Arquitectura de Computadores

TEMA 5

Multiprocesadores y redes de interconexión

DEPARTAMENTO DE
ARQUITECTURA DE COMPUTADORES
Y AUTOMÁTICA

Curso 2014-2015

Contenidos

- Introducción
- Arquitecturas paralelas
  - Memoria compartida
  - Paso de mensajes
- Redes de interconexión
  - Descripción estructural
  - Topología de Red
    - Redes estáticas
    - Redes dinámicas
  - Técnicas de conmutación
  - Control de flujo
  - Encaminamiento (routing)
- Bibliografía
  - Apéndice F de [HePa12]
  - Cap 1 y 10 de [CuSi99]

Nota.- En la elaboración de este material se han utilizado contenidos desarrollados por los profesores Manuel Prieto, Luis Piñuel e Ignacio Martín. También se han utilizado figuras de [HePa12] y [CuSi99].

Introducción

- Una definición clásica de computador paralelo
  - Colección de elementos de procesamiento que cooperan y se comunican para resolver problemas grandes (cálculo, almacenamiento) "rápidamente". (Almansi and Gottlieb 1989)

- Objetivo
  - Rendimiento / Eficiencia
  - Replicación (Tolerancia a fallos)

- Esta definición da lugar a muchas preguntas:
  - ¿Qué potencia tienen los elementos de proceso?
  - ¿Cuánta memoria?
  - ¿El número puede crecer de una manera sencilla: es escalable el sistema?
  - ¿Cómo se comunican y cooperan dichos elementos?
  - ¿Cómo se transmiten los datos entre los procesadores?
  - ¿Qué tipo de interconexión conecta los distintos procesadores?
  - ¿Qué abstracción (primitivas) proporciona el hardware /software al programador?

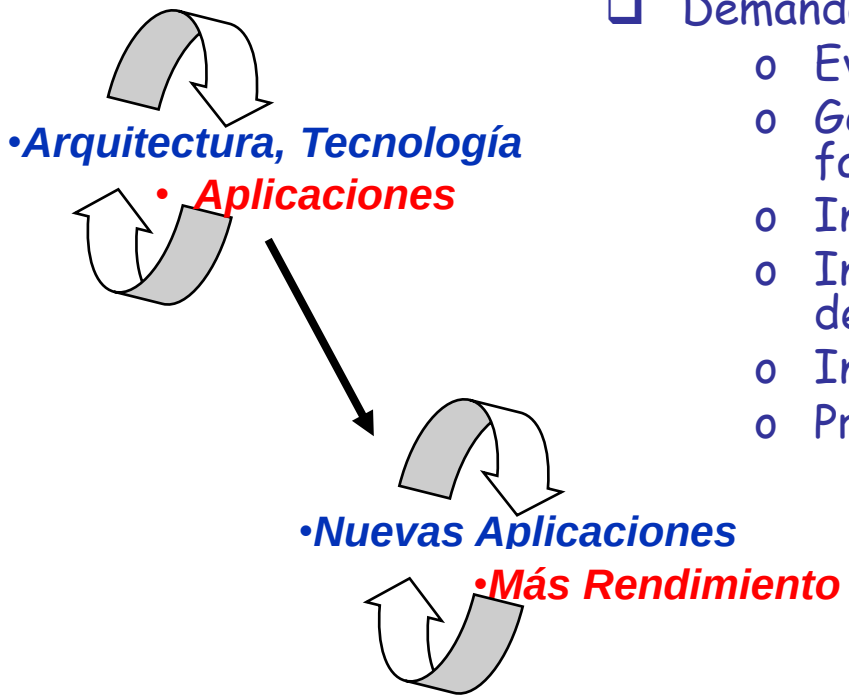
Introducción

- Creciente interés por los multiprocesadores

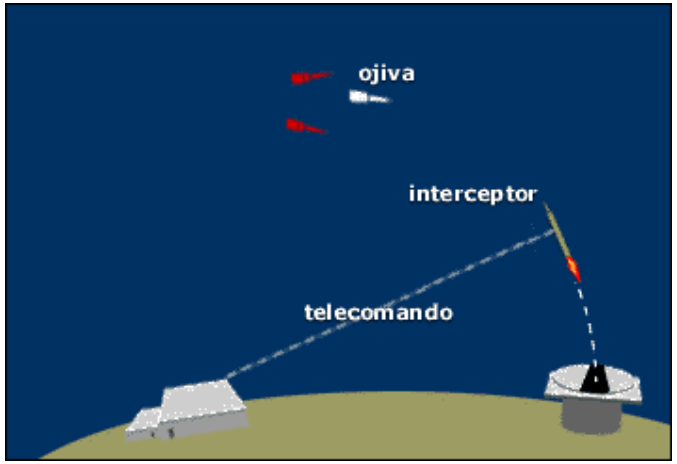
> *"We are dedicating all of our future product development to multicore designs. We believe this is a key inflection point for the industry."*
>
> Intel President Paul Otellini,
> describing Intel's future direction at the
> Intel Developer Forum in 2005

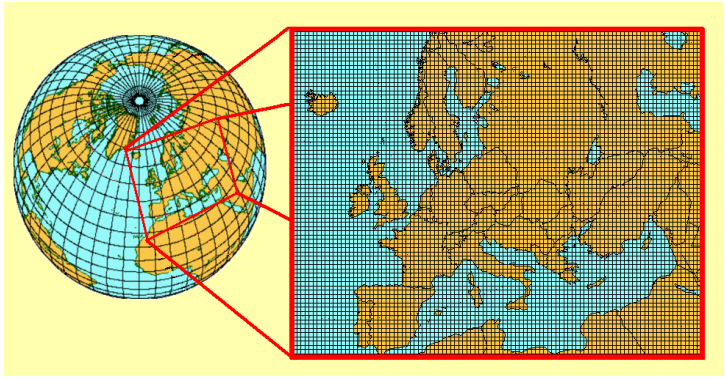
- Algunos factores determinantes
  - Agotamiento de ILP
    - Consumo de energía y coste creen más rápido que rendimiento
  - Creciente interés en servidores de altas prestaciones
    - Cloud computing, Software-as-a-Service (SaaS)
  - Aumento de las aplicaciones intensivas en datos
    - Geociencia, Biología computacional, ...
    - Disponibilidad de enormes cantidades de datos vía Internet
  - Mejora del conocimiento sobre cómo usar eficazmente los multiprocesadores
    - Entornos de grandes BD, multitud de accesos simultáneos, problemas de cálculo científico
  - Aprovechamiento de los diseños vía replicación del procesador

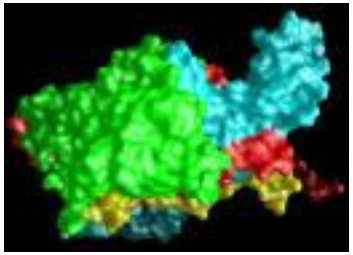
Aplicaciones

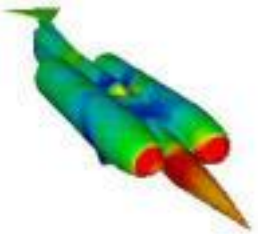


- Demanda: Aplicaciones "Grand-Challange"
  - Evolución climática-Predicción meteorológica
  - Genómica y Proteómica (Industria farmacéutica)
  - Industrias Aeronáutica y Automovilística
  - Industria Militar (Proyecto ASCI departamento defensa US)
  - Industria Entretenimiento
  - Procesamiento Transaccional









Aplicaciones: Ejemplo

- Evolución climática-Predicción meteorológica.

- Necesidades de memoria y cálculo
  - El clima es una función de la longitud , latitud, altura, tiempo
  - Para cada uno de estos puntos se debe calcular temperatura, presión, humedad, y velocidad del viento (3 componentes)
  - Conocido clima(i,j,k,t), el simulador debe proporcionar el valor clima(i,j,k,t+Δt)

    - Predicción 1 Minuto

      Celdas de 1km x1km y 10 celdas en altura

      → $5x10^9$ celdas (0.1 TB)

      Δt de un minuto: 100 flop por celda (estimación muy optimista)

      → $100x5x10^9$ operaciones en menos de un minuto: 8 GFlops

    - Predicción del tiempo a 7dias en 24 horas → 56 Gflops

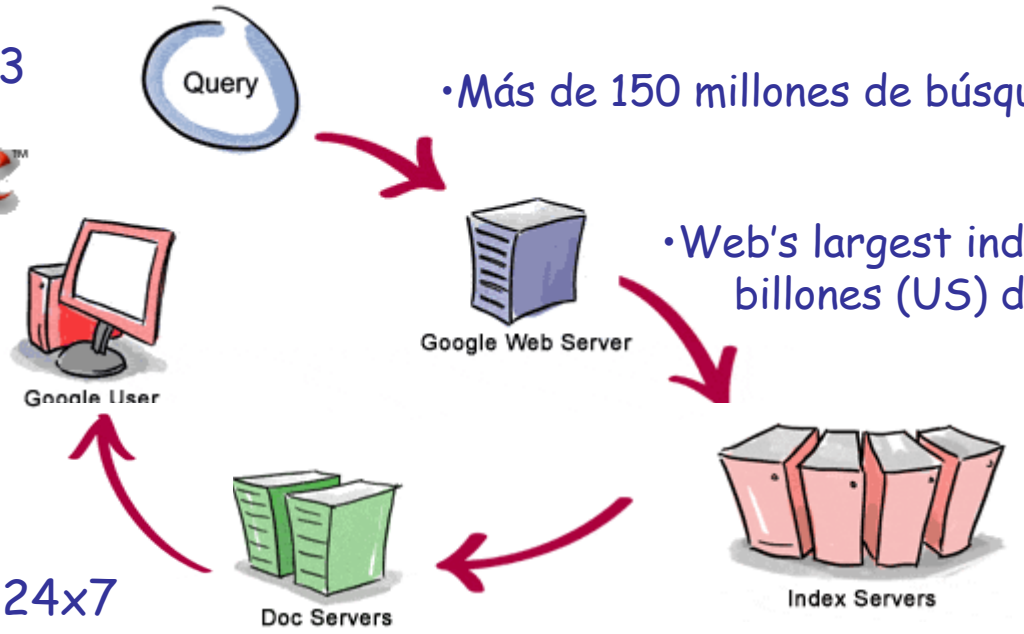
    - Predicción de clima a 50 años en 30 días → 4.8 Tflops

Aplicaciones: Ejemplo

- Búsquedas en Internet

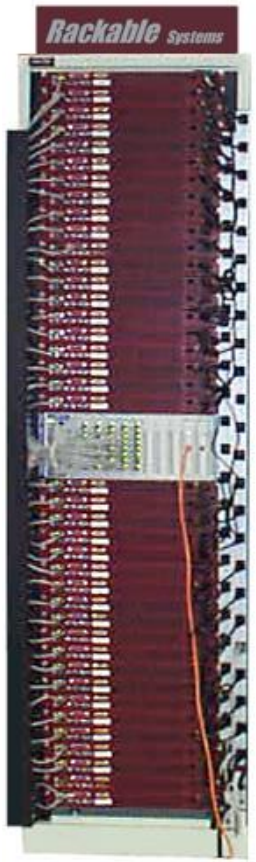
Febrero 2003

Google



- Más de 150 millones de búsquedas/día
- Web's largest index: Más de 3 billones (US) de páginas
- World's largest commercial **Linux cluster**: Más de 10,000 servidores

Availability: 24x7



Arquitecturas Paralelas

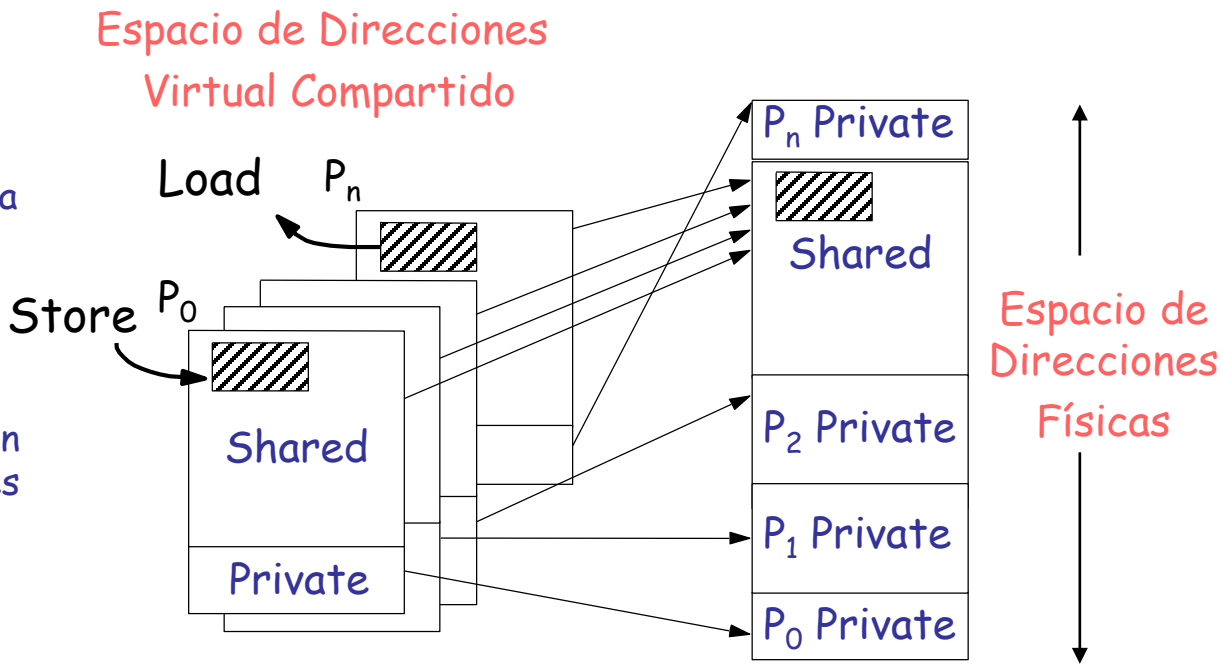
- Modelo de Programación: concepto de máquina que utiliza el programador
  - ¿Cómo se comparte la información? ¿Cómo transferir la información entre distintas partes de un programa que se ejecuta en paralelo?
  - ¿Cómo se lleva a cabo la coordinación de actividades? ¿Qué primitivas de sincronización hay para coordinar actividades?
  - Típicamente: el modelo de programación esta incorporado a
    - Lenguaje de Programación (compilación)
    - Librería

    Son los encargados de realizar la correspondencia
    - Construcciones del lenguaje o llamadas de la librería ⇔ Primitivas disponibles a nivel usuario (Arquitectura de Comunicaciones)
    - Algunas operaciones las realiza una combinación Hw/SO
- Memoria compartida vs. paso de mensajes

- Un espacio de direcciones compartido
  - Referente: multiprogramación en monoprocesadores
    - Objetivo: Productividad (entornos time-sharing)
    - Los procesos pueden configurarse de modo que parte de su espacio de direcciones sea compartido:
      - Comunicaciones: implícitas (operaciones LOAD/STORE)
      - Sincronizaciones: SO, Operaciones Atómicas

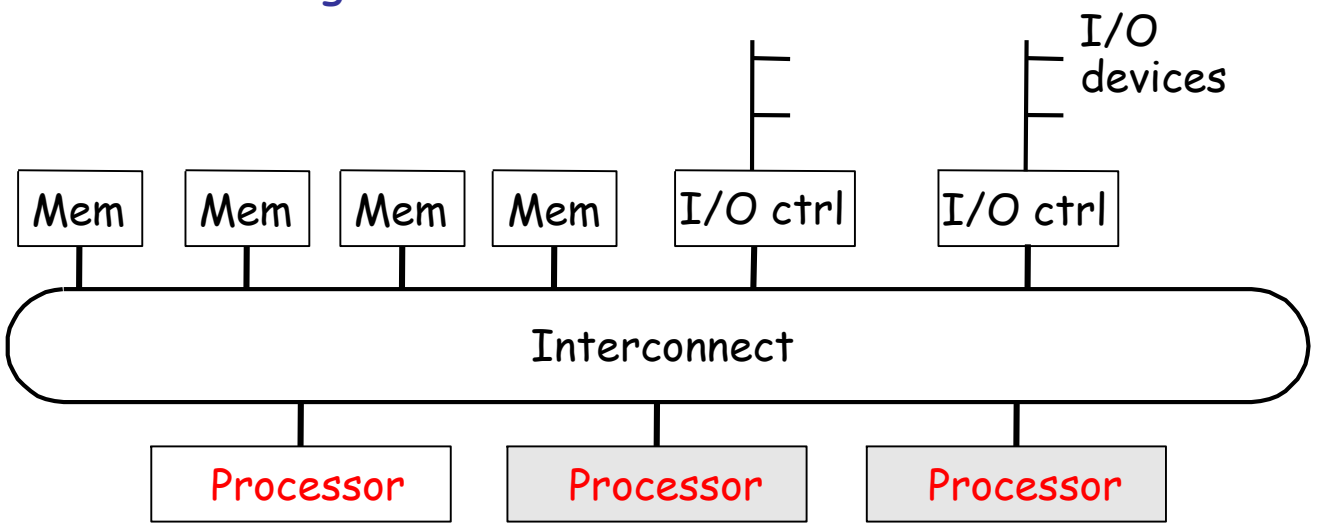
Múltiples procesos pueden tener una región de memoria compartida proyectada sobre su espacio de direcciones virtual

Las escrituras de variables residentes en esta región son visibles para el resto de los procesos.



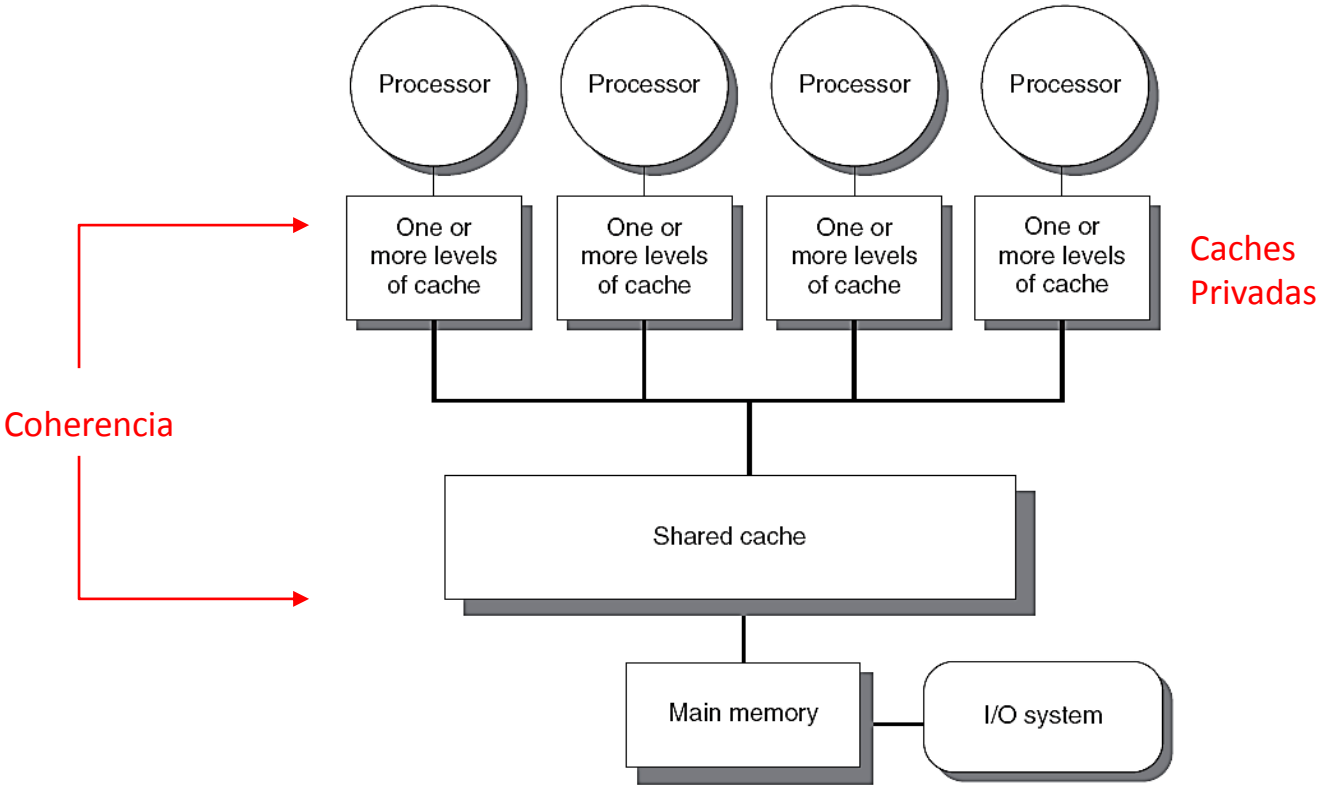
Arquitecturas de Memoria Compartida

- Multiprocesador de memoria compartida:
  - Extensión natural de un sistema en el que diferentes procesos pueden compartir memoria ➔ añadir uno o varios procesadores
  - Red de interconexión vs. Nº de procesadores
    - Pocos procesadores: red sencilla (p. ej. Bus)
    - Muchos procesadores: red más compleja (p. ej. red multietapa)
  - Escalabilidad: ¿Cuántos procesadores se pueden conectar sin que la red se convierta en un cuello de botella que degrade el rendimiento global?



