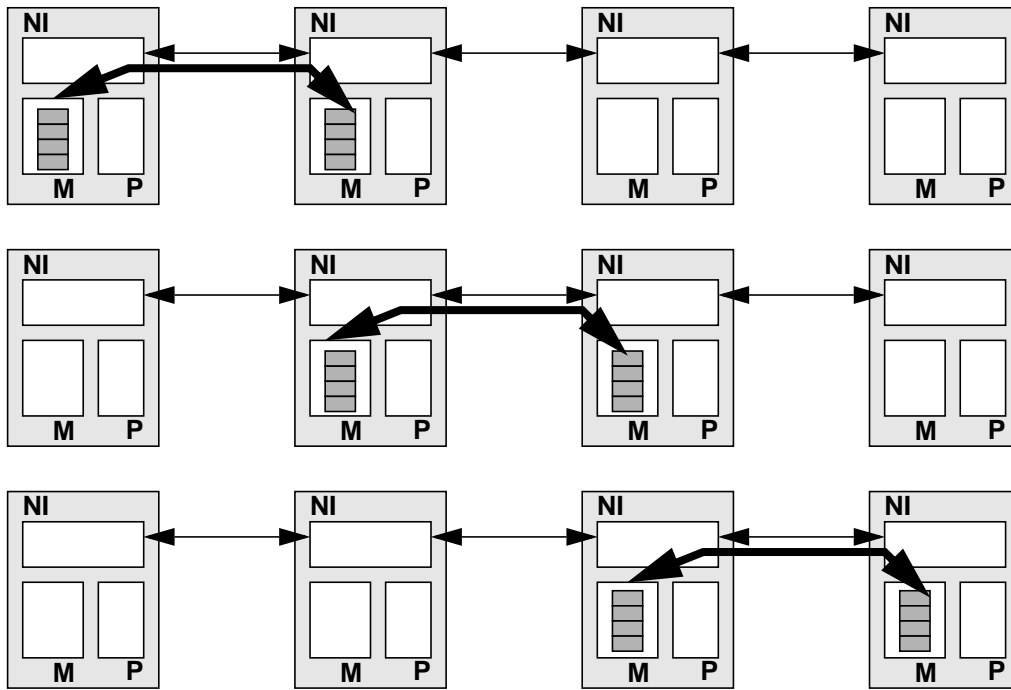


✓ Son las topologías preferidas en la actualidad para un elevado número de procesadores.

Conmutación y Control de Flujo

❑ Store-and-Forward

Cuando un mensaje llega a un nodo/switch intermedio, es almacenado completamente en un buffer.

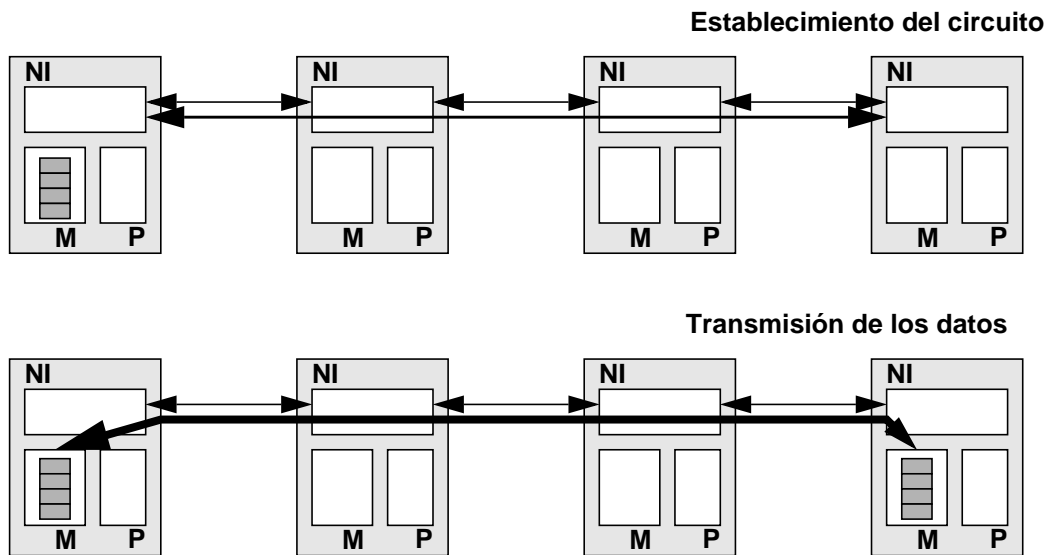


✓ Usado en los primeros hipercubos

Conmutación y Control de Flujo

❑ Circuit switching

La cabecera del mensaje establece un circuito físico que es reservado exclusivamente para la transmisión del resto del mensaje.

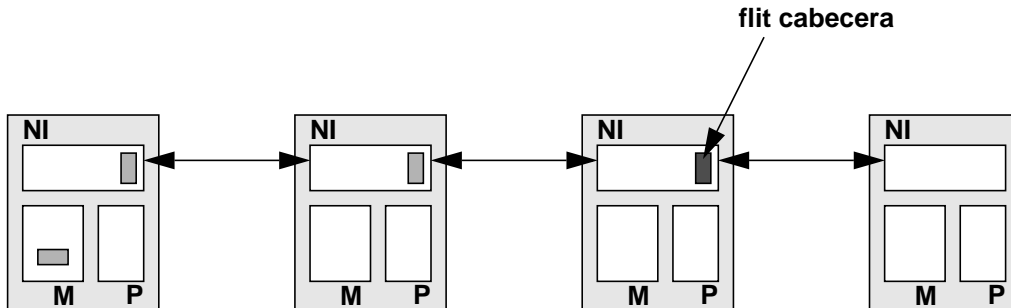


✓ Se reduce la latencia de la comunicación (independiente de la distancia)

Conmutación y Control de Flujo

❑ Wormhole

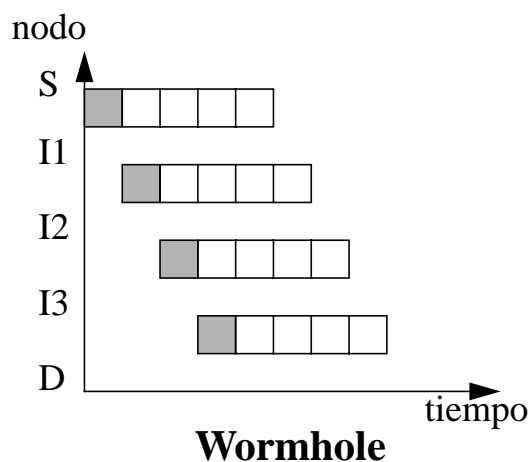
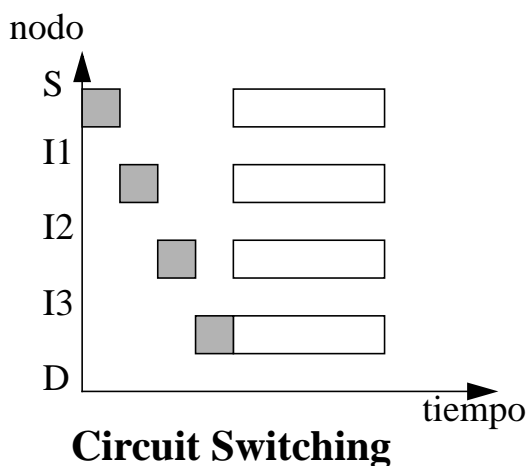
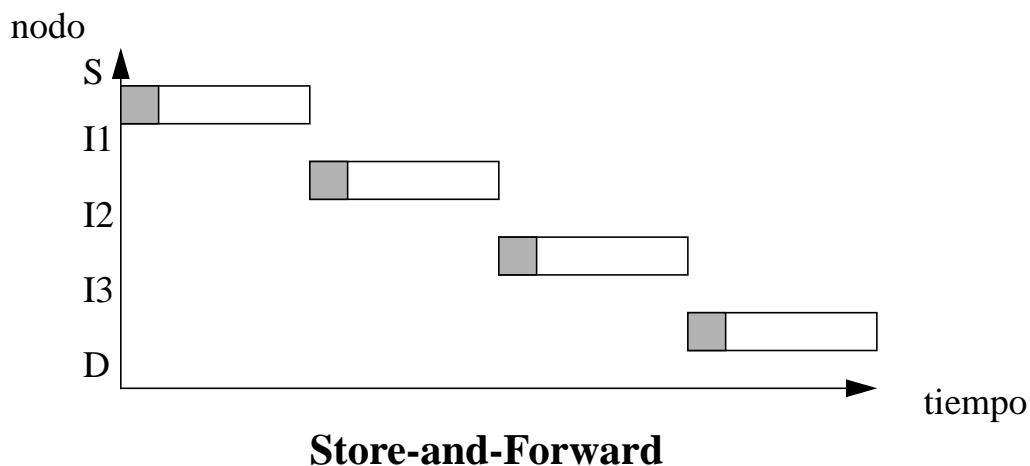
El mensaje se descompone en piezas pequeñas (flits), que avanzan por el camino de manera segmentada. Cuando el flit cabecera se bloquea, el resto de los paquetes se bloquean también a lo largo del camino establecido.



- ✓ La latencia también es independiente de la distancia pero permite una mejor ocupación de los enlaces.
- ✓ Es el más usado en la actualidad

Conmutación y Control de Flujo

Comparación



✓ Cuando hay comunicación intensa, los beneficios de circuit switching y wormhole sobre store-and-forward no son tan importantes.