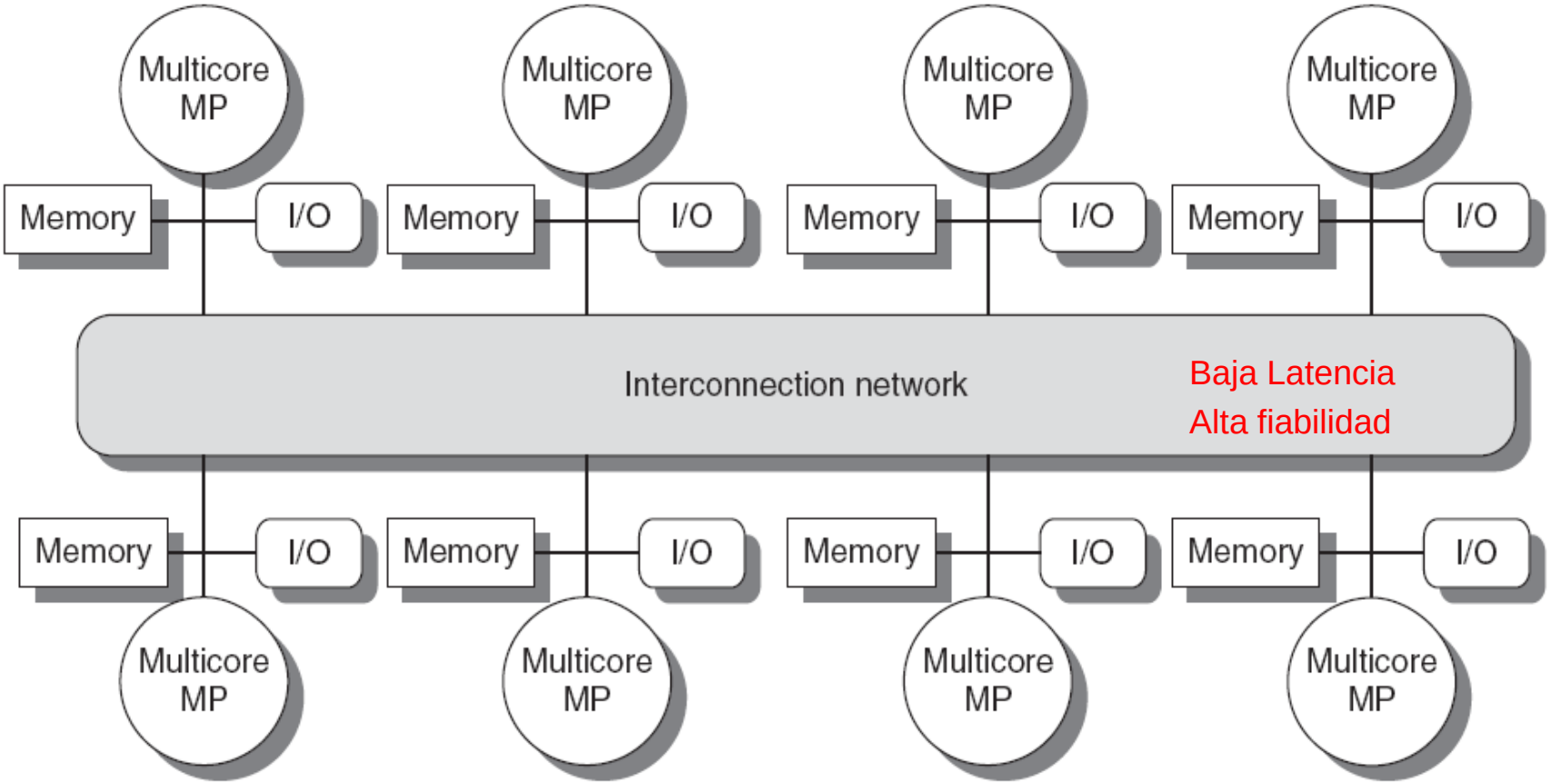
Arquitecturas de Memoria Compartida

- Diseño para un nº reducido de procesadores
  - Symmetric (Shared-Memory) Multiprocessor (SMP)
  - Memoria: centralizada con tiempo de acceso uniforme ("UMA")
    Interconexión vía bus común, E/S compartida
  - Anchura de banda limitada. Limite en el numero máximo de procesadores
  - Caches son cruciales: Coherencia
  - Posibilidad de usar otras redes: crossbar, multietapa
  - Ejemplos: Sun Enterprise 6000, SGI Challenge, Intel SystemPro, ……



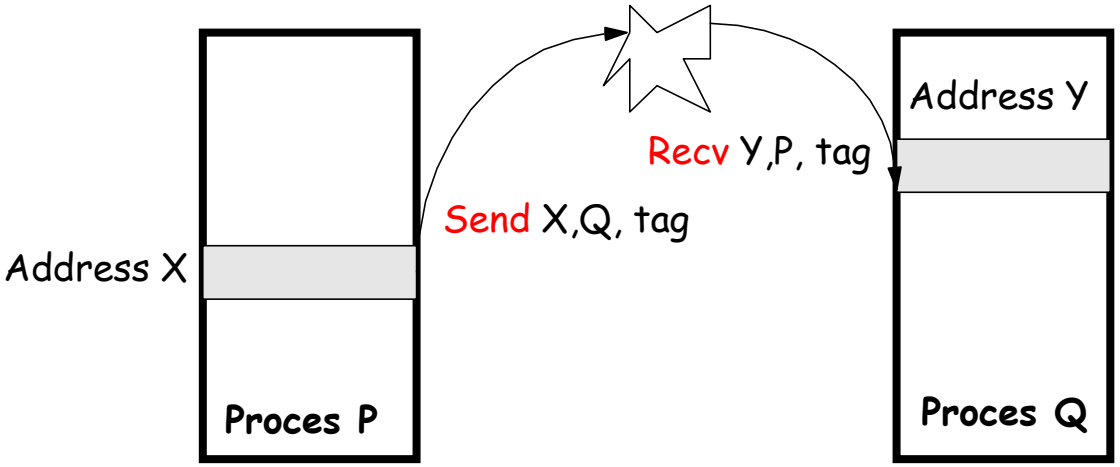
Arquitecturas de Memoria Compartida

- Diseño para un número elevado de procesadores
  - Memoria: Distribuida con tiempo de acceso no uniforme ("NUMA")
  - Interconexión escalable.
  - Anchura de banda crece con el numero de EP
  - Ejemplos: Cray T3E



Arquitecturas de Paso de Mensajes

- Elemento de Proceso: Computador Completo (microproces, mem, E/S)
  - Modelo de Programación
    - Acceso directo al espacio de direcciones privado (memoria local)
    - Comunicación explícita mediante operaciones I/O (llamadas al SO: send/recv)



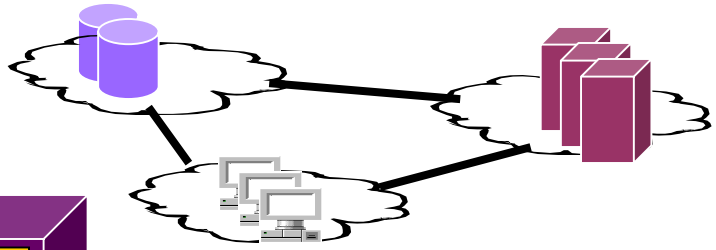
La combinación send-recv establece:

- copia memoria-memoria
- sincronización

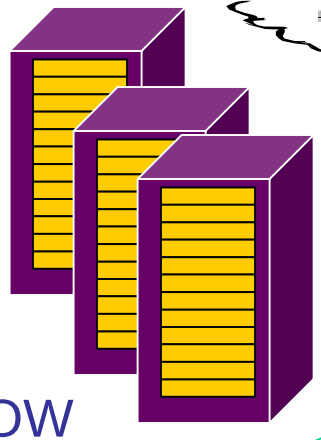
- Sobrecarga:
  - Construir la cabecera del mensaje; copiar los datos en el buffer de la red; enviar el mensaje; copiar los datos en el buffer receptor; copiar los datos **del espacio S.O. al espacio usuario**
- Diagrama de Bloques Semejante a NUMA
  - Diferencia: Las comunicaciones están **integradas en el sistema de E/S** en lugar de en el sistema de memoria
- Semejante a los clusters de PCs

Redes de interconexión: Introducción

Red de interconexión como elemento vertebrador de los sistemas digitales actuales



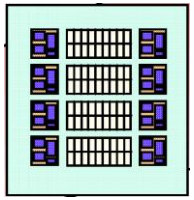
IP Networks
LAN, WAN



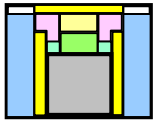
Servidores
Supercomputadores

SAN / NOW
Clusters

PC / WS



Placas

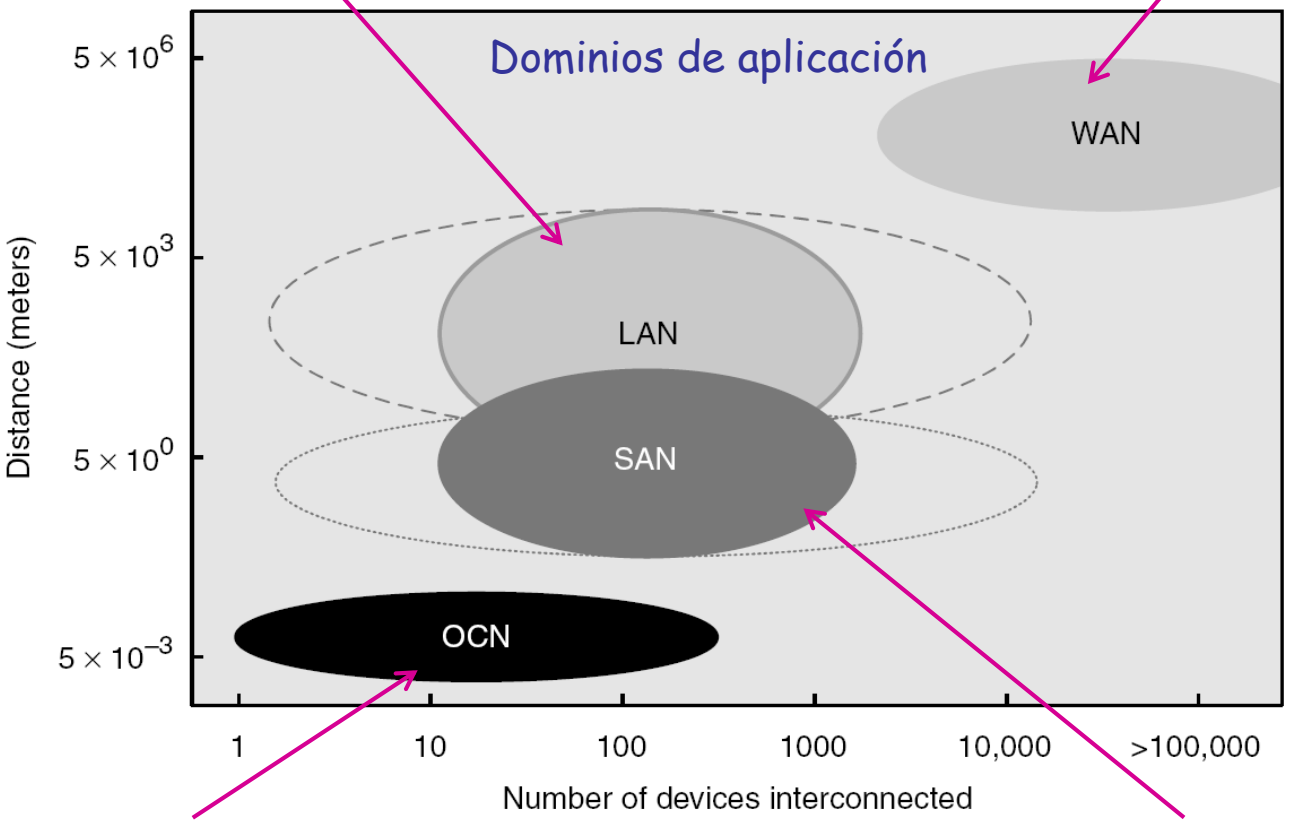


Chips

Diferentes exigencias para el diseño de la red de interconexión

Redes de interconexión: Introducción

- Local Area Network (LAN). Ej: Ethernet (hasta 10 Gbps sobre una distancia de 40 Km). Tb Clusters de PCs
- Wide Area Network (WAN). Conexión a escala global. Ej: ATM



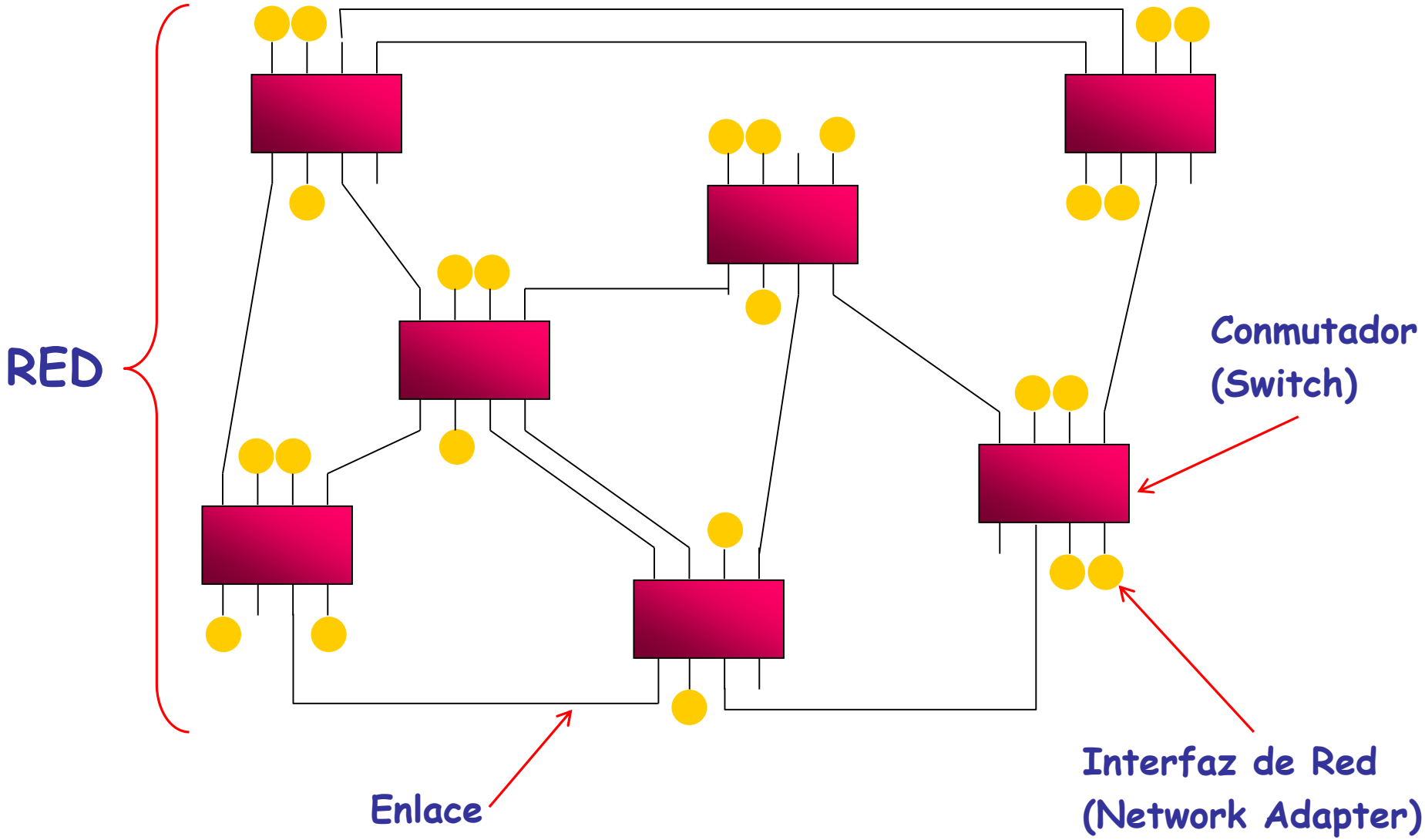
- Network-on-Chip (NoC, OCN). Ej: AMBA de ARM, SCCC de Intel, TILE-GX de Tilera
- System/Storage Area Network (SAN). Comunicación proc-mem-discos. Ej: Infiniband (hasta 200 Gbps sobre una distancia de 300 m)

Redes de interconexión: Introducción

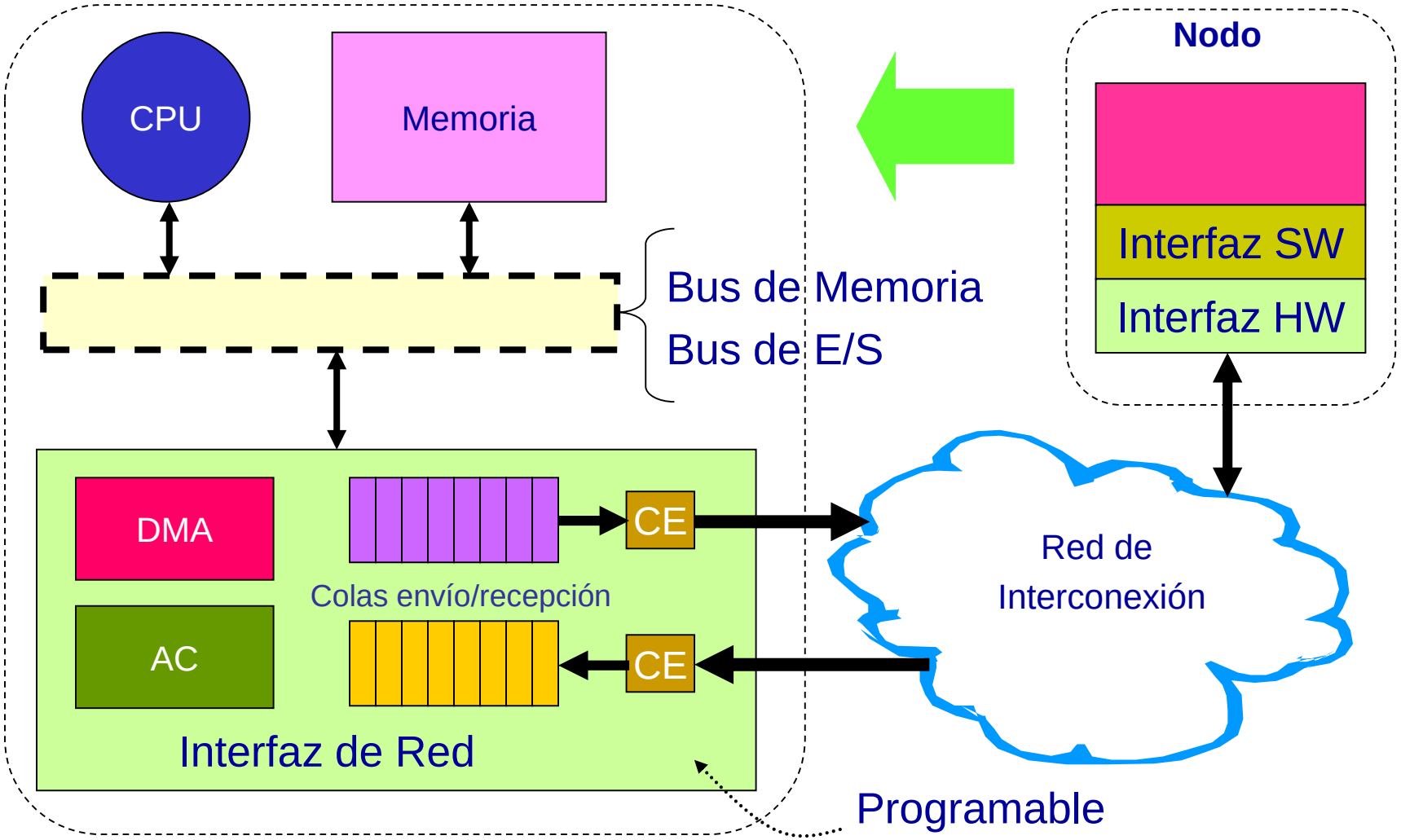
- Factores de diseño: El principal factor en computadores paralelos son las prestaciones de la red:
  - Latencia
  - Productividad

- Otros factores de diseño:
  - Escalabilidad
  - Fiabilidad
  - Expansión incremental
  - Particionado y seguridad
  - Simplicidad y bajo coste
  - Distancia de expansión
  - Consumo/disipación de energía