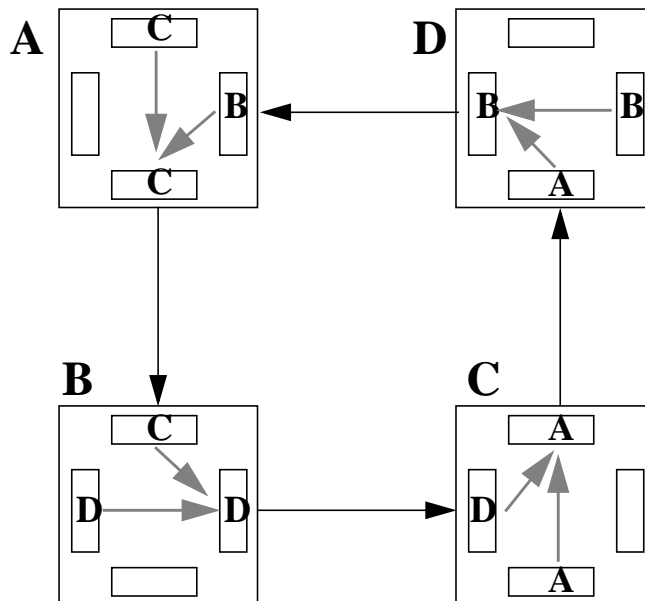
Algoritmo de Encaminamiento

❑ Encaminamiento Determinista

Sólo existe un posible camino entre cada pareja de nodos. Este camino está completamente determinado por las direcciones de los nodos fuente y destino.

✓ Es rápido y barato.

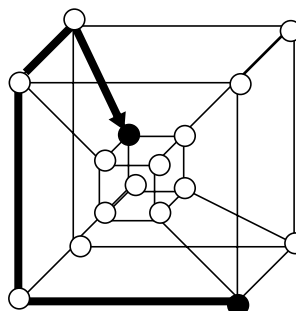
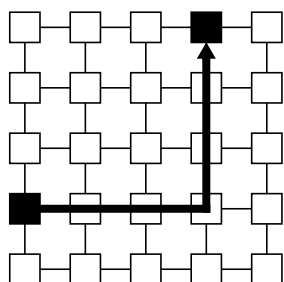
✓ Permite soluciones fáciles al problema del abrazo mortal.



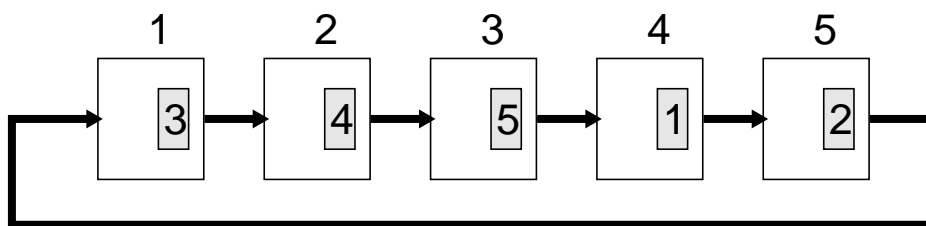
Algoritmo de Encaminamiento

❑ Encaminamiento Determinista

Dimension ordering routing (e-routing)



✓ Evita abrazo mortal en mallas e hipercubos, pero no en toros

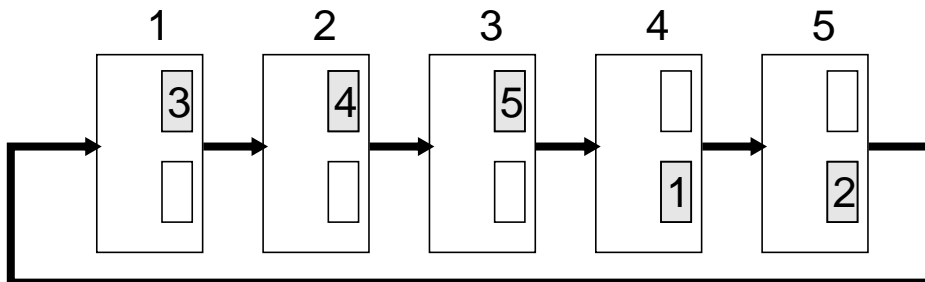
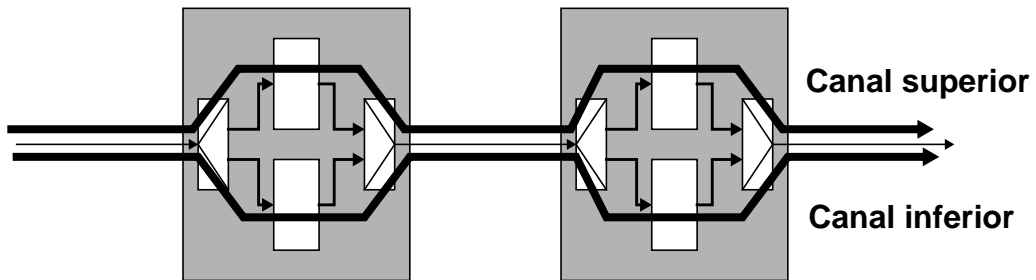