Descripción estructural: Elementos

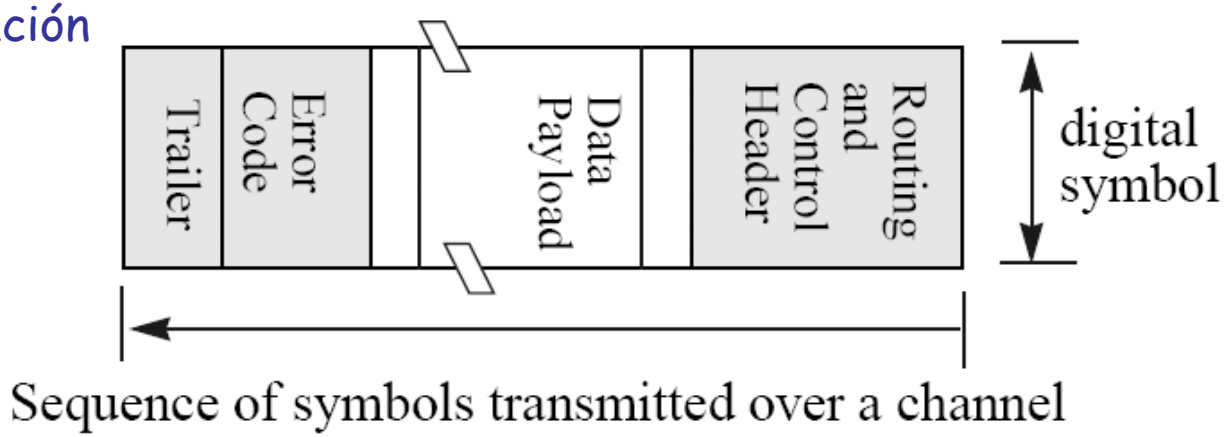


Esquema básico de un interfaz de red

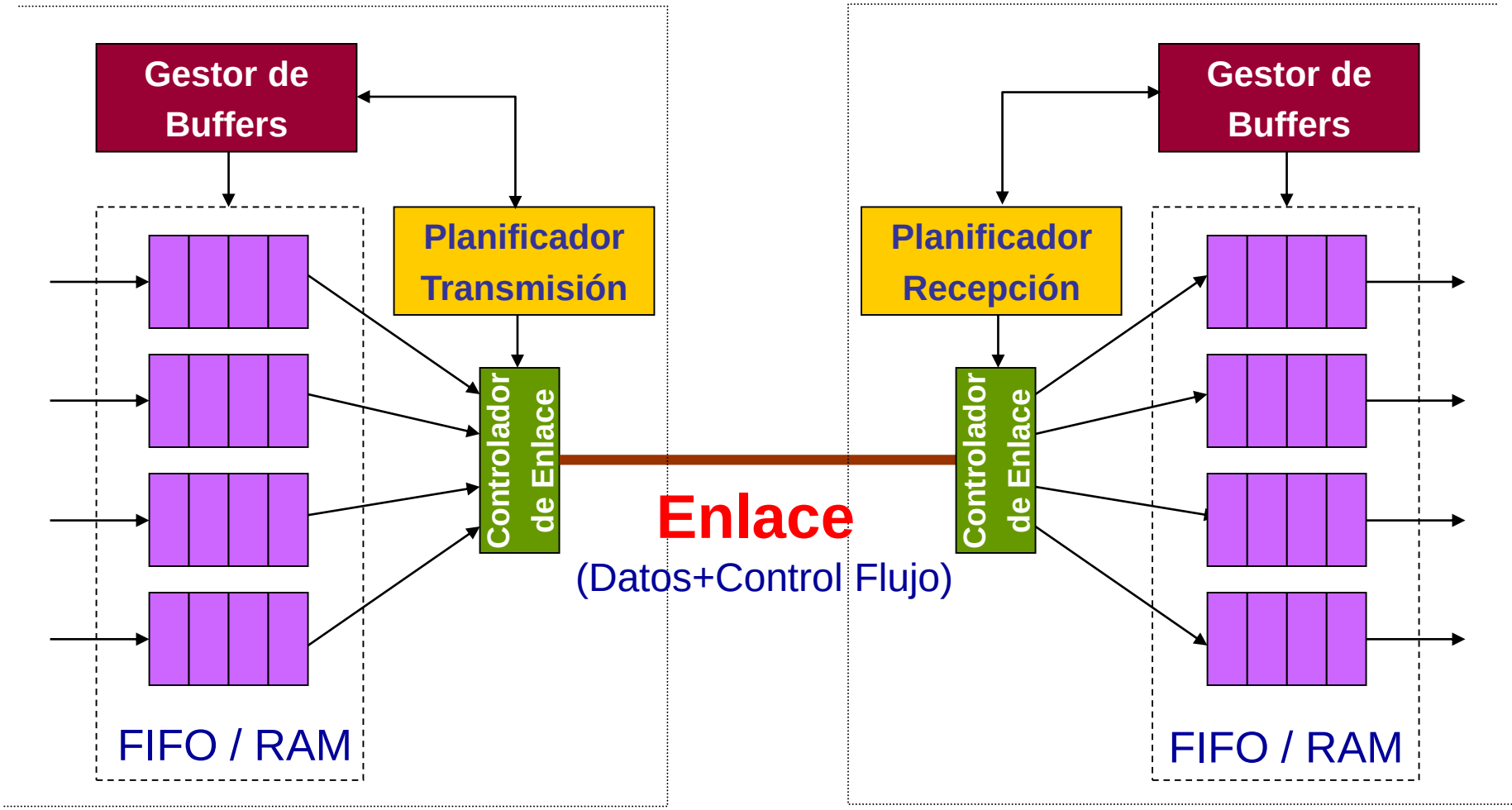


Descripción estructural: Interfaz de Red

- Mensaje
  - Unidad lógica de comunicación
  - Es la única unidad vista a nivel de aplicación.
- Paquete
  - Unidad de transferencia entre interfaces de red (end-to-end)
  - También es la menor unidad que contiene información de encaminamiento.
  - Mecanismos de abstracción:
    - Encapsulado
    - Fragmentación

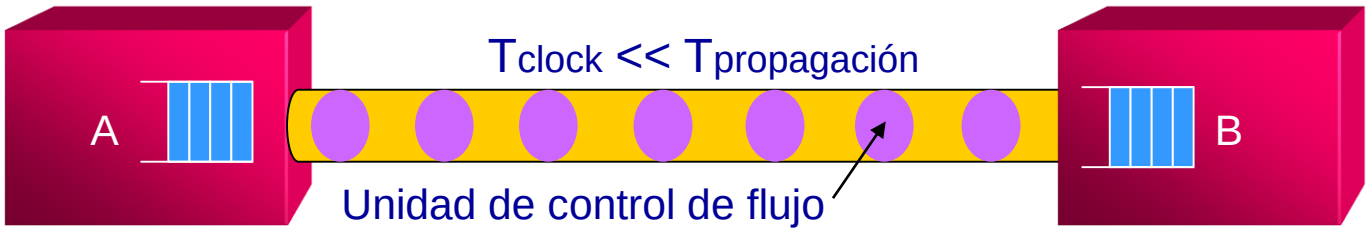


Esquema básico de un enlace punto-a-punto

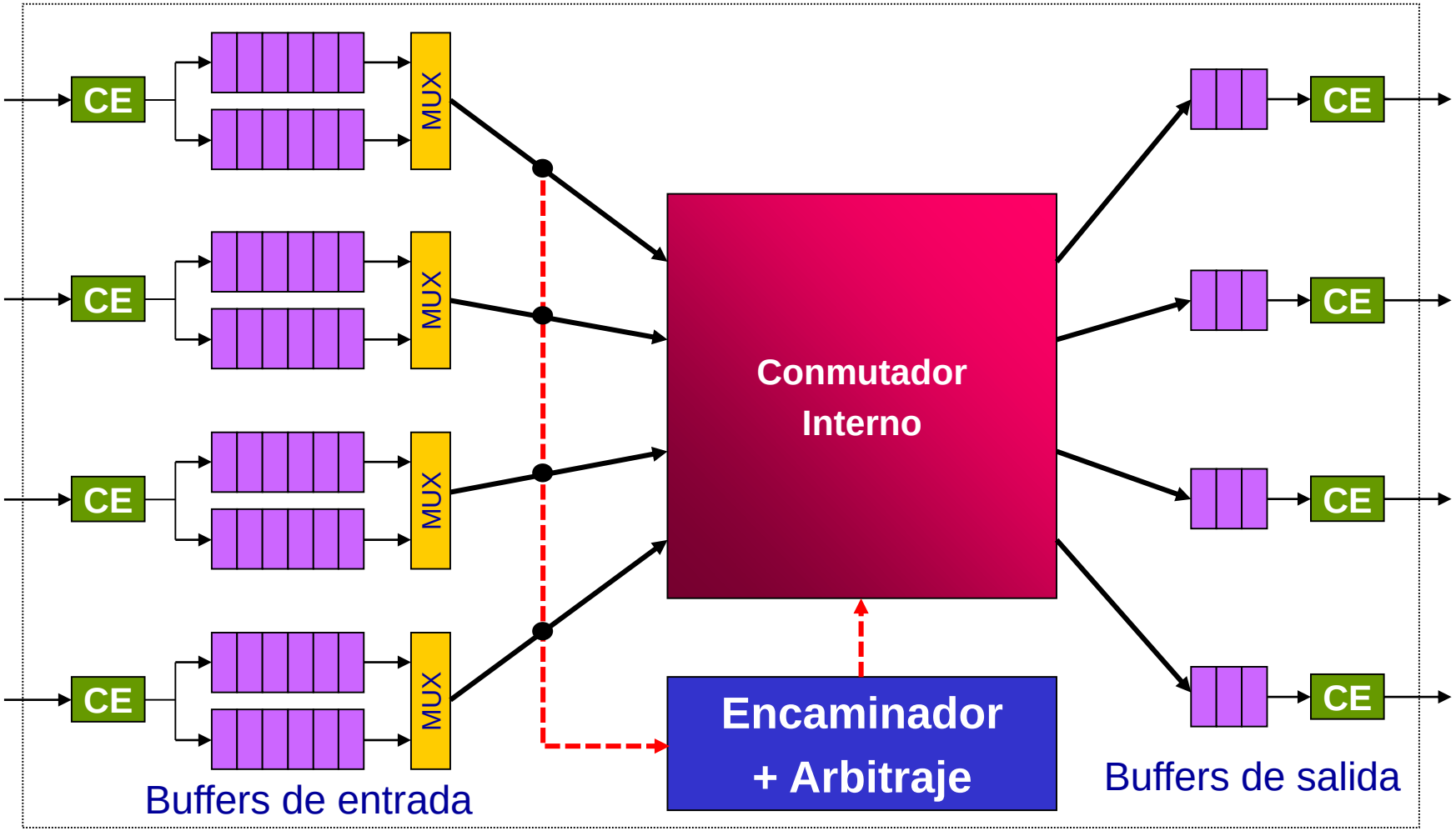


Descripción estructural: Enlace

- Características de un enlace
  - Ancho: **w**
  - Velocidad de señal (frecuencia): **f = 1/τ**
  - Ancho de banda: **B = wf**
  - Unidad de transferencia por ciclo: **Phit (physical unit) = w**
  - Unidad de control de flujo: **Flit (flow-control unit)**
  - "Longitud"
    - Corto: un solo phit en el enlace
    - Largo: varios phit en el enlace (segmentado)



Esquema básico de un conmutador



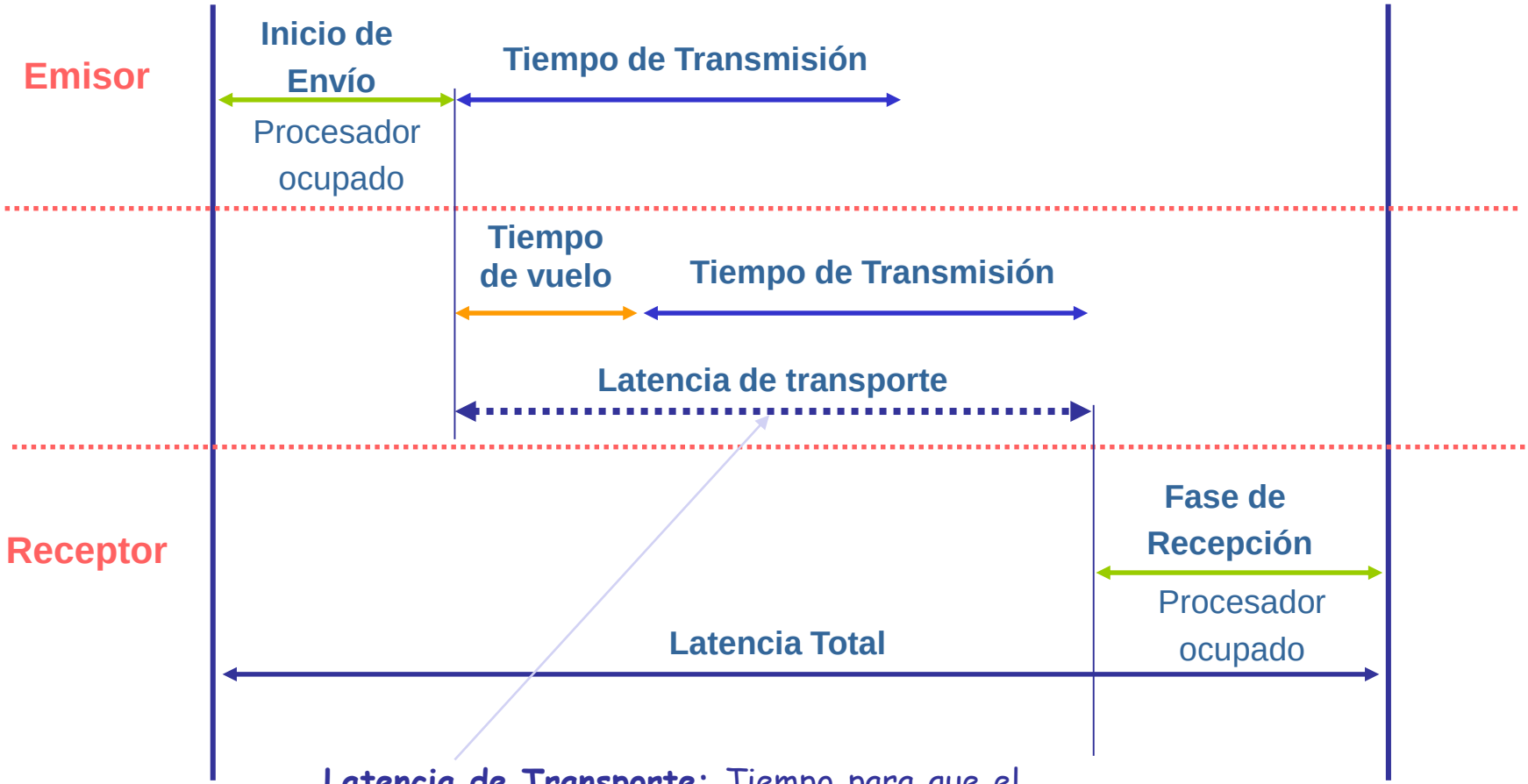
Descripción estructural: Conmutador

- Características de un conmutador
  - Grado conmutador/nodo
    - Número de canales de entrada (salida)
  - Implementación conmutador interno
  - Buffers de entrada/salida
  - Lógica de control:
    - Arbitraje
    - Encaminamiento

Descripción estructural: Red

- Red de interconexión
  - Grafo de V = {conmutadores y nodos} conectada por canales de comunicación $C \subseteq V \times V$
- Ruta
  - La secuencia de conmutadores/nodos y canales seguidos por un mensaje
    - Pensar en calles e intersecciones
- Características de la red
  - Topología
    - Estructura de la interconexión física de la red
      - Definido por su función de interconexión
      - Regular vs irregular
      - Clasificación: medio compartido, estáticas, dinámicas
  - Conmutación
    - Cómo atraviesan su ruta los datos de un mensaje
    - Conmutación de circuitos vs. conmutación de paquetes
  - Control de flujo
    - En qué momento el mensaje, o porciones de éste, se mueven a lo largo de la ruta
  - Encaminamiento
    - Qué rutas pueden seguir los mensajes a través del grafo de la red
      - Encaminamiento determinista vs adaptativo

❑ Latencia total extremo-a-extremo



Latencia de Transporte: Tiempo para que el mensaje pase a través de la red asumiendo que **no hay contención** y **sin incluir las sobrecargas** de inyectar el mensaje en la red ni de salir cuando llega al destino.

Topología de red

- Definiciones y propiedades
  - Red de interconexión
    - Grafo de V = {conmutadores y nodos} conectada por canales de comunicación $C \subseteq V \times V$
  - Grado de un nodo/conmutador (d).
    - Número de canales de E/S del nodo/conmutador.
  - Tamaño de la red.
    - No. de EP conectados.
  - Número de conmutadores.
  - Diámetro de la red (D).
    - Máxima longitud de los caminos óptimos entre pares de EP
      - Si la topología importa, mejor cuanto D más pequeño

Topología de red

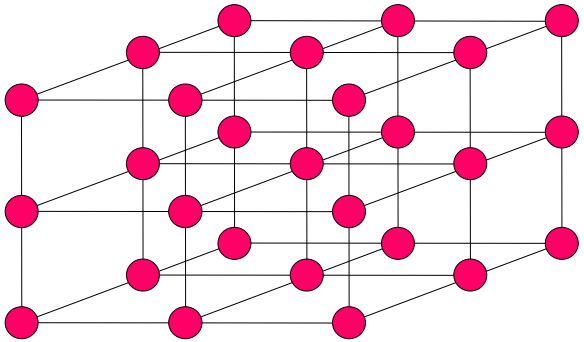
- Definiciones y propiedades
  - Simetría
    - Visión de la red por parte de los nodos
    - Patrón de tráfico uniforme: produce una carga uniforme de los canales de la red si ésta es simétrica
  - Homogeneidad
    - Igualdad en las características de los nodos
  - Regularidad
    - Patrón de construcción/organización de la red
    - Los nodos tienen el mismo grado
  - Ancho de bisección
    - nº canales a cortar para dividir la red en dos mitades iguales
    - Ancho de banda de bisección: volumen de tráfico a través de esos canales

Clasificación topológica

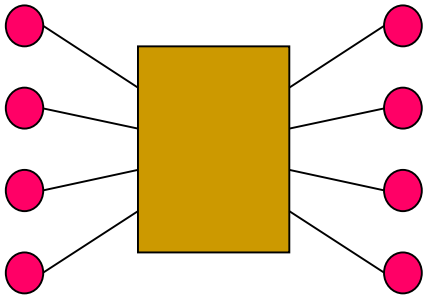
- Tipos de clasificaciones
  - Medio compartido vs. Punto-a-Punto
  - Regulares vs. Irregulares
  - Estáticas/Directas vs. Dinámicas/Indirectas

- Habitualmente (Duato, Yalamanchili, Ni)
  - Redes de medio compartido
    - Económicas, pero poco escalables
    - Bus, LAN (ej. Ethernet, etc)
  - Redes estáticas o directas
    - Malla, Toro, Hipercubo, etc
  - Redes dinámicas o indirectas
    - Crossbar
    - Redes multietapa (MIN)

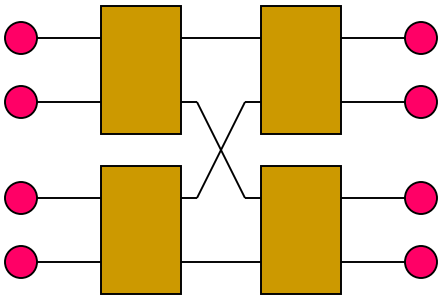
Clasificación topológica



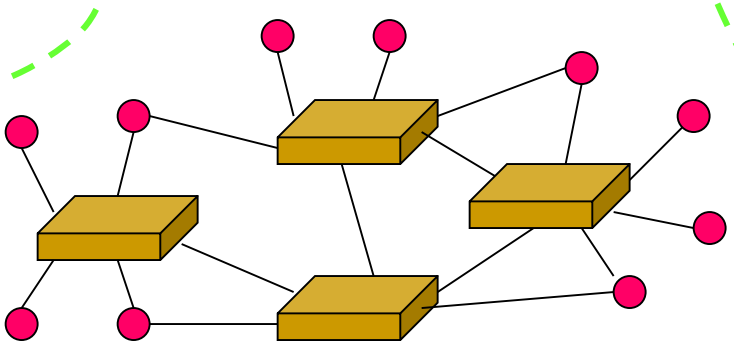
Red estática con topología regular



Crossbar



Multietapa

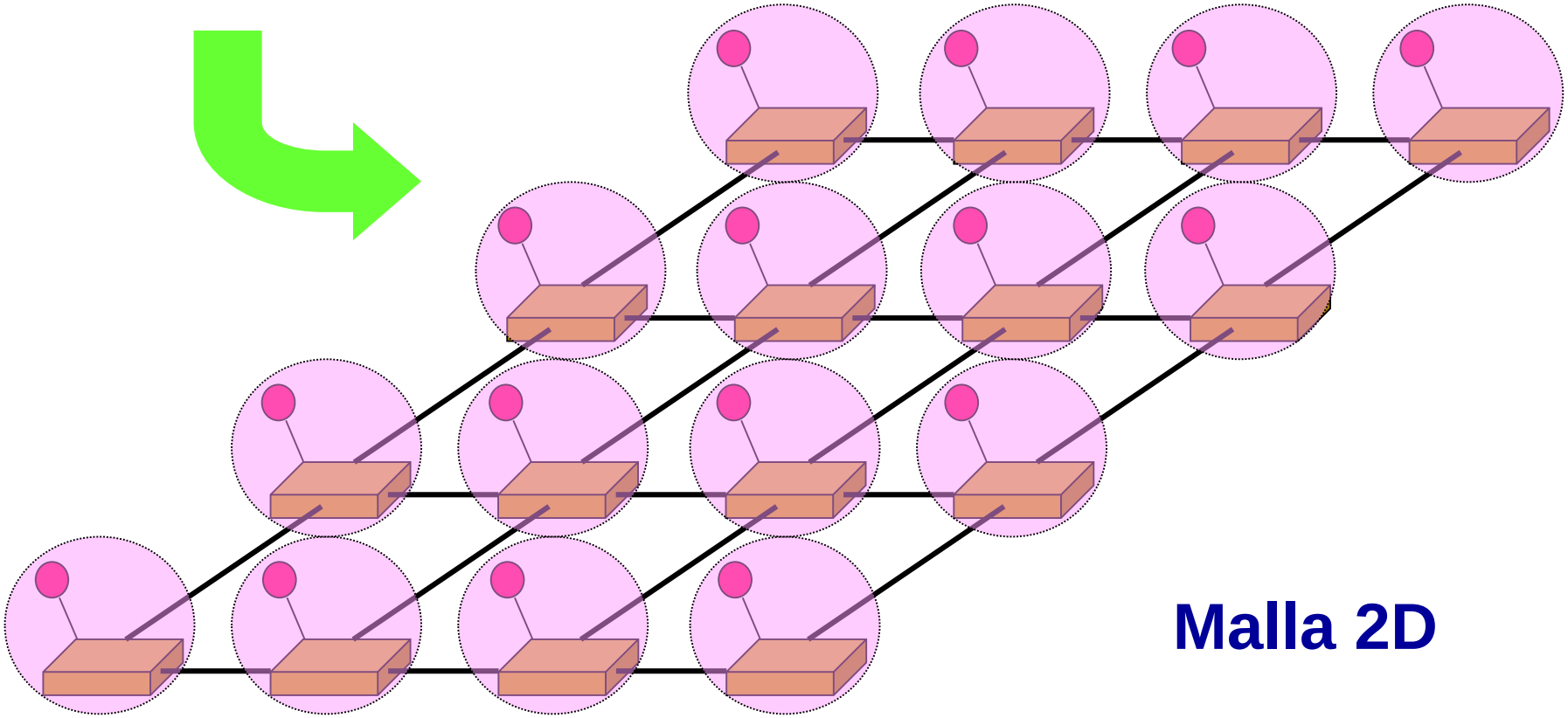


Red irregular

Redes dinámicas

Clasificación topológica

Una red directa equivale a una red indirecta donde a cada conmutador se conecta un único nodo