Situación de abrazo mortal en un anillo

Algoritmo de Encaminamiento

❑ Encaminamiento Determinista

Canales virtuales



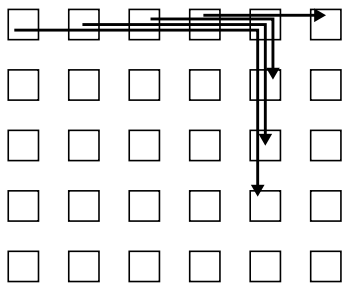
si destino > fuente => usar canal superior
sino usar canal inferior

Algoritmo de Encaminamiento

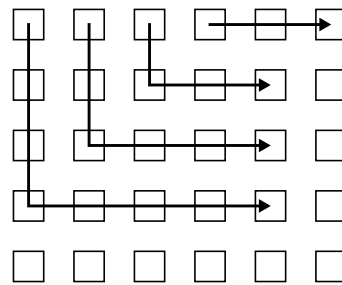
❑ Encaminamiento Adaptativo

El camino se establece de forma dinámica, durante el viaje del mensaje, en función de situaciones de congestión, fallos, etc.

- ✓ Es más lento y más caro.
- ✓ El problema del abrazo mortal es más difícil de resolver.
- ✓ Permite distribuir mejor la carga.



Congestión con
encaminamiento determinista
dimension ordering



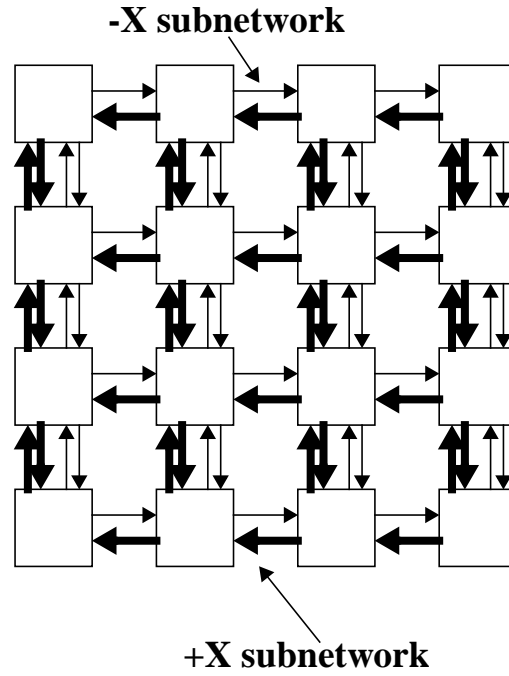
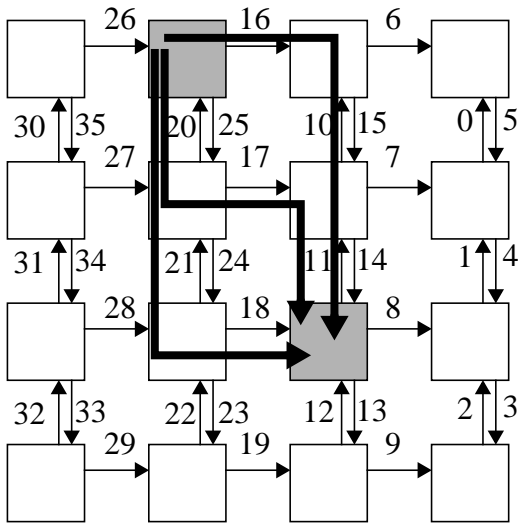
Uso de caminos alternativos

Algoritmo de Encaminamiento

❑ Encaminamiento Adaptativo

Ejemplo de encaminamiento adaptativo

si destino a la derecha usa +X
si destino a la izquierda usa -X
usa canales en orden descendente