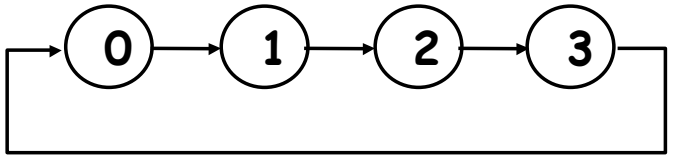
Malla 2D

Redes estáticas

- Anillo unidireccional, *p* nodos

  $F(i)=(i+1) \bmod p,\ 0 \le i \le p$

  - *Grado: d= 1, Diámetro: D = p-1*
  - Simetría = Si

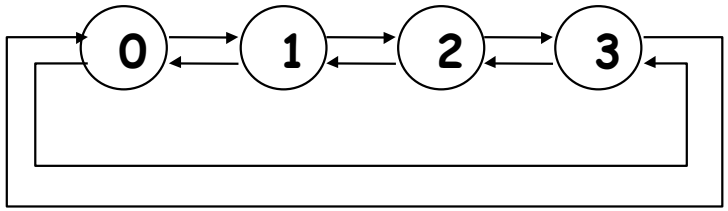


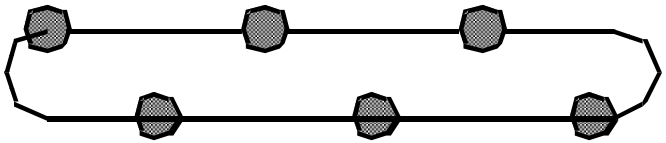
- Anillo bidireccional, *p* nodos

  $F_{+1}(i)=(i+1) \bmod p,$

  $F_{-1}(i)=(i+p-1) \bmod p,$

  - *d=2, D = p/2*
  - Simetría = Si
  - Ejemplos: FDDI, SCI, KSR1, FiberChannel Arbitrated Loop, Sandy Bridge





Redes estáticas

- Malla 2D, *r x r* nodos

  $F_{+x}(i)=(i+1)$ (existe si $i \bmod r \neq r-1$)

  $F_{-x}(i)=(i-1)$ (existe si $i \bmod r \neq 0$)

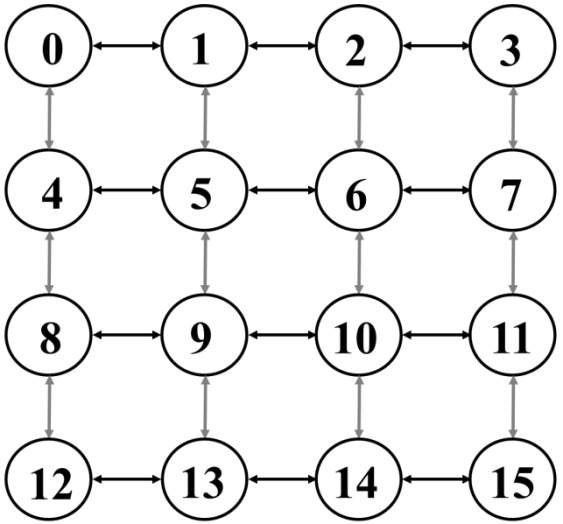
  $F_{+y}(i)=(i+r)$ (existe si $i < p - r$)

  $F_{-y}(i)=(i-r)$ (existe si $i \geq r$)

  donde $0 \leq i \leq p-1$, $p = r \times r$

  $d = 4$, $D = 2(r-1)$

  - Simetría = No



- Toro 2D, *r x r nodos*

  $F_{+x}(i)=(i+1) \bmod r + (i/r)\, r$

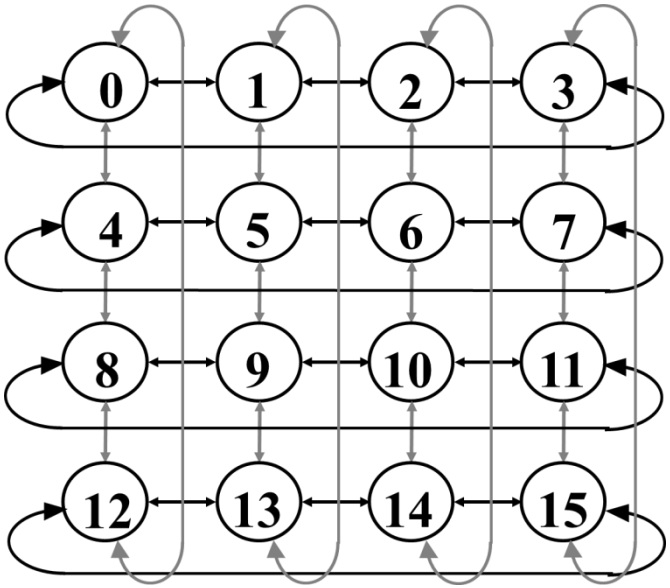
  $F_{-x}(i)=(i-1) \bmod r + (i/r)\, r$

  $F_{+y}(i)=(i+r) \bmod p$

  $F_{-y}(i)=(i+p-r) \bmod p$

  donde $0 \leq i \leq p-1$, $p = r \times r$

  - $d = 4$, $D = 2(r/2)$ ← *división entera*
  - Simetría = Sí



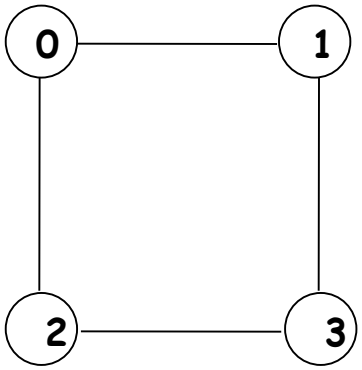
- Hipercubo binario, p= 2^n nodos

Hipercubo de orden $n = \log_2(p)$

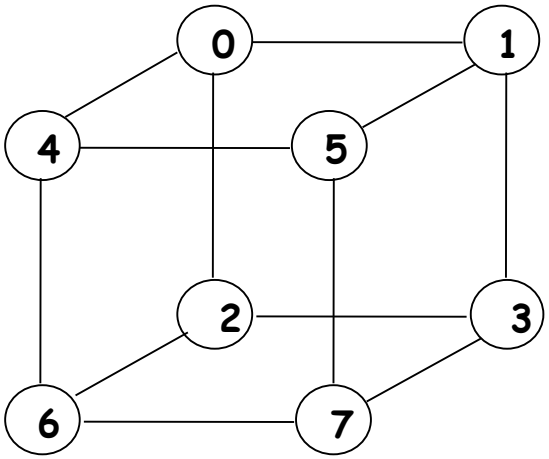
n funciones de interconexión, una por cada dimensión

$$F_i(h_{n-1}h_{n-2} \dots h_i \dots h_1 h_0) = h_{n-1}h_{n-2} \dots \bar{h}_i \dots h_1 h_0$$

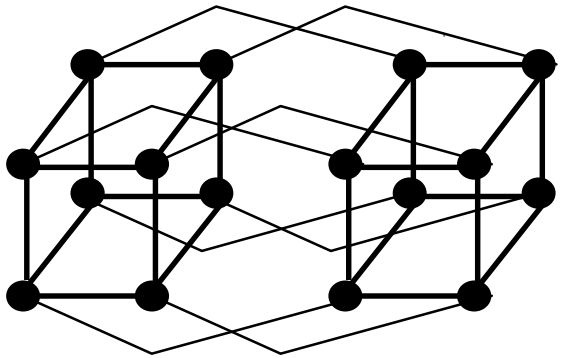
- *d = n, D = n*
- Simetría = Sí
- Ejemplo: Origin 2000/3000



2-cubo



3-cubo


4-cubo

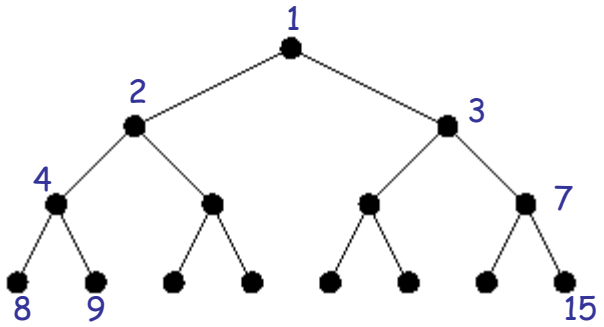
Redes estáticas

- Árboles binario, p nodos
  - Equilibrado

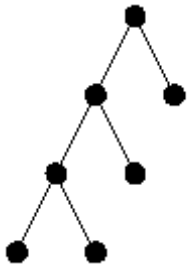
    $F_{izq}(i) = 2i, (i \leq p-1 / 2)$

    $F_{der}(i) = 2i+1$

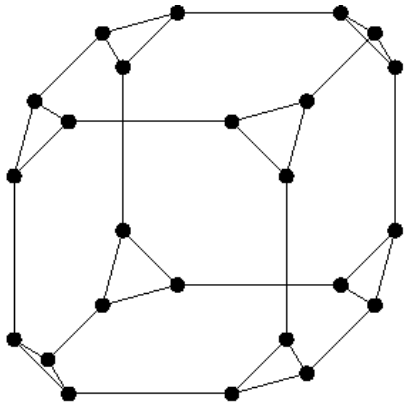
    donde $1 \leq i \leq p$



  - No-equilibrado



- Topologías híbridas
  - Reducir el grado del nodo manteniendo un diámetro pequeño
  - Ej: Ciclos conectados en Cubo



Algunos ejemplos

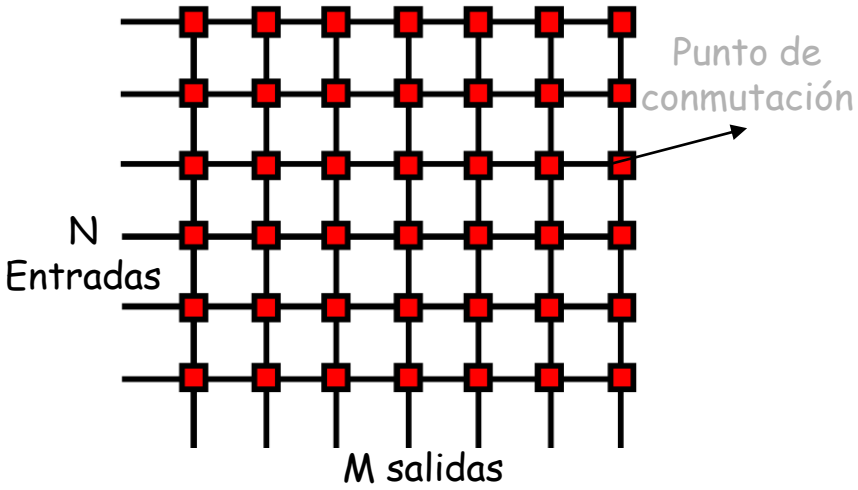
- Intel iPSC2 Hypercube:
  - Topología de hipercubo binario, canales serie de 2.8 Mbytes/s
- Chaos Router:
  - Toro 2D, 8 bits de datos en cada canal, 180 MHz por canal, 360 Mbytes/s en cada dirección
- Cray T3D:
  - Toro 3D bidireccional, hasta 1024 nodos (8x16x8), 24 bits de datos en cada dirección (16 de datos y 8 de control, 150 MHz por canal, 300 Mbytes/s por canal)
- Cray T3E:
  - Toro 3D bidireccional, 14 bits de datos en cada dirección, 375 MHz por canal, 600 Mbytes/s por canal
- MIT M-Machine:
  - Malla 3-D, 800 Mbytes/s por cada canal

Redes dinámicas

- Los EP se conectan a un conmutador, reconfigurable dinámicamente
- Modelo
  - Grafo con conmutadores como vértices y canales como arcos
- Topologías:
  - Regulares: Multiprocesadores UMA, SIMD
  - Irregulares: Networks of Workstations (NOW), LAN conmutadas
- Diferencias con redes estáticas:
  - 1 conmutador puede estar conectado a 0 o más EP
  - Sólo los conmutadores conectados a EP pueden ser origen/destino de un mensaje

Redes dinámicas (Crossbar)

- Caso ideal:
  - Interconexión global → red de barras cruzadas (crossbar):



- Definición
  - Red conmutada de N entradas y M salidas que permite hasta min(M,N) interconexiones punto a punto sin contención.

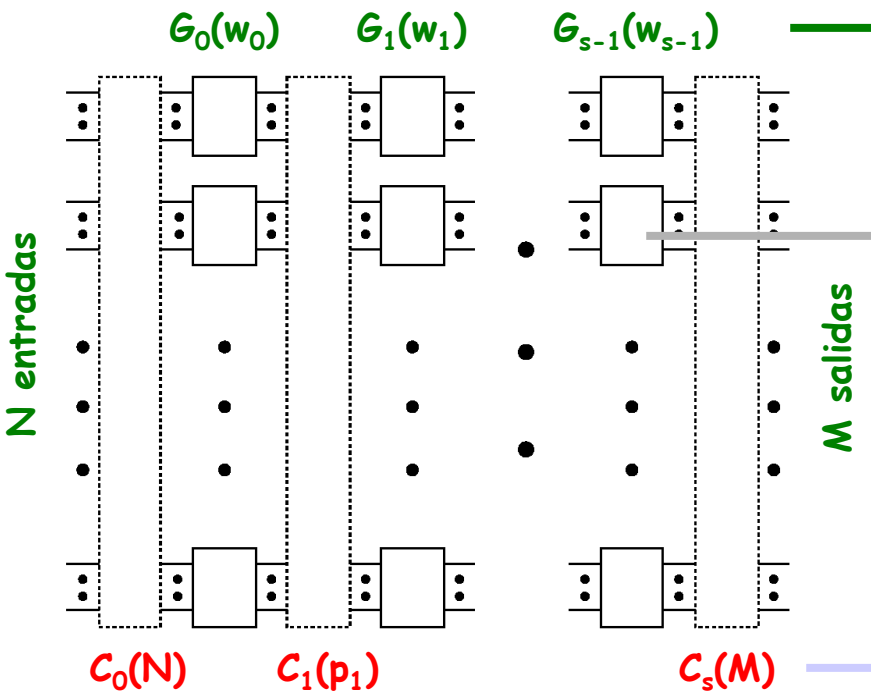
- Propiedades
  - Es posible una conexión siempre que los puertos de origen y destino estén libres
  - Coste $O(NM)$ → caro y difícil de construir si N y M grandes (nº de pines).

- Aplicaciones:
  - Multiprocesadores UMA de pequeña escala, para conectar procesadores con módulos de memoria
    - Todos los procesadores pueden acceder simultáneamente a memoria siempre que accedan a módulos diferentes.
  - Construcción de conmutadores para redes indirectas

Redes dinámicas (MIN)

- Multistage Interconnection Network (MIN)
- Descripción
  - Conectan dispositivos de entrada a dispositivos de salida a través de etapas compuestas por conmutadores y patrones de interconexión
  - Cada conmutador puede conectar sus E con sus S en cualquier permutación deseada.
- Estados de un conmutador 2x2 (unidireccional)

Modelo General de MIN



Conmutadores: interesa que todos los conmutadores sean idénticos para amortiguar el coste del diseño

- Las MINs fueron utilizadas inicialmente en redes telefónicas y posteriormente en procesadores en array.
  - En ambos casos, un controlador central establecía el camino entre la entrada y la salida, mediante la configuración de las señales de control de cada etapa de conmutadores, Gi.

Modelo general de MIN

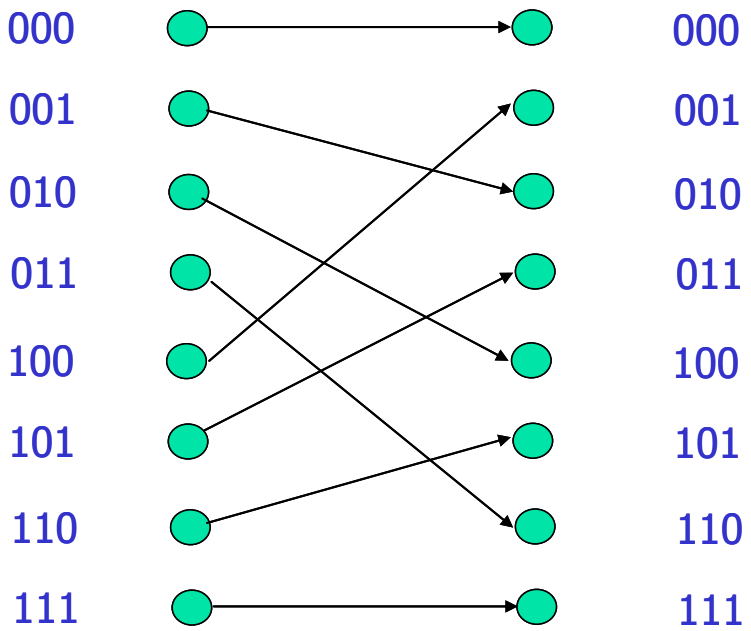
- Redes Banyan:
  - Una sola ruta entre un origen y un destino
- Red delta de N nodos: subclase Banyan con $N=k^n$:
  - Tiene n etapas, con N/k conmutadores (crossbar) de k x k en cada etapa
  - Es muy normal que k=2
- Permutaciones
  - Conexiones uno a uno entre los k^n puertos que definen los patrones de interconexión entre etapas
- Conectividad
  - Bloqueantes
    - El establecimiento de una conexión entre un nodo de entrada y otro de salida puede impedir el establecimiento de otras conexiones.
    - Ejemplo: redes Banyan
  - No bloqueantes
    - El conjunto de entradas puede conectarse con el conjunto de salidas simultáneamente en cualquier permutación.
    - Su conectividad es similar a la de un crossbar

- *Perfect shuffle* (σ)

  Permutación: $\sigma(x_{n-1}\, x_{n-2} \ldots x_1\, x_0) = (x_{n-2} \ldots x_1\, x_0\, x_{n-1})$

  e inversa: $\sigma^{-1}(x_{n-1}\, x_{n-2} \ldots x_1\, x_0) = (x_0\, x_{n-1}\, x_{n-2} \ldots x_1)$

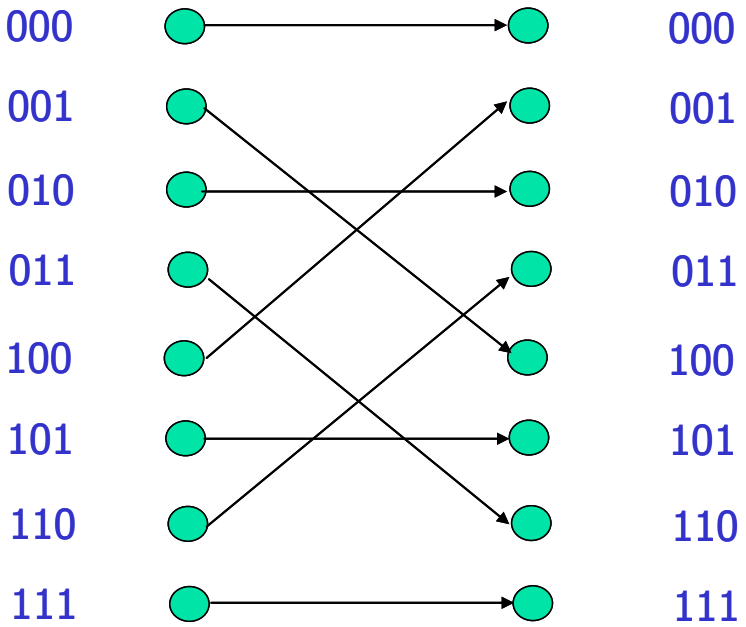
Ejemplo: *N=8, n=3, k=2.*



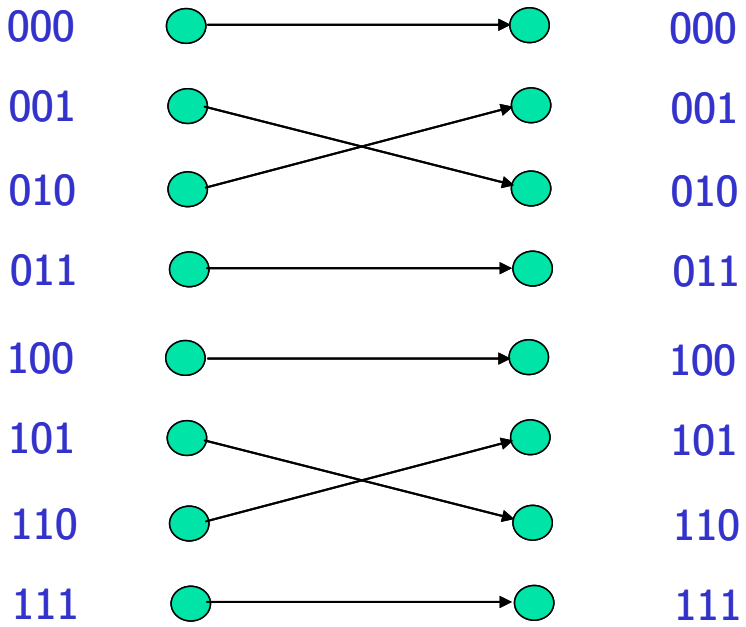
Butterfly (β)

$$\beta_i(x_{n-1}\ldots x_{i+1}\,x_i\,x_{i-1}\ldots x_1 x_0) = (x_{n-1}\ldots x_{i+1}\,x_0\,x_{i-1}\ldots x_1 x_i)$$