cualquiera de los tres caminos
mínimos es posible

✓ Necesita canales virtuales

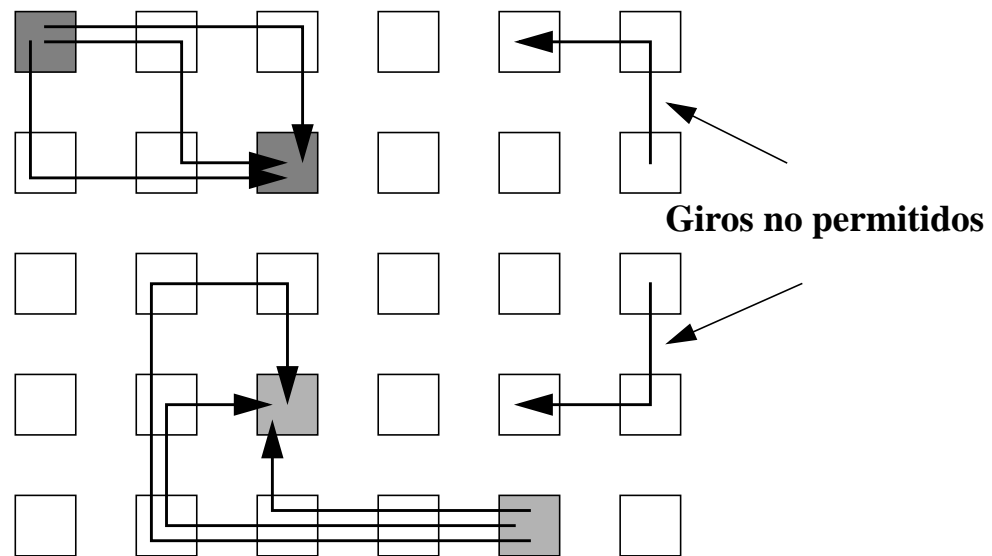
✓ Es completamente adaptativo (todos los caminos son posibles)