Ejemplo: $N=8, n=3, k=2, i=2$



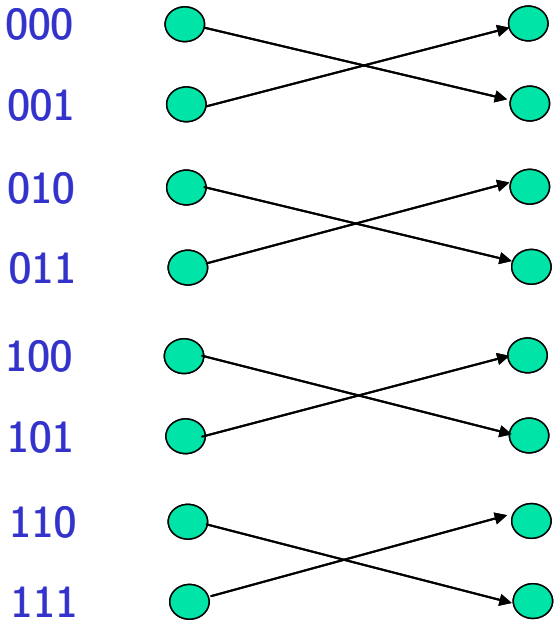
Ejemplo: $N=8, n=3, k=2, i=1$



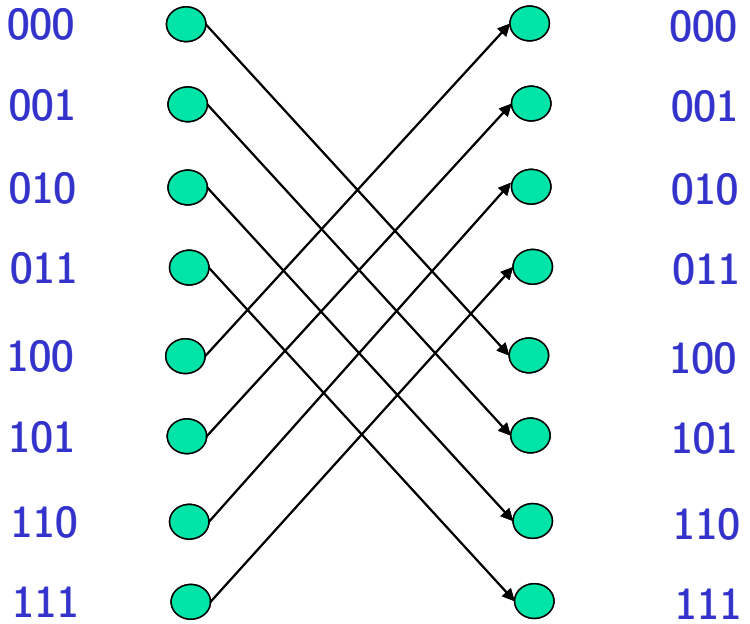
Exchange (E)

$$E_i(x_{n-1} \dots x_{i+1}\, x_i\, x_{i-1} \dots x_1\, x_0) = (x_{n-1} \dots x_{i+1}\, x'_i\, x_{i-1} \dots x_1\, x_0)$$

Ejemplo: $N=8, n=3, k=2, i=0$



Ejemplo: $N=8, n=3, k=2, i=2$



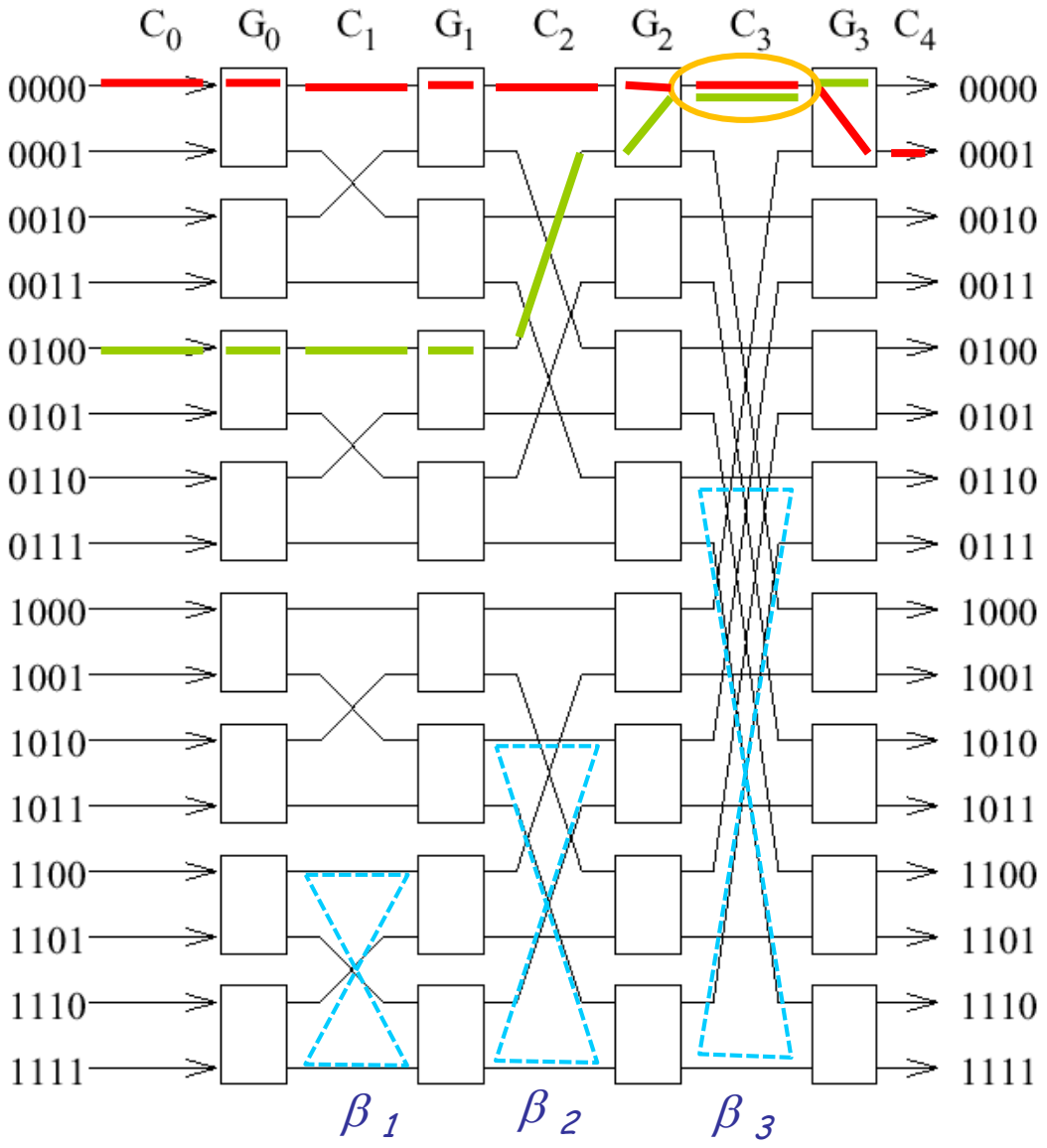
Redes MIN relevantes

- MIN *Butterfly*

Red delta de N nodos,
$N = k^n$ puertos,
$C_i \rightarrow \beta_i$, $C_0 \rightarrow \beta_0$, $C_n \rightarrow \beta_0$

Ejemplo:
$N=16, k=2, n=4$

$C_0 = C_4 = \beta_0 = I$
$C_1 = \beta_1$
$C_2 = \beta_2$
$C_3 = \beta_3$



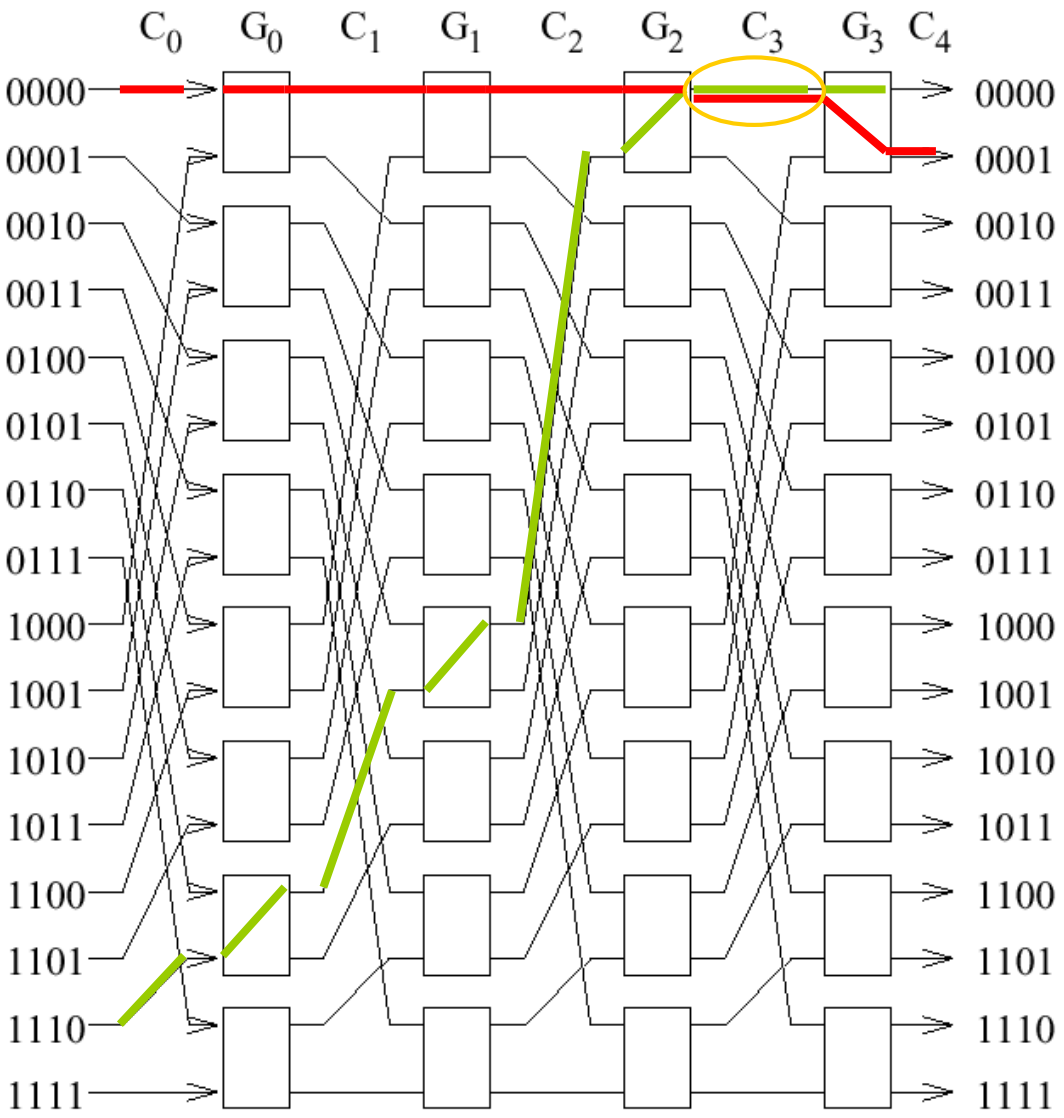
Bloqueante: P. ej. la ruta de 0 a 1 no es compatible con la ruta del 4 al 0.

Redes MIN relevantes

- MIN *Omega*

Red delta de N nodos,
$N = k^n$ puertos,
$C_i \rightarrow \sigma$, $C_n \rightarrow I$

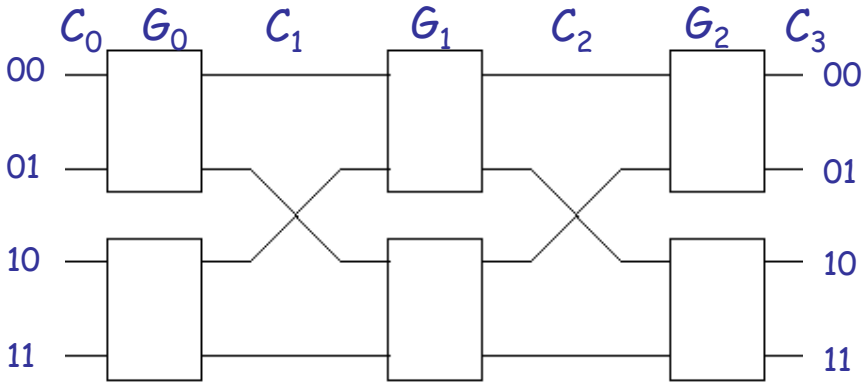
Ejemplo:
$N=16, k=2, n=4$



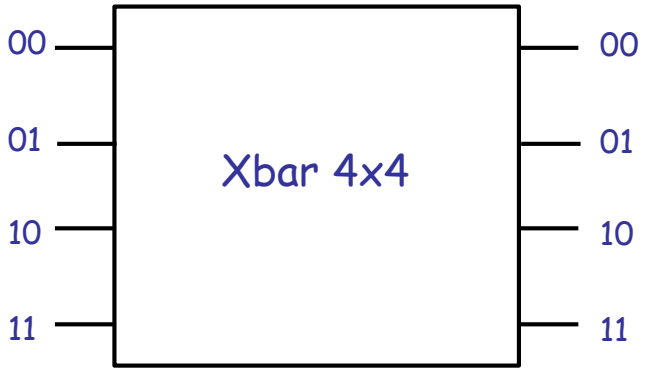
Bloqueante: P. ej. la ruta de 0 a 1 no es compatible con la ruta del 14 al 0.

Redes MIN relevantes

- MIN *Beneš*
  - Evitar bloqueo: poner etapas de conmutadores (2x2) adicionales o usar conmutadores más complejos (p. ej. 4x4)
  - La red Beneš de N nodos (Beneš NxN) usa $2(\log_2 N) - 1$ etapas
    - $\log_2 N$ como antes + ($(\log_2 N) - 1$) adicionales
  - Ejemplo: red Beneš 4x4 (n=2, k=2 ➔ $N=n^k=4$)
    - 3 etapas ($C_1 = \sigma^{-1}$, $C_{2n-2} = C_2 = \sigma$)
    - No hay bloqueo
    - Cualquier permutación E/S es posible ➔ equivalencia funcional con un Xbar 4x4



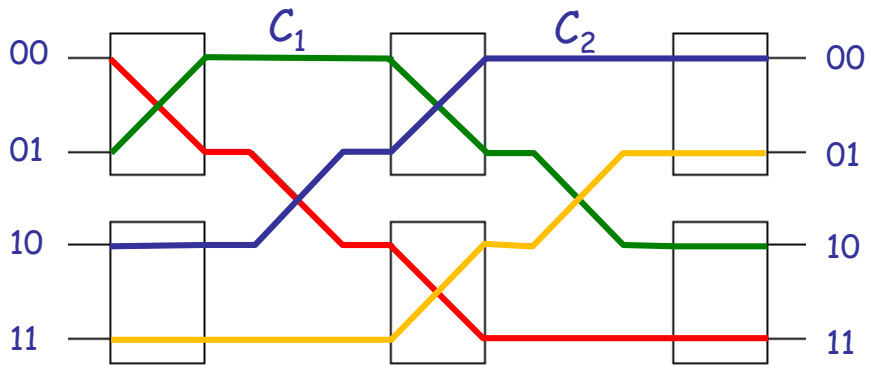
Beneš (4x4)



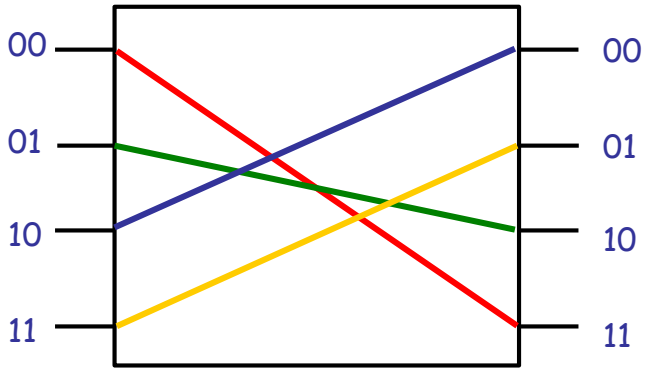
Redes MIN relevantes

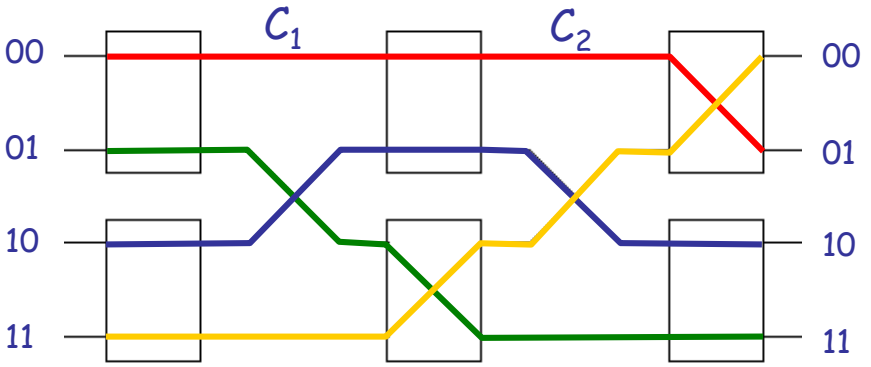
- Ejemplos

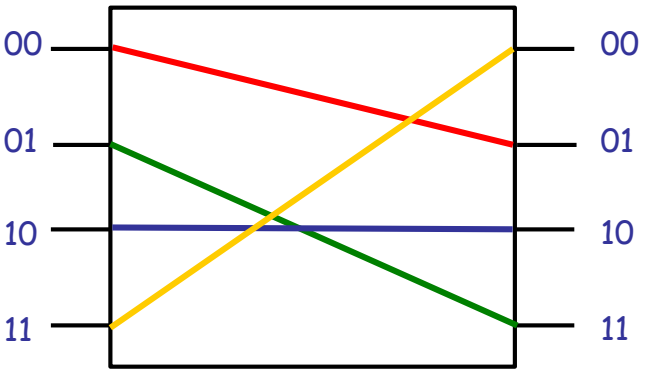
Beneš (4x4)



Xbar 4x4

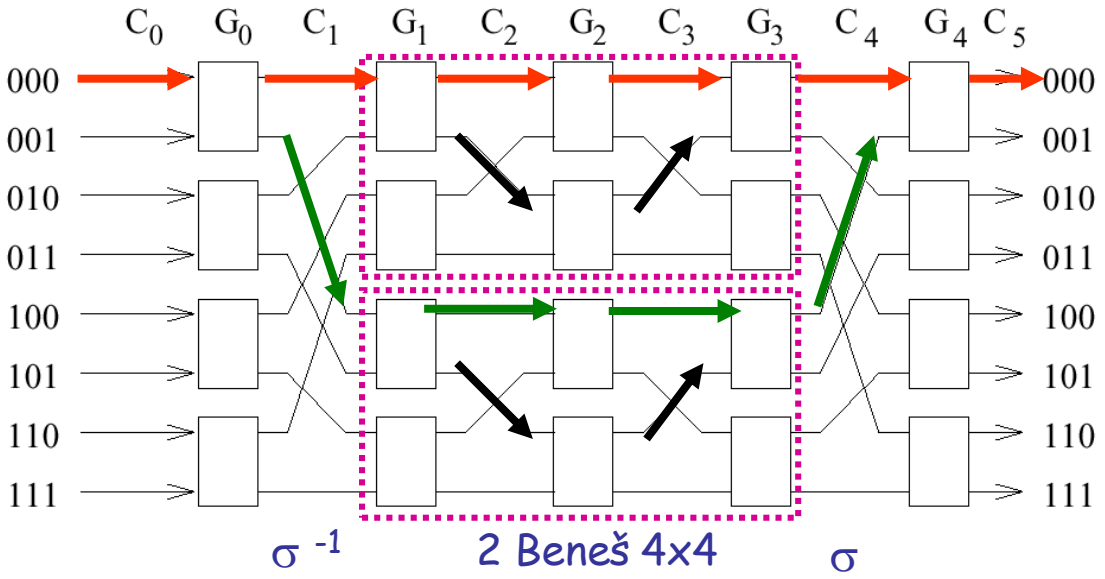






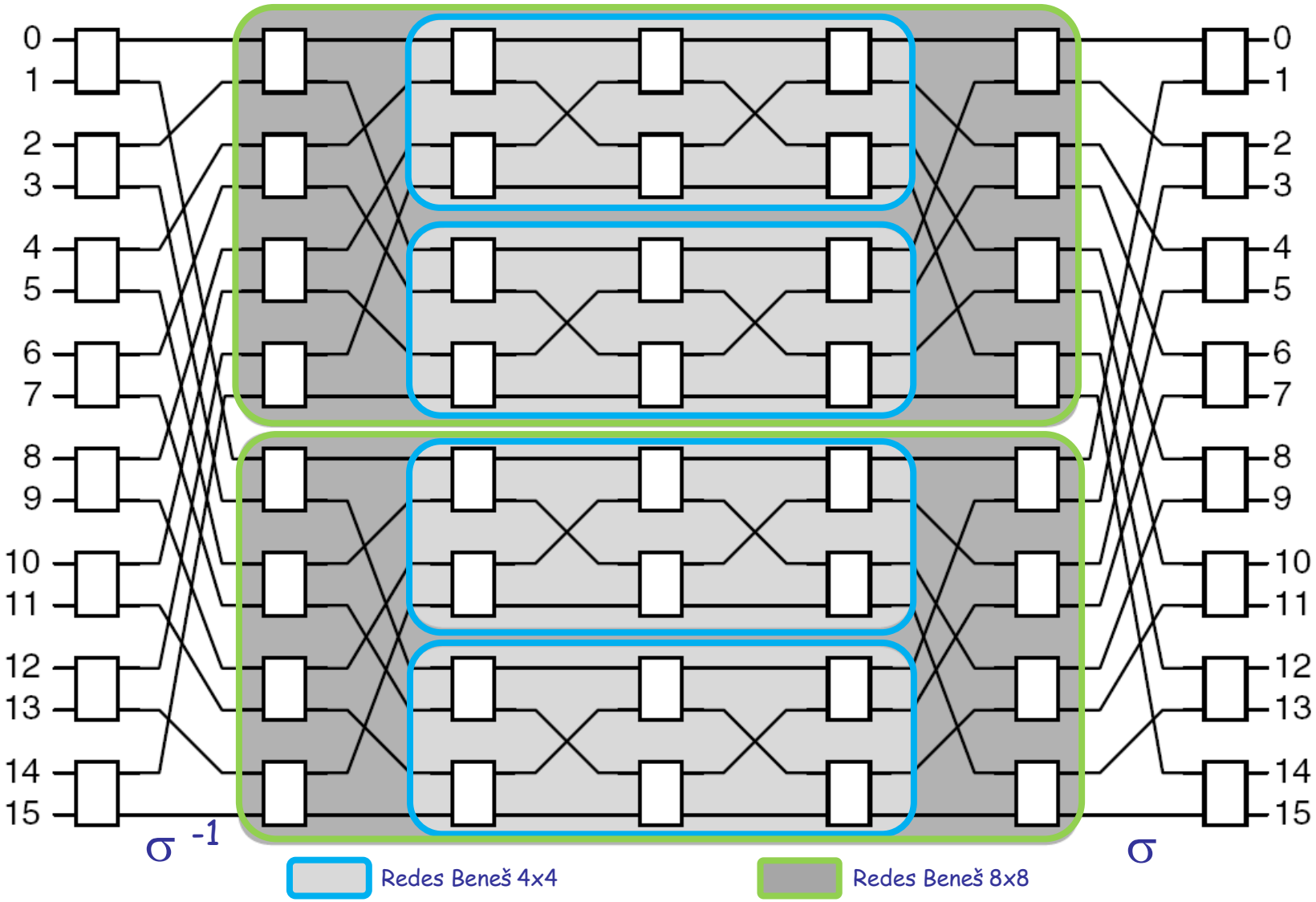
Redes MIN relevantes

- MIN Beneš
  - Se pueden construir redes con mayor nº de elementos a partir de redes más pequeñas
  - Una red Beneš de NxN, se construye con:
    - 2 redes de Beneš de N/2 x N/2
    - $C_1 \rightarrow \sigma^{-1}$, $C_{2n-2} \rightarrow \sigma$
  - Ejemplo: Beneš 8x8 (n=3, etapas: 2*3 – 1 = 5)
    - Las flechas coloreadas muestran caminos alternativos desde la entrada 0 a la salida 0.



Redes MIN relevantes

- Construcción iterativa de una red Beneš 16x16

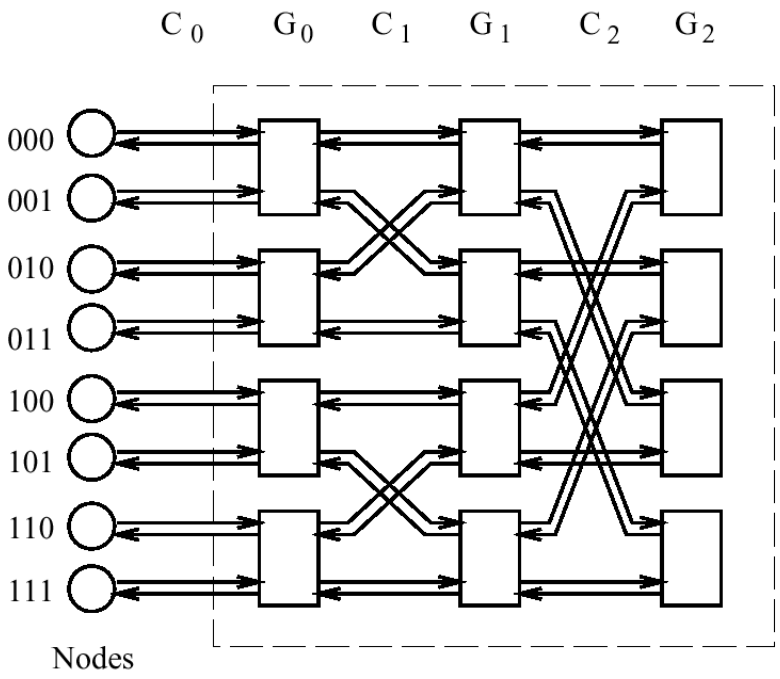


Redes MIN relevantes

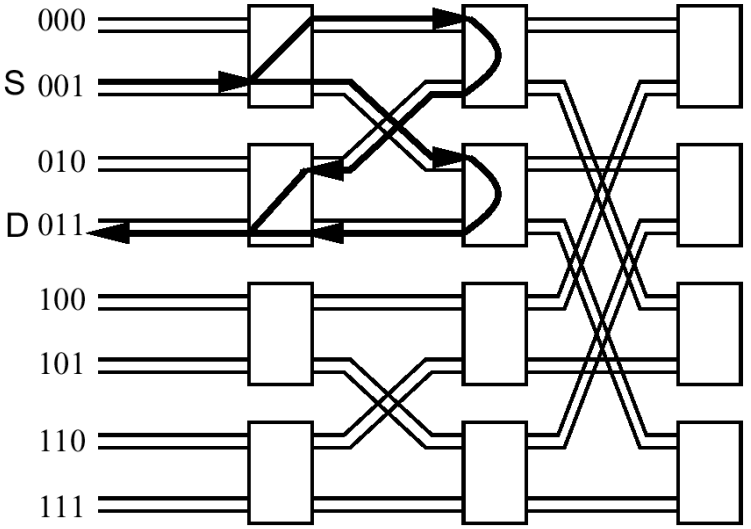
- MIN *Butterfly* bidireccional

  Pierde C_n y $C_i \rightarrow \beta_i$

Ejemplo: *N=8* y *k=2*

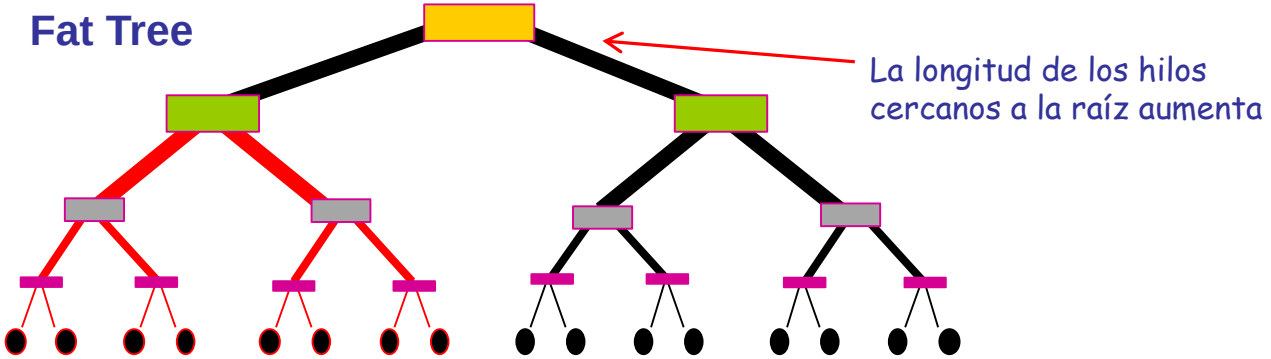


¡La bidireccionalidad proporcional caminos alternativos!



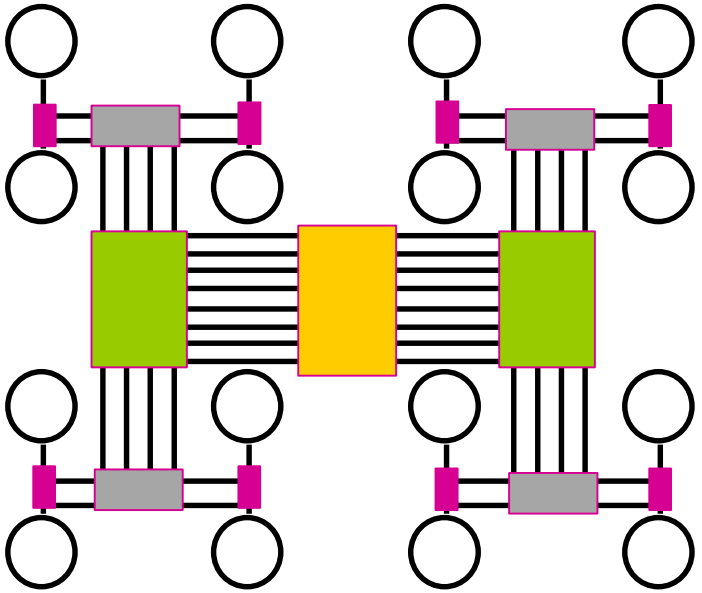
Redes MIN relevantes

- *Fat Tree*
  - MIN en árbol en el que el ancho se incrementa añadiendo más enlaces en los conmutadores cercanos al nodo raíz



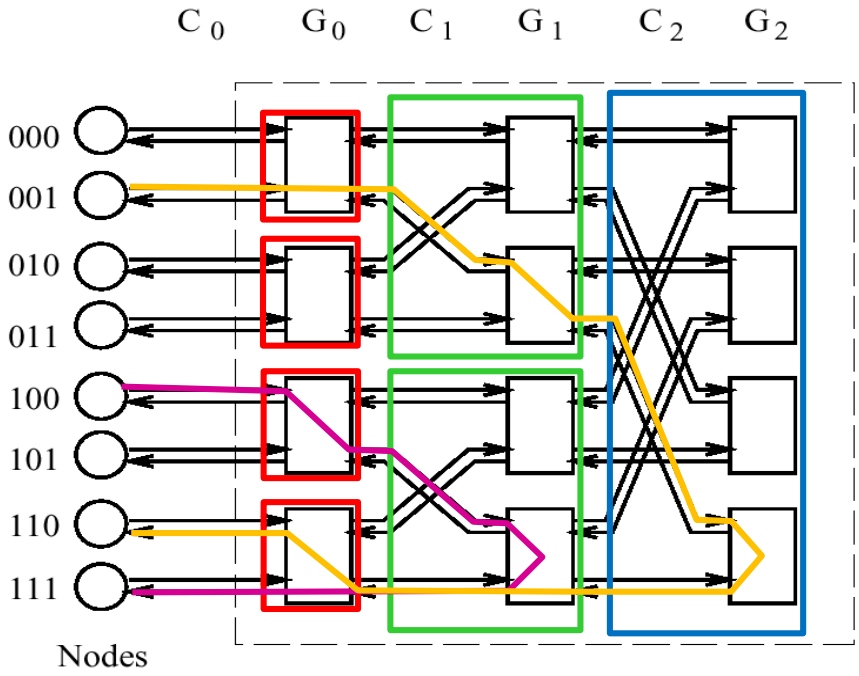
Emplazamiento: H-tree

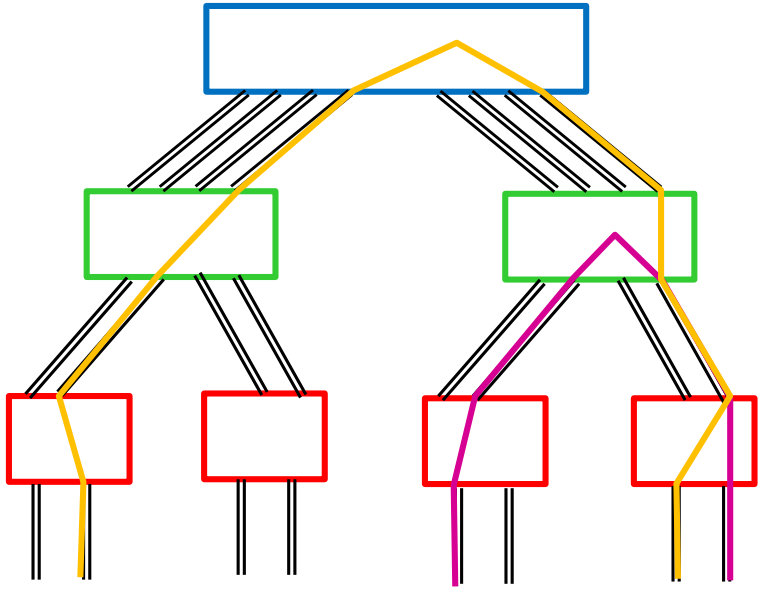
(Reduce el problema de la longitud de los hilos cercanos a la raíz)



Redes MIN relevantes

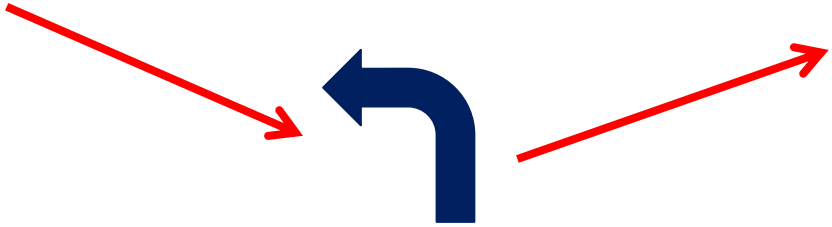
- Equivalencia Butterfly bidireccional ⇔ Fat-tree





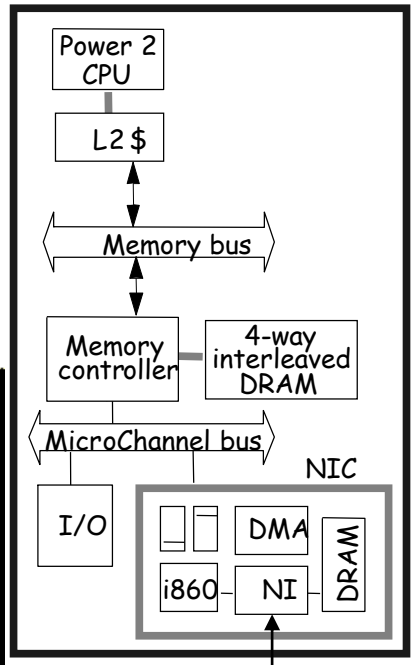
Conexión 4→7

Conexión 1→6

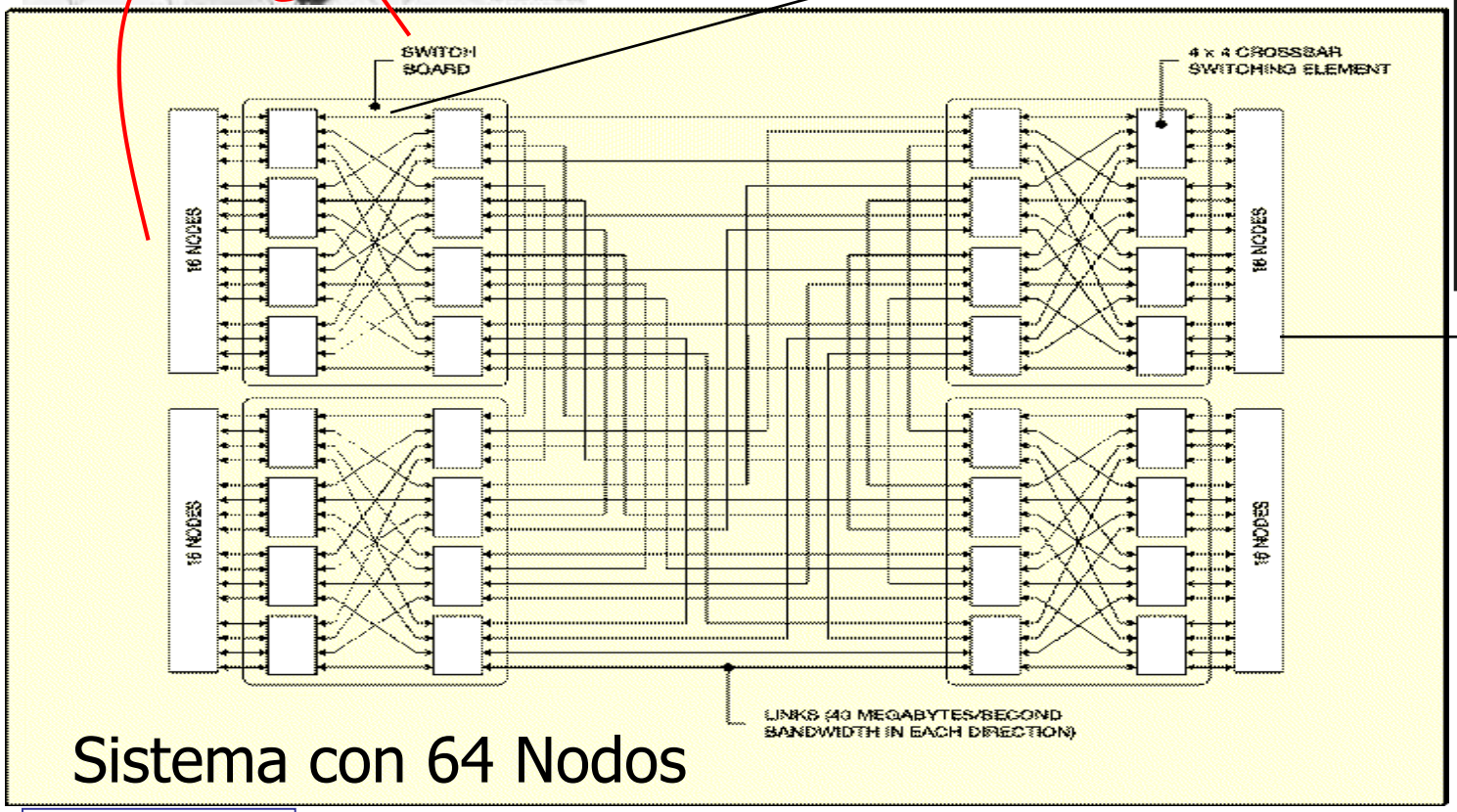


Ejemplos: IBM SP2

Nodo original IBM SP-2



Permutación Baraje Perfecto base 4



Sistema con 64 Nodos

Técnicas de conmutación

- Enfoques básicos: Conmutación de circuitos vs. conmutación de paquetes

- Conmutación de circuitos.
  - Antes del comienzo del envío del mensaje se establece cuál será la secuencia de enlaces que atravesarán los datos desde el origen al destino.
    - Un mensaje sin datos (cabecera) recorre el camino completo y va reservando los enlaces.
    - Cuando se alcanza el destino se devuelve un ACK al origen por el mismo camino ya establecido.
  - Una vez establecido el camino todos los datos lo siguen.
  - Los canales que componen el camino quedan reservados desde el comienzo hasta el final de la transmisión ➔ durante ese tiempo no puede usarse para otras comunicaciones.