Algoritmo de Encaminamiento

❑ Encaminamiento Adaptativo

Encaminamiento adaptativo west-first

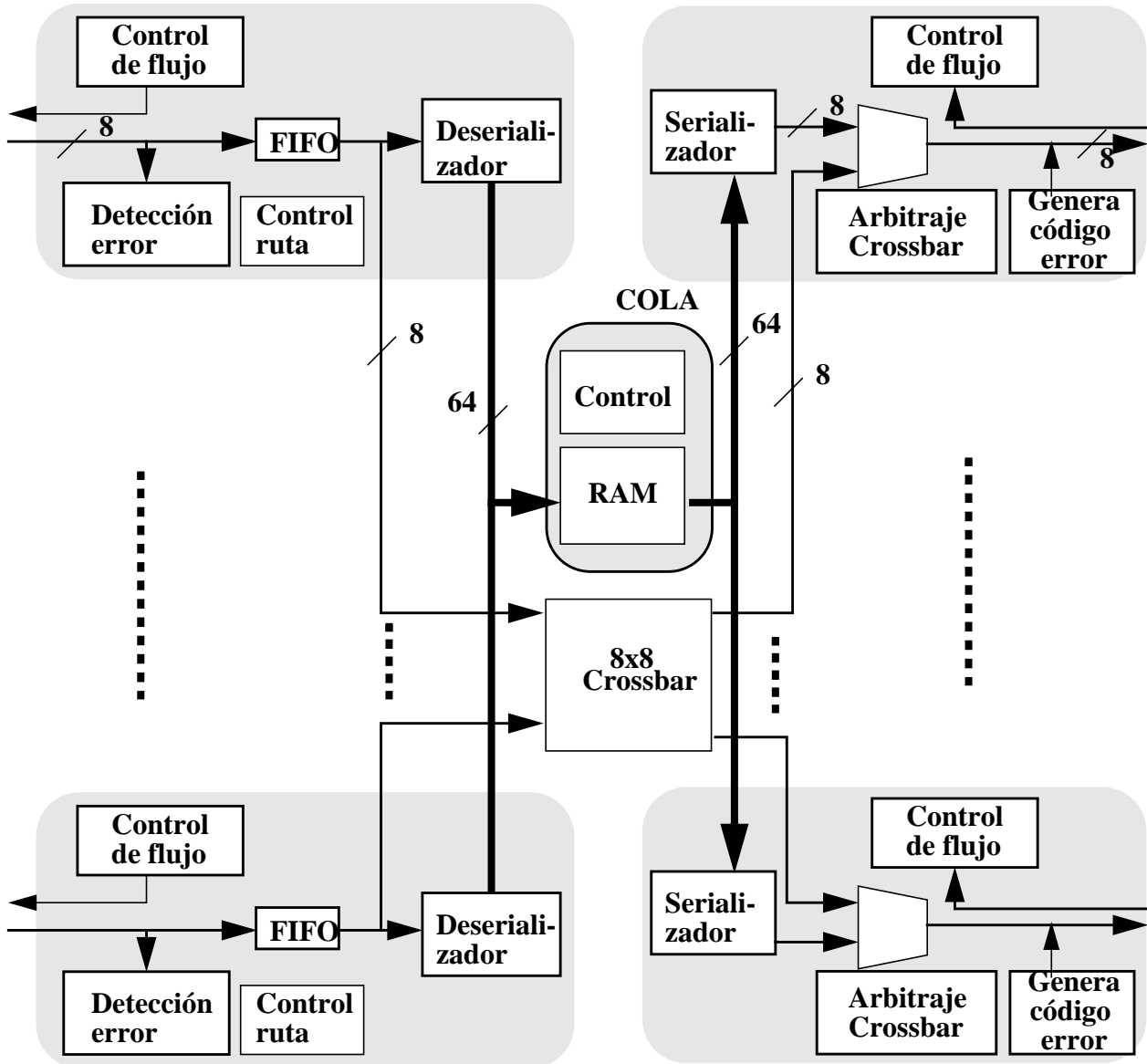


✓ No requiere canales virtuales

✓ Adaptatividad limitada

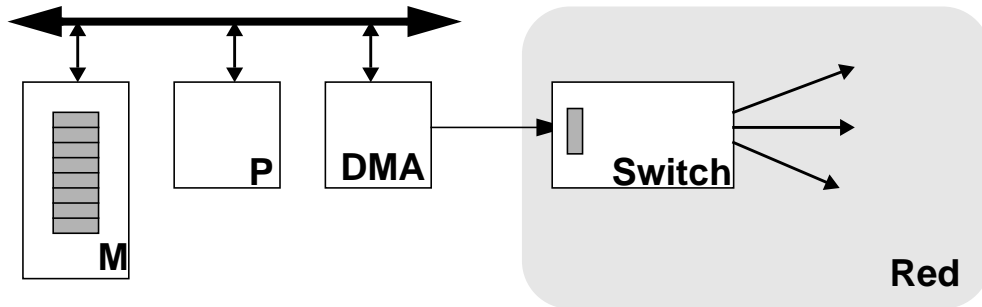
Organización del Switch

□ Ejemplo: El switch del IBM SP2



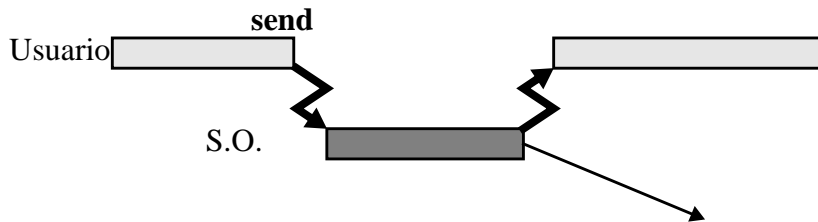
Interfaz Procesador-Red

□ Interfaz clásica

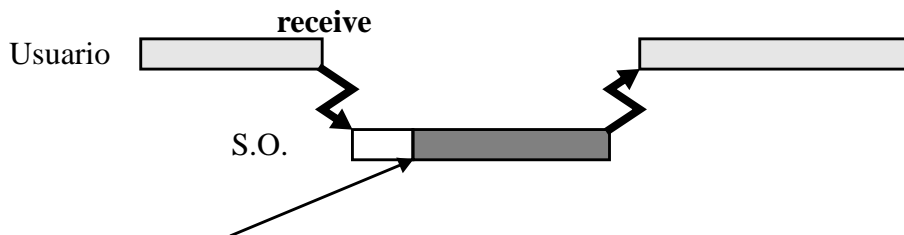


Servicios del S.O.

send (datos, destino)



receive (datos, fuente)



Interfaz Procesador-Red

❑ Problema de la Interfaz clásica

- ✓ Muy alto overhead software

❑ Direcciones de mejora

- ✓ Nuevos modelos de funcionamiento para los servicios del S.O.
- ✓ Incorporar mecanismos hardware en los procesadores
- ✓ Ofrecer mecanismos a nivel de usuario, eliminando la necesidad de “pasar” por el S.O.

Interfaz Procesador-Red

□ Nuevos modelos para los servicios del S.O.

Ejemplo: Mensajes Activos

La cabecera del mensaje indica la rutina que hay que ejecutar en el nodo destino al recibirse el mensaje.

Primitivas

put (nodod, dird, L, flag, datos)

Se envia el mensaje *datos*, de tamaño *L*, al nodo *nodod*. A la recepción del mensaje, el nodo *nodod* copia el mensaje a partir de la posición de memoria *dird* y activa *flag*.

get (nodof, dirf, L, flag, nodod, dird)

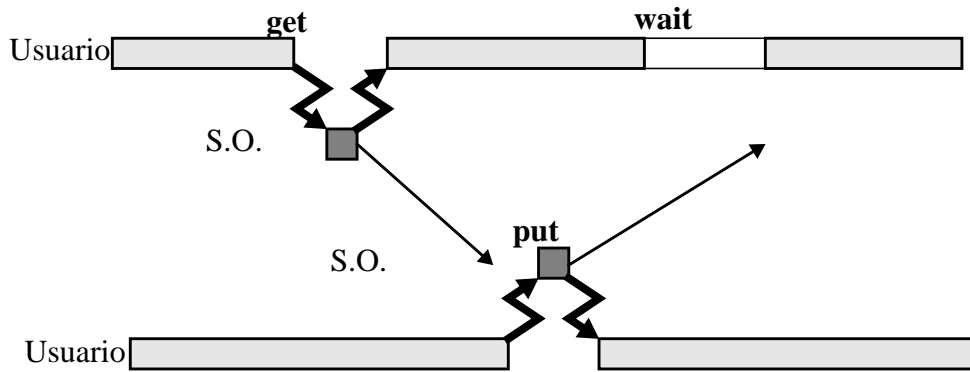
Mediante esta primitiva, *nodod* solicita a *nodof* una copia de los datos almacenados a partir de *dirf* y tamaño *L*. Esta copia debe almacenarse a partir de *dird* de *nodod*. A la recepción de la petición, *nodof* contesta con: *put (nodod, dird, L, flag, datos)*



Interfaz Procesador-Red

□ Nuevos modelos para los servicios del S.O.