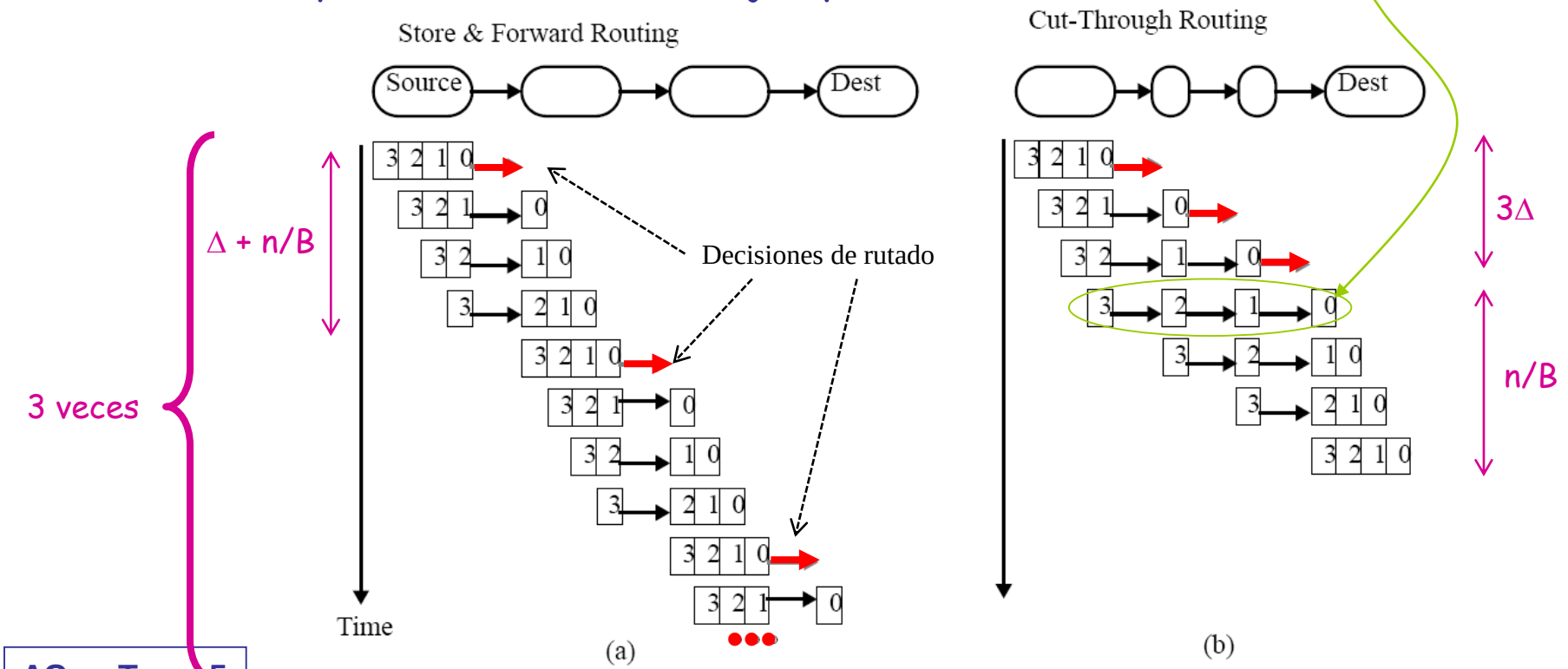
- Conmutación de paquetes.
  - Cada decisión de rutado se va tomando según el mensaje progresa
  - Los enlaces que no estén siendo utilizados en un momento dado, pueden usarse para otra comunicación .
  - Variantes:
    - Store-and-Forward
    - Cut-Through

Técnicas de conmutación: latencia

- Suposiciones
  - No hay congestión en la red: latencia básica
  - Se quiere transmitir un mensaje que atravesará h enlaces (hops)
  - El tamaño del mensaje es n bits y el ancho de banda es B
  - El tiempo para cada decidir cuál deber ser el enlace siguiente a recorrer (rutado) y establecer la conexión con ese enlace (conmutación) es Δ
- Latencia con conmutación de circuitos
  - $Tcc(n,h) = h\Delta + n/B$
    - tiempo previo para decidir la ruta, más tiempo de transmitir el mensaje completo de fuente a destino
    - Ventajas: una vez seleccionada ruta (circuito) la transmisión se produce a la máxima velocidad (determinada por el AB)
    - Inconvenientes: reserva de los canales (desde antes de comenzar la transmisión hasta que el último dato llega al destino)
- Latencia con Store-and-Forward
    - El mensaje completo se transmite de un nodo al siguiente.
    - Cuando se ha recibido completamente, se decide el siguiente enlace a seguir y se vuelve a transmitir ➔ h transmisiones + h decisiones
  - $Tsf(n,h) = h\Delta + h(n/B) = h(\Delta + n/B)$
    - Menos enlaces reservados que en CC, pero ...
    - Latencia mucho peor: Proporcional a h (topología)
  - ¿Hay alguna forma de no reservar enlaces y tener mejor latencia?

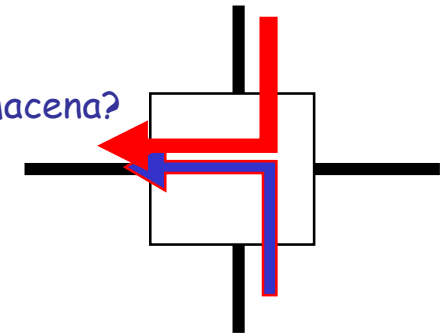
Técnicas de conmutación: latencia

- Latencia con Cut-Through
  - El paquete está dividido en trozos de forma natural: flits
  - Cuando se recibe el primer flit decidir el siguiente enlace de la ruta y comenzar la retransmisión, sin esperar a la recepción de los restantes flits
  - Tct (n,h) = hΔ + n/B
    - La transmisión es similar a CC, pero de forma segmentada.
    - No hay reserva de circuitos, pero tiempo básicamente comparable a CC
- Comparación SF vs. CT (ejemplo: h=3)



Técnicas de conmutación: contención

- El problema de la contención
  - Dos paquetes intentan usar el mismo canal al mismo tiempo
    - ¿Quién pasa? Arbitraje
    - ¿Qué ocurre con el paquete que se bloquea? ¿Dónde se almacena?
    - El tamaño de la memoria es limitado
    - Cómo afecta a las prestaciones → latencia
    - Caso extremo → Saturación e Interbloqueos (Deadlock)

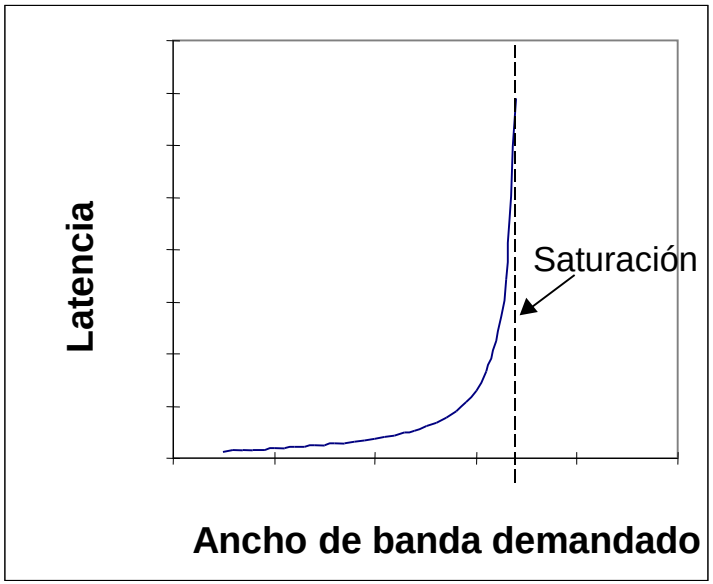


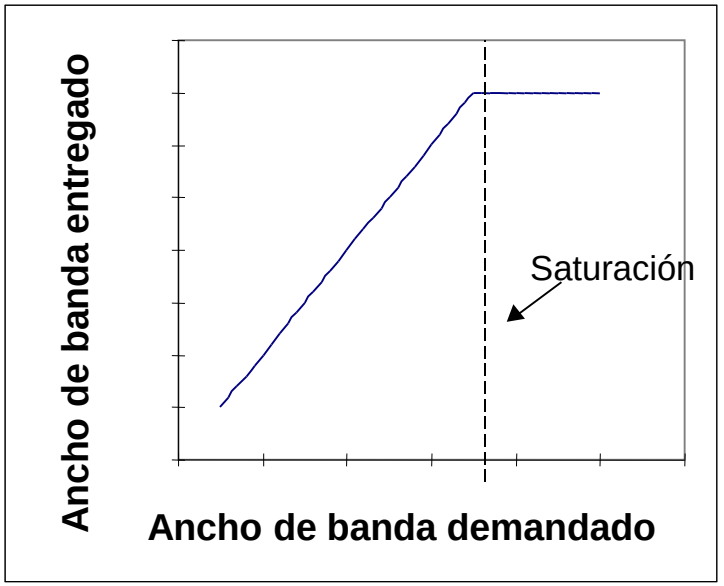
- *Store&Forward*
  - El paquete bloqueado se recibe y se almacena completo en buffers del nodo/encaminador

- *Cut-through*
  - *Virtual Cut-Through*. Los flits del paquete bloqueado se van recibiendo y almacenando en buffers del encaminador
    - En situaciones de contención ➔ comportamiento se degrada hasta ser similar a SF
    - Necesitamos buffers con tamaño suficiente para almacenar paquete
  - *Wormhole*. Los buffers del encaminador sólo almacenan unos pocos flits del mensaje bloqueado. Los restantes flits permanecen en los encaminadores previos sin avanzar.
    - Buffers más pequeños
    - Un paquete bloqueado en la red puede estar repartido entre varios encaminadores a lo largo de la ruta ➔ Propagación hacia atrás (back-pressure) de la contención (mecanismo de control del flujo)
    - Cuidado con interbloqueos --> ocupación extensiva de los canales

Latencia y AB

- La latencia y el AB entregado por la red son dependientes del AB demandado
  - Mientras que el AB demandado está claramente por debajo de la capacidad máxima de la red (poca contención) la latencia se mantiene baja (básicamente cte)
  - A partir de cierto punto, si se demanda más AB (los EP producen más paquetes por ut), se produce la saturación
    - La latencia aumenta rápidamente
    - El AB entregado no se incrementa
  - Recordar! Un multiprocesador es un sistema cerrado
    - Aumento AB demandado ➔ Aumento latencia comunicaciones ➔ Reducción avance programas ➔ Reducción paquetes generados ➔ Reducción AB demandado.

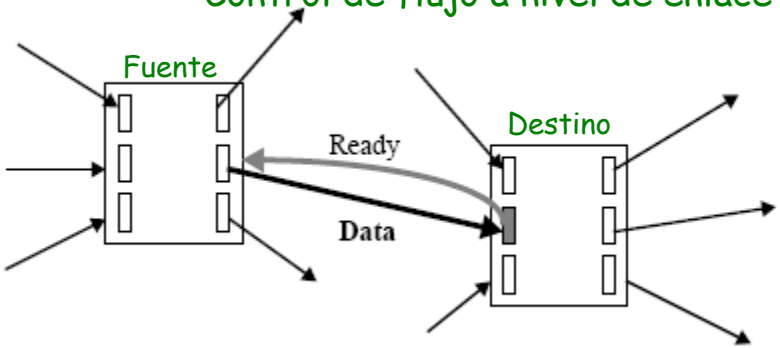




Control de flujo

- Problema: Cómo proceder cuando múltiples flujos de datos intentan usar los recursos compartidos de la red al mismo tiempo
  - Problema típico de las redes, pero ...
  - En computadores paralelos tiene características específicas
    - Los datos deben ser trasmitidos de forma tan fiable como se hace p. ej. en un bus
    - Muchos flujos de datos simultáneos
    - Escala de tiempo muy pequeña
  - Variantes del problema
    - A nivel de enlace: Entre nodos físicamente conectados
    - Extremo-a-extremo

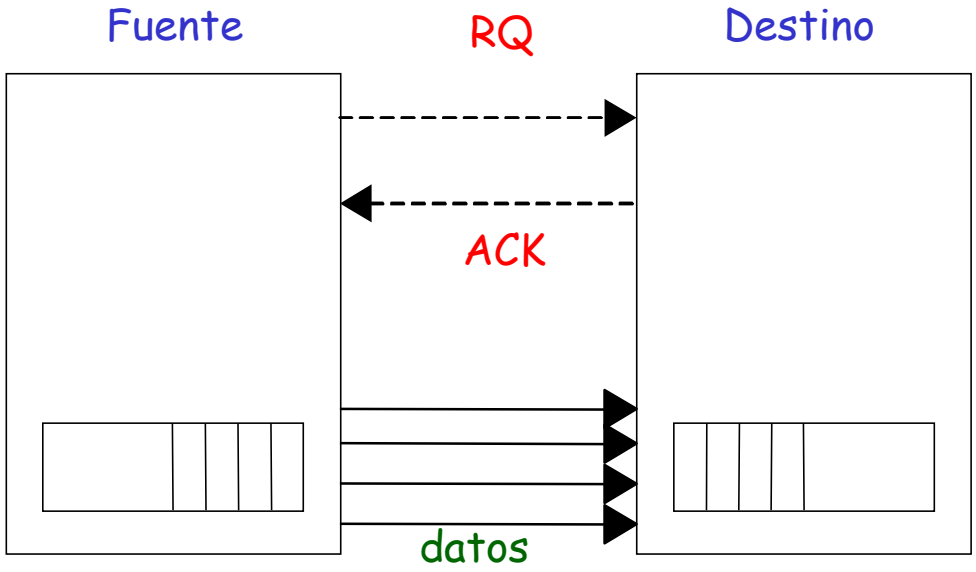
Control de flujo a nivel de enlace



Si el buffer de entrada del nodo destino no puede recibir datos, éstos deben retenerse en el nodo fuente

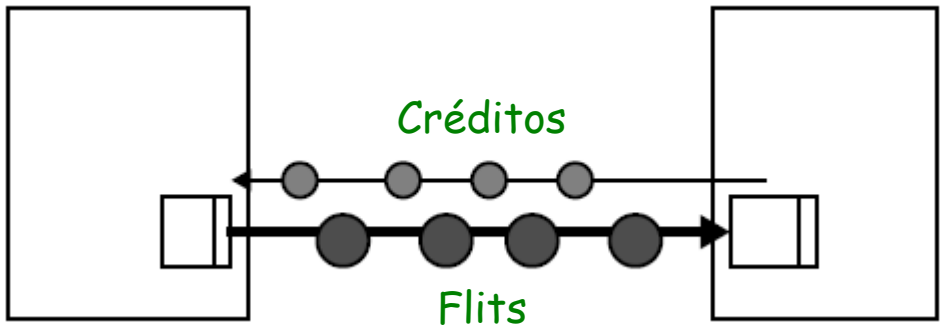
Control de flujo: enlace

- Enlaces cortos
  - Control de flujo asíncrono a nivel de flit
  - La fuente activa RQ cuando tiene un flit para transmitir
  - El destino reconoce la recepción (activa ACK) cuando está listo para recibir el siguiente flit
  - Hasta ese momento la fuente permanece transmitiendo el mismo flit



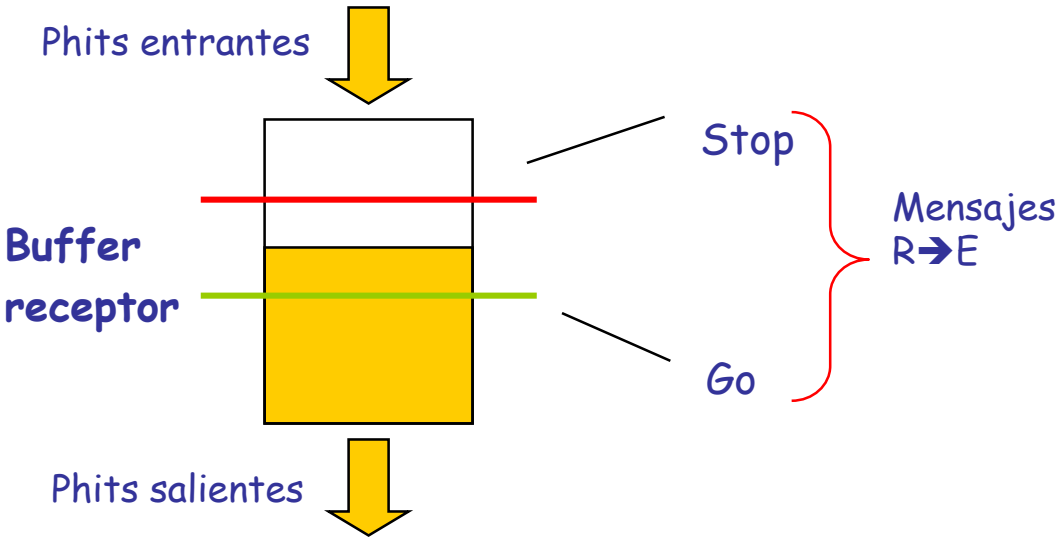
Control de flujo: enlace

- Enlaces largos
  - Problema del retardo de propagación de los flits en el medio
    - No es operativo un "handshake" a nivel de flits individuales
  - Mecanismo de control basado en créditos
    - El emisor decrementa el contador con cada envío de un flit
    - Para el envío cuando llega a cero
    - El contador se incrementa con cada reconocimiento (crédito)
    - Los símbolos de crédito son enviadas por el receptor al ir vaciando su buffer
    - Problemas
      - Robustez: Sensible a la pérdida de símbolos de crédito
      - Retardos de propagación (inercia)



Control de flujo: enlace

- Enlaces largos
  - Mecanismo Stop-and-Go
    - El buffer de entrada del receptor se considera como una especie de "tanque" que almacena mensajes.
    - Cuando el "nivel" de mensajes supera un determinado valor, el receptor manda una orden "STOP" al emisor.
    - Cuando el "nivel" de mensajes almacenados desciende por debajo de cierto umbral, el receptor manda al emisor una orden "GO"
    - Ventaja: los símbolos STOP y GO pueden enviarse más de una vez sin causar problemas a la red



Control de flujo: extremo a extremo

- Influencia del nivel de enlace en el flujo extr-extr
  - Propagación hacia atrás de la congestión persistente en un nodo (back-pressure)
- Hot-spots. Ejemplo: varias fuentes quieren transmitir a un cierto destino.
  - Si el tráfico que generan las fuentes es demasiado para la capacidad del destino, habrá un efecto de back-pressure y las fuentes regularán su tráfico
  - Problema: cuando las fuentes detecten esa congestión, todos los buffers del árbol que va desde las fuentes al destino (hot-spot) están llenos.
  - Más problema: cualquier otra comunicación que atraviese ese árbol se verá afectada por la congestión.
- Comunicación extremo a extremo basada en créditos
  - Aproximación al problema de los hot-spots
  - P. ej. permitir solo el envío de un mensaje extr-extr hasta que se reciba la confirmación procedente del destino

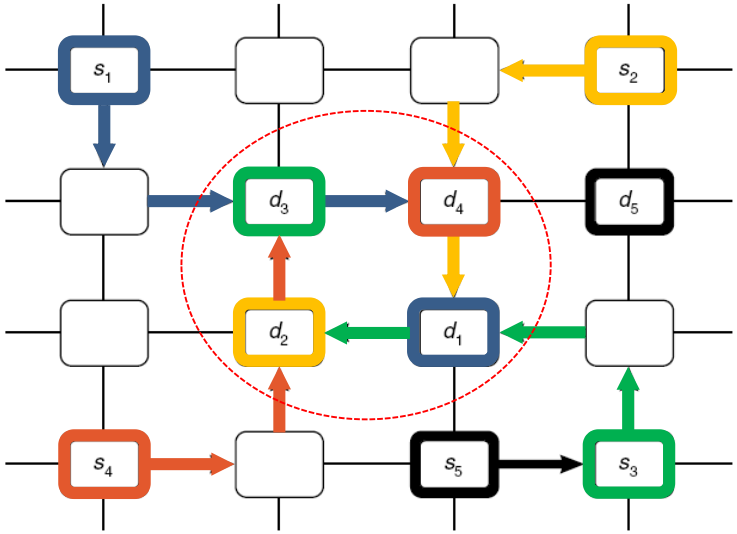
Encaminamiento

- Objetivo
  - Determinar la ruta que debe seguir un mensaje desde el origen hasta el destino
  - Comportamiento ideal
    - Baja latencia
    - Distribución de la carga
    - Tolerancia a fallos
    - Libre de interbloqueos
- Mecanismo de encaminamiento
  - Aritmético: determinar el siguiente tramo de la ruta mediante una op aritmética simple
  - Selección de la ruta en origen: la información de ruta acompaña al paquete. Problema: incremento de longitud.
  - Mediante tablas: Cada conmutador contiene una tabla de rutas. Cada paquete contiene un índice de acceso a la tabla de cada de cada conmutador atravesado.
- Propiedades del encaminamiento
  - Mínimo / No mínimo
  - Determinista / Adaptativo

Problemas del encaminamiento

- Interbloqueo (deadlock)
  - Un paquete está a la espera de un evento que no puede ocurrir
    - P. ej. Los paquetes no pueden avanzar hacia su destino porque cada uno está esperando en un enlace a que otro paquete deje disponible un buffer de entrada. Pero eso no puede ocurrir porque los enlaces bloqueados forman un ciclo
  - Una red bien diseñada debe estar libre de interbloqueos

Ejemplo (interbloqueo en una malla 2D): 4 mensajes simultáneos de s_i a d_i (i=1,4)



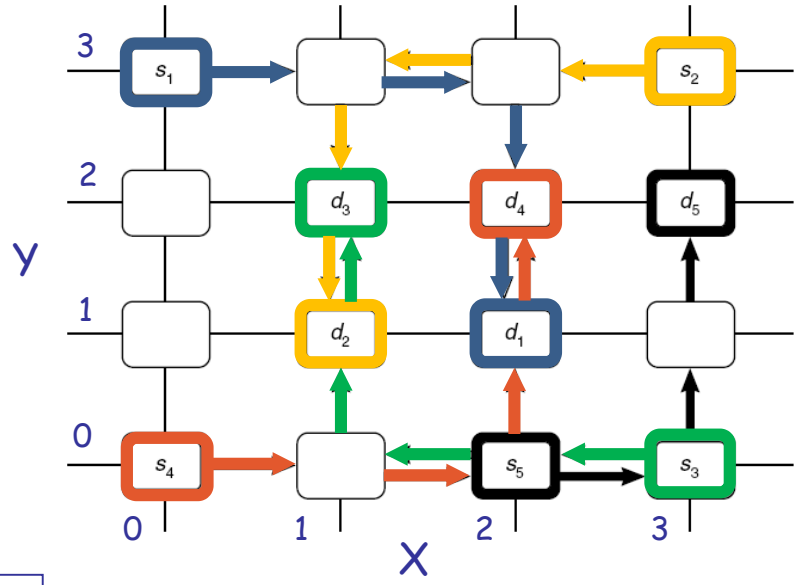
Cada mensaje hace tres saltos y queda interbloqueado en el ciclo (d1, d2, d3, d4)

El mensaje de s5 a d5 tb se ve afectado por el interbloqueo

- Espera indefinida (starvation)
  - Un paquete está a la espera de un evento que puede ocurrir, pero no ocurre. Puede deberse a una mala planificación del sistema (fairness)
- Ciclos infinitos (livelock)
  - Un paquete está recorriendo nodos sin llegar nunca a su destino. Sólo puede ocurrir con encaminamiento adaptativo no mínimo

Mecanismos de encaminamiento

- Orden dimensional (Dimension-order routing)
  - En la propagación del paquete las diferentes dimensiones de la red (X, Y, Z, ...) se siguen en un orden estricto
  - Los enlaces en una cierta dimensión no pueden ser utilizados por un paquete hasta que no ha recorrido todos los enlaces necesarios (para alcanzar el destino) en todas las dimensiones precedentes.
  - Aplicación en mallas, hipercubos, ...
  - Ejemplo en una malla 2D: los paquetes siguen el orden dimensional X,Y.
    - El anterior interbloqueo no existe
    - Problema potencial: sólo existe una ruta posible para cada comunicación (determinista)
      - Saturación, no tolerancia a fallos, desaprovechamiento de enlaces poco usados, ...

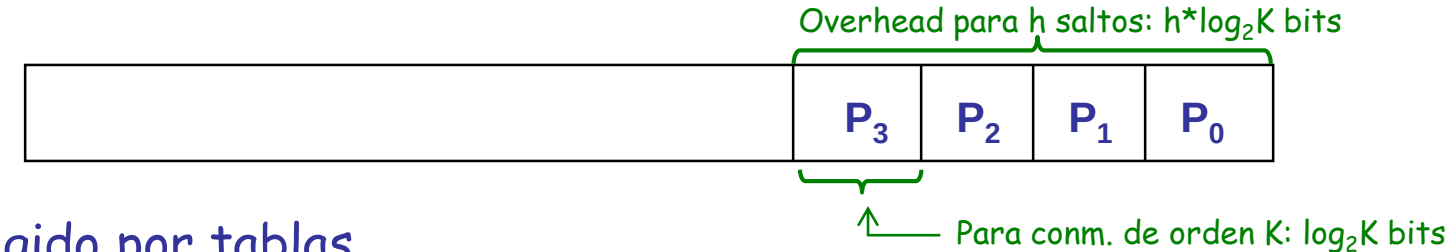


Algoritmo: paquete desde (x1,y1) hasta (x2,y2)

- Nodo actual: (x,y). Inicialmente (x,y) = (x1,y1)
- (Dx,Dy)=(x2-x, y2-y)
- Si Dx>0, incrementar x (este)
- Si Dx<0, decrementar x (oeste)
- Si Dx=0
  - Si Dy>0, incrementar y (norte)
  - Si Dy<0, decrementar y (sur)

Mecanismos de encaminamiento

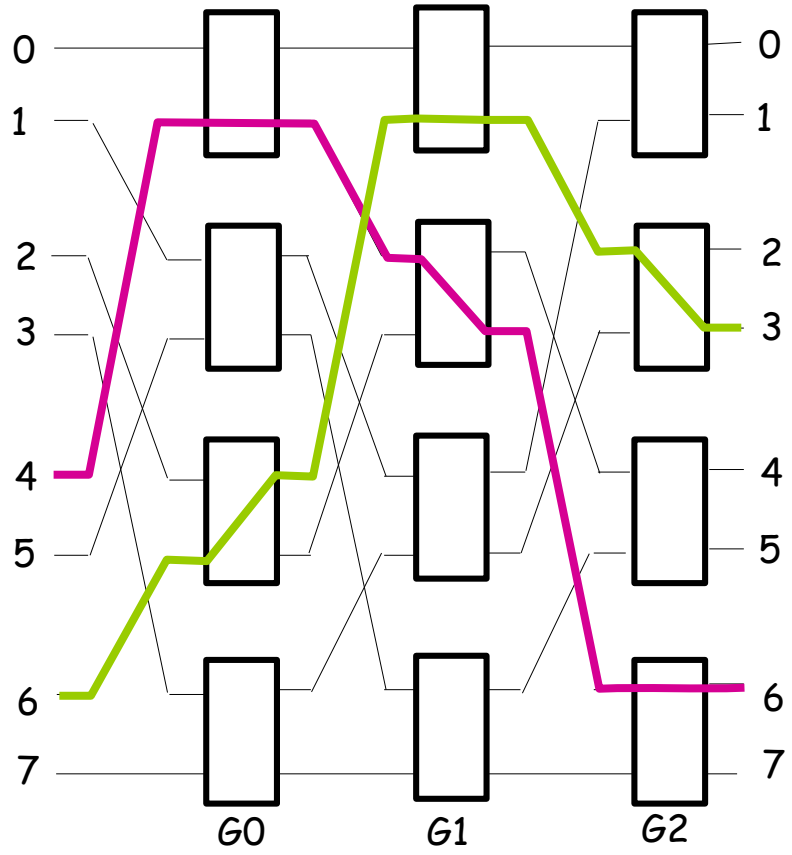
- Selección en origen
  - La cabecera del mensaje contiene la serie de puertos seleccionados
  - Utilizado y dividido en ruta: Cada conmutador usa el primer puntero y lo elimina de la lista.
  - Ejemplos: Meiko CS-2, Myrinet, MIT Artic



- Dirigido por tablas
  - Cada conmutador contiene una tabla de rutado, R
  - La cabecera del mensaje contiene un índice, i., que determina la entrada de la tabla de rutado que especifica el puerto de salida.
    - Out_port = R[i] ; (R[i] = entrada i-ésima de tabla de rutado)
  - Generalmente la tabla también proporciona el índice, i', para el siguiente switch
    - (Out_port, i') = R[i]
  - Ejemplo: ATM

MIN: Uso de la dirección de destino

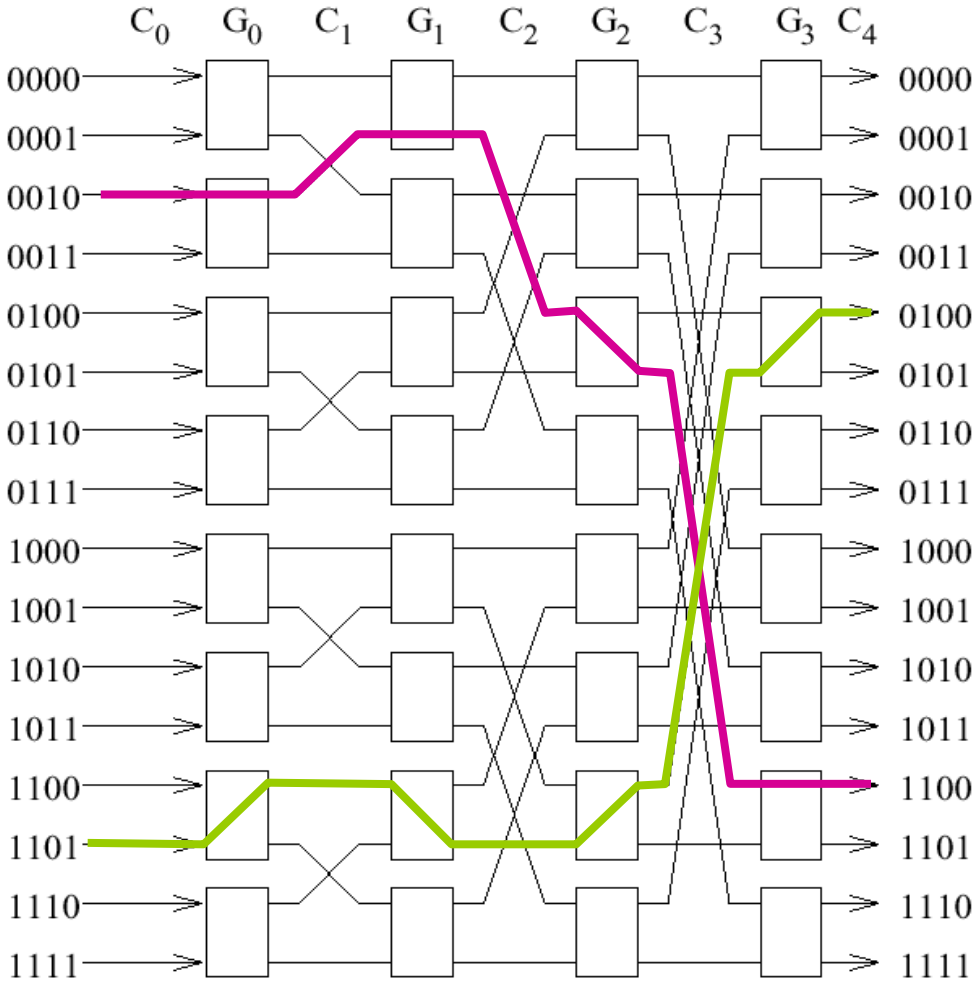
Red Omega



- Funcionamiento:
  - Etapas: n=3
  - Dir destino: (d2,d1,d0)
  - El bit di se usa para seleccionar el funcionamiento de los conmutadores de la etapa n-i-1
    - di = 0 ➔ Salida superior
    - di = 1 ➔ Salida inferior
- Ejemplo 1: paquete de 4 a 6
  - La dir destino es (110)
    - d2=1 ➔ Salida inferior en etapa 0
    - d1=1 ➔ Salida inferior en etapa 1
    - d0=0 ➔ Salida superior en etapa 2
- Ejemplo 2: paquete de 6 a 3
  - La dir destino es (011)
    - d2=0 ➔ Salida superior en etapa 0
    - d1=1 ➔ Salida inferior en etapa 1
    - d0=1 ➔ Salida inferior en etapa 2
- La configuración de los conmutadores no depende del nodo de entrada

MIN: Uso de la dirección destino

Red Butterfly



- Funcionamiento:
  - Etapas: n=4
  - Dir destino: (d3,d2,d1,d0)
  - El bit di se usa para seleccionar el funcionamiento de los conmutadores de la etapa i-1 (d0 determina etapa n-1)
    - di = 0 → Salida superior
    - di = 1 → Salida inferior
- Ejemplo 1: paquete de 2 a 12
  - La dir destino es (1100)
    - d1=0 → Salida superior en etapa 0
    - d2=1 → Salida inferior en etapa 1
    - d3=1 → Salida inferior en etapa 2
    - d0=0 → Salida superior en etapa 3
- Ejemplo 2: paquete de 13 a 4
  - La dir destino es (0100)
    - d1=0 → Salida superior en etapa 0
    - d2=1 → Salida inferior en etapa 1
    - d3=0 → Salida superior en etapa 2
    - d0=0 → Salida superior en etapa 3

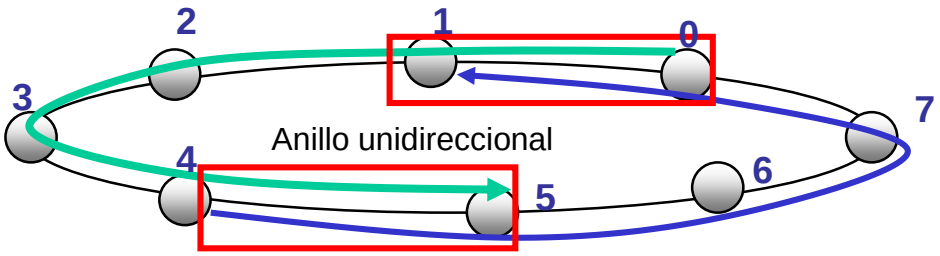
Interbloqueo: evitación

- Interbloqueo: ¿Cómo puede producirse?
  - Condiciones necesarias:
    - Recurso compartido
    - Adquirido incrementalmente
    - Sin mecanismo de expulsión
    - Los agentes en espera forman una cola circular
  - Canales como recurso compartido adquirido incrementalmente
    - Buffer fuente unido a buffer destino
    - Canales forman una ruta
- ¿Cómo puede evitarse?
  - Restringiendo cómo se adquirieren los canales
    - Ejemplo: orden dimensional
  - Aportando recursos adicionales
    - Canales virtuales

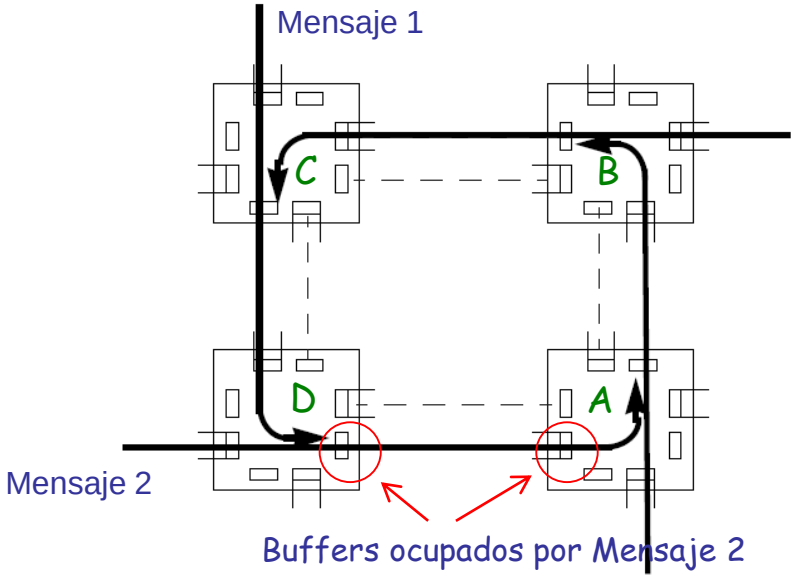
Interbloqueo: evitación

- Ejemplos de situaciones de interbloqueo

1)



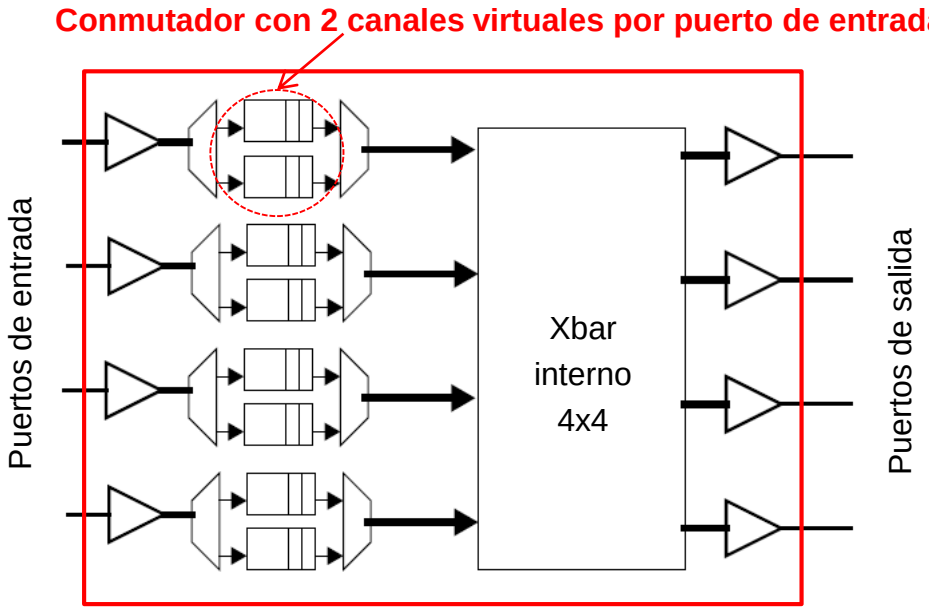
2)



Sup comunicaciones en curso:
A➔C; B➔D; C➔A; D➔B

Interbloqueo: evitación

- Canales virtuales:
  - Mecanismo básico para construir redes libres de interbloqueos con encaminamiento worm-hole
    - Back-pressure puede implicar la reserva incremental de muchos enlaces en la red
  - Técnica: asociar con cada enlace físico más de un buffer de entrada, que se comportan cómo canales adicionales (virtuales)
    - Proporcionan una vía de escape en situaciones que provocarían interbloqueo
    - No aumenta el nº de enlaces en la red
    - No aumenta el tamaño del crossbar interno del conmutador: los buffers asociados a cada canal físico, están conectados al crossbar interno mediante mux



Interbloqueo: evitación

- Canales virtuales: ejemplo malla 2D
  - Sup encaminamiento: en caso de congestión del canal "natural" (orden dimensional), se puede usar un canal alternativo.
    - Ventaja: mejor aprovechamiento de los recursos de la red
    - Problema: posibilidad de interbloqueo (ver tr. 73)
    - Solución: usar dos canales virtuales (Hi, Lo) por enlace físico
    - Un paquete que entra por Hi/Lo usa el mismo canal virtual (Hi/Lo) en el siguiente salto, excepto si hace un giro norte➔oeste, en cuyo caso usa el canal virtual contrario (Lo/Hi)

Sup comunicaciones en curso:

A➔C; B➔D; C➔A; D➔B

Sin CV: interbloqueo

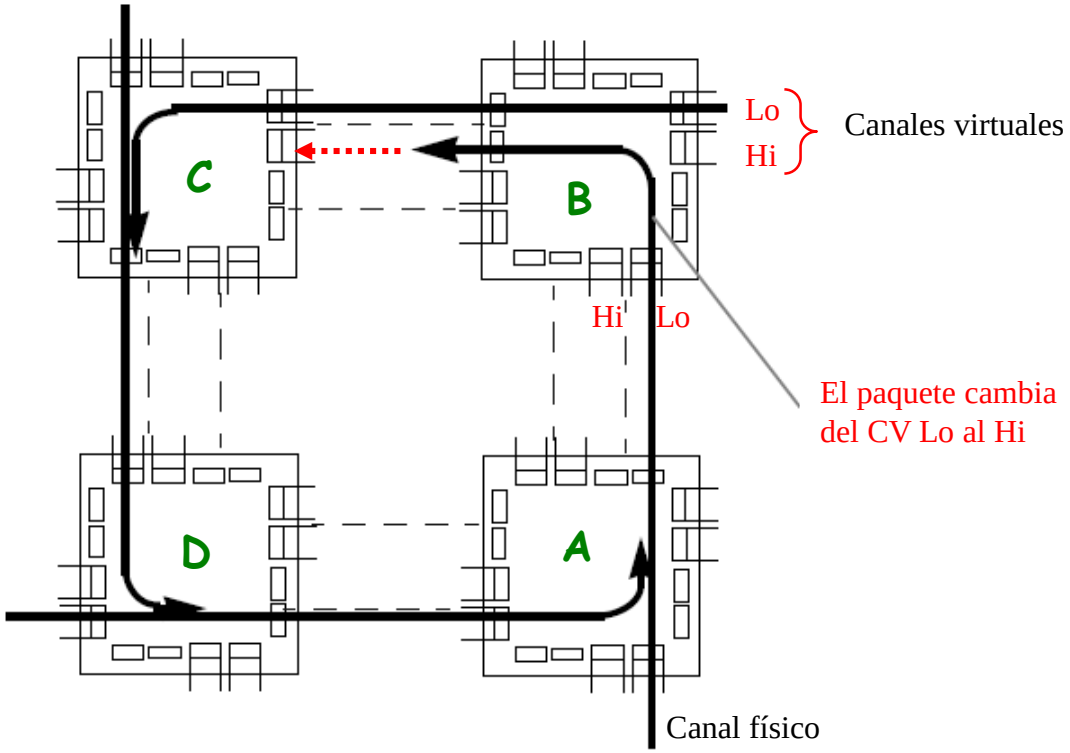
Con CV:

A alcanza C, libera A-B

D alcanza B, libera D-A

C alcanza A, libera C-D

B alcanza D, libera B-C



Ejemplos: topología, routing, conmutación en sistemas comerciales

Company	System [network] name	Max. number of nodes [× # CPUs]	Basic network topology	Switch queuing (buffers)	Network routing algorithm	Switch arbitration technique	Network switching technique
Intel	ASCI Red Paragon	4510 [× 2]	2D mesh (64 × 64)	Input buffered (1 flit)	Distributed dimension-order routing	2-phased RR, distributed across switch	Wormhole with no virtual channels
IBM	ASCI White SP Power3 [Colony]	512 [× 16]	Bidirectional MIN with 8-port bidirectional switches (typically a fat tree or Omega)	Input and central buffer with output queuing (8-way speedup)	Source-based LCA adaptive, shortest-path routing, and table-based multicast routing	2-phased RR, centralized and distributed at outputs for bypass paths	Buffered wormhole and virtual cut-through for multicasting, no virtual channels
Intel	Thunder Itanium2 Tiger4 [QsNet^{II}]	1024 [× 4]	Fat tree with 8-port bidirectional switches	Input buffered	Source-based LCA adaptive, shortest-path routing	2-phased RR, priority, aging, distributed at output ports	Wormhole with 2 virtual channels
Cray	XT3 [SeaStar]	30,508 [× 1]	3D torus (40 × 32 × 24)	Input with staging output	Distributed table-based dimension-order routing	2-phased RR, distributed at output ports	Virtual cut-through with 4 virtual channels
Cray	X1E	1024 [× 1]	4-way bristled 2D torus (~23 × 11) with express links	Input with virtual output queuing	Distributed table-based dimension-order routing	2-phased wavefront (pipelined) global arbiter	Virtual cut-through with 4 virtual channels
IBM	ASC Purple pSeries 575 [Federation]	>1280 [× 8]	Bidirectional MIN with 8-port bidirectional switches (typically a fat tree or Omega)	Input and central buffer with output queuing (8-way speedup)	Source and distributed table-based LCA adaptive, shortest-path routing, and multicast	2-phased RR, centralized and distributed at outputs for bypass paths	Buffered wormhole and virtual cut-through for multicasting with 8 virtual channels
IBM	Blue Gene/L eServer Solution [Torus Net.]	65,536 [× 2]	3D torus (32 × 32 × 64)	Input-output buffered	Distributed, adaptive with bubble escape virtual channel	2-phased SLQ, distributed at input and output	Virtual cut-through with 4 virtual channels