Ejemplo: Mensajes Activos



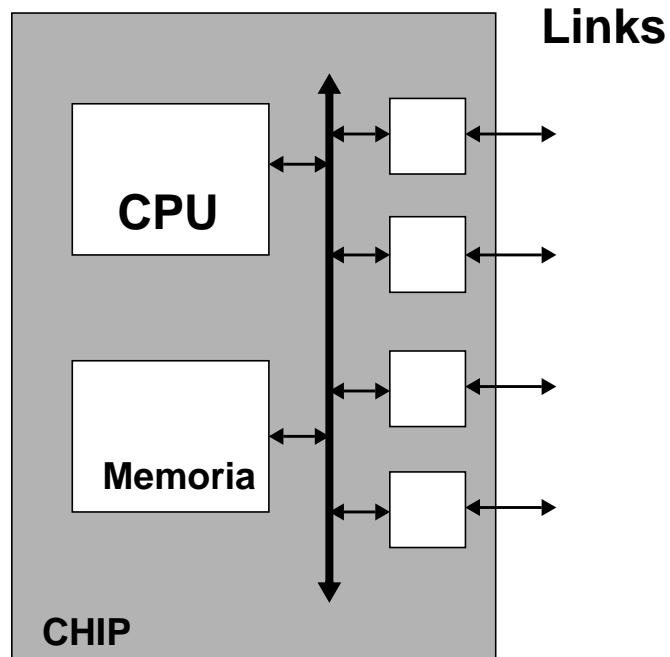
✓ Overhead pequeño

Interfaz Procesador-Red

❑ Incorporar mecanismos hardware en los procesadores

- ✓ Integración en un solo chip
- ✓ Instrucciones de lenguaje máquina
- ✓ Mecanismos de scheduling orientados a mensajes

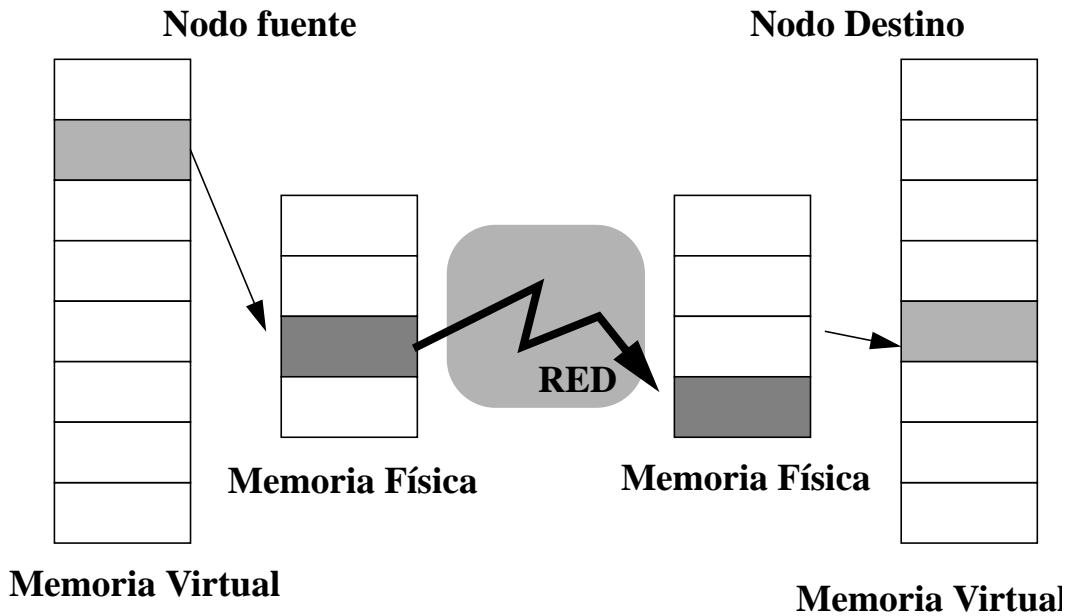
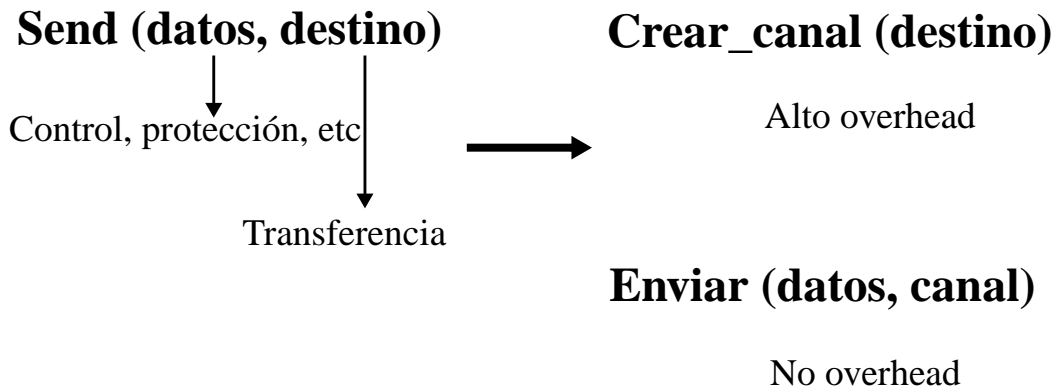
Ejemplo: Transputer



Interfaz Procesador-Red

❑ Ofrecer mecanismos a nivel de usuario

Ejemplo: SHRIMP, Memoria Virtual Mapeada

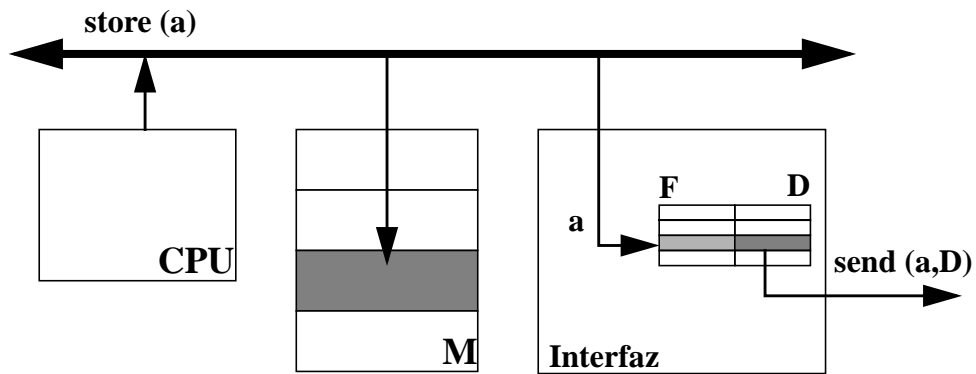


Interfaz Procesador-Red

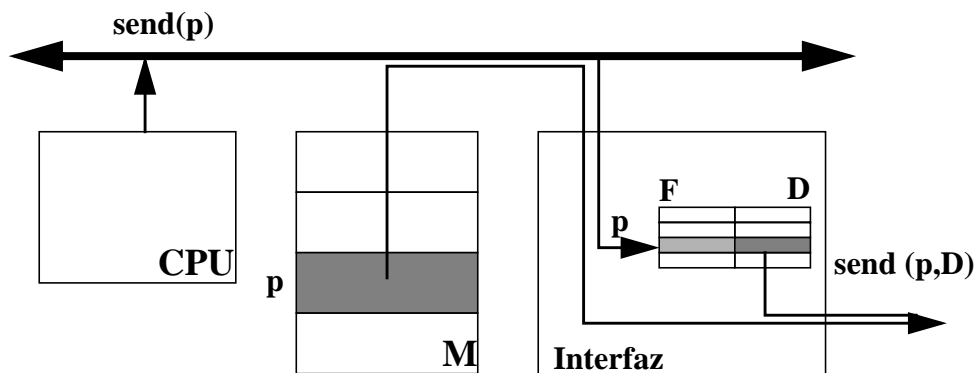
❑ Ofrecer mecanismos a nivel de usuario

Ejemplo: SHRIMP, Memoria Virtual Mapeada

Automatic Update

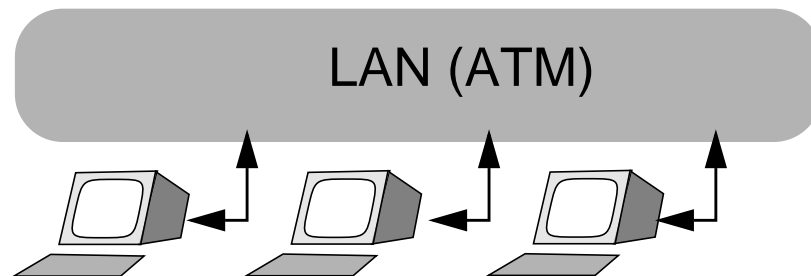


Deliverate Update



Networks of Workstations (NOW)

□ Idea Básica



- ✓ Compartición de recursos (disco, memoria)
- ✓ Plataforma paralela
- ✓ Programación en paso de mensajes