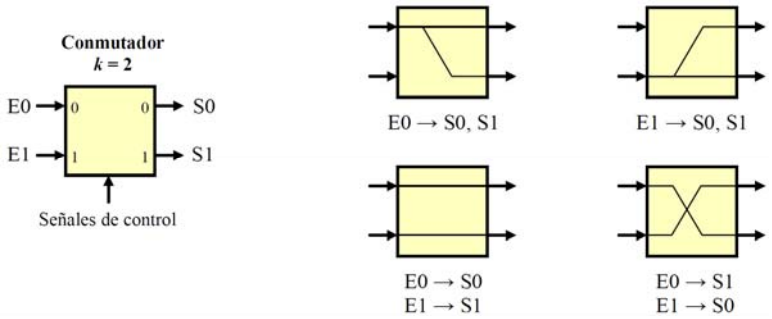
REDES FORMADAS POR CONMUTADORES

La función de la red de comunicación de un sistema paralelo es permitir la comunicación entre procesadores (o entre procesadores y memoria), es decir, transmitir datos. Las redes de transmisión de datos se han utilizado desde hace mucho tiempo en otras áreas tecnológicas, más en concreto en la telefonía. Por ello, se ha aprovechado la experiencia acumulada durante años en esa área y se ha adaptado el uso de las redes más habituales en telefonía, redes construidas a partir de conmutadores, a los sistemas de cómputo paralelo. A estas redes se les conoce también como redes dinámicas, porque el camino de comunicación entre origen y destino se construye dinámicamente (no existen enlaces fijos entre los nodos). Analicemos en qué consisten y cómo se usan estas redes.

1.- El conmutador (switch)

Comencemos describiendo qué es un conmutador, "dispositivo" que sirve para construir redes dinámicas. El conmutador más simple, de grado $k = 2$, es un dispositivo que conecta dos entradas (E0 y E1) y dos salidas (S0 y S1). Mediante él, y en función de una señal de control, se pueden establecer las siguientes cuatro conexiones:



Por medio de las dos primeras conexiones, la información de una de las entradas se distribuye a ambas salidas (ese tipo de comunicación se conoce como *broadcast*). Sin embargo, las conexiones que más nos interesan son las otras dos, en las que se conecta cada entrada con una salida, seguidas o cruzadas, para transmitir información.

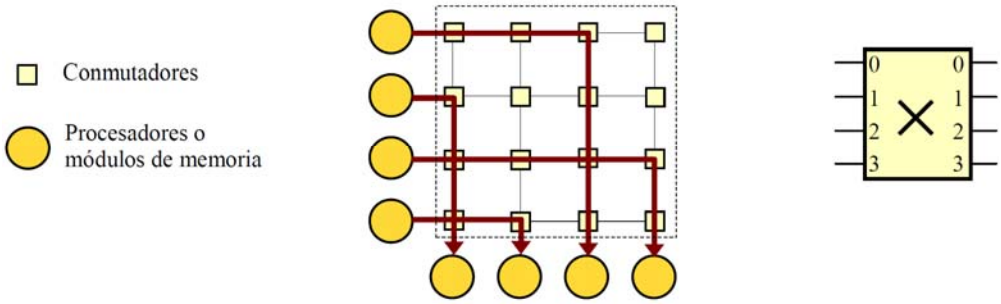
Visto desde otro punto de vista, mediante un conmutador se consigue una "permutación" de las entradas en las salidas. Por ejemplo, con un conmutador de grado 2 podemos establecer las dos siguientes conexiones: (0, 1) → (0, 1) o (0, 1) → (1, 0), es decir, las dos permutaciones de las entradas.

Las conexiones que se crean en cada conmutador son "dinámicas", ya que van a ir cambiado en el tiempo en función de las necesidades de comunicación.

2.- Red *crossbar*

Tal y como hemos comentado anteriormente, la red de comunicación ideal sería la que permitiera, en cualquier momento y en un solo "paso", que se comunicaran cualquier par de procesadores simultáneamente. Según lo hemos definido, un conmutador de P entradas cumple con esa condición: permite efectuar P conexiones simultáneas. A esta red se la denomina también *crossbar*.

Un *crossbar* puede construirse de muchas maneras, generalmente mediante conmutadores de menor grado. En la figura se muestra un ejemplo de implementación de un *crossbar* que conecta cuatro procesadores entre sí, o con cuatro módulos de memoria, construido mediante conmutadores de grado 2. Cada conmutador permite conectar una fila con una columna, y ofrece la posibilidad de seguir o girar.



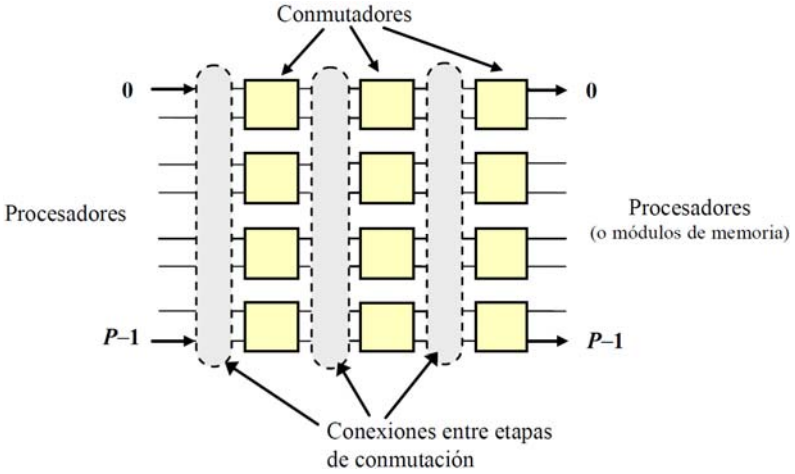
Tal como aparece en la figura, esta estructura de conmutadores permite efectuar simultáneamente *P* comunicaciones (cualesquiera, siempre que los destinos sean todos diferentes). Sin embargo, el coste de esta red es muy alto si el número de procesadores es elevado, ya que el número de conmutadores y de conexiones (la complejidad de la red) es del orden de P^2; además, el control tiene que ser centralizado.

Las redes de tipo *crossbar* se han venido utilizando, normalmente, con un número pequeño de procesadores, aunque en ocasiones también en sistemas con un número de procesadores elevado (por ejemplo, en el computador Earth Simulator), a veces con conmutadores organizados en varias etapas.

3.- Redes multietapa (*multistage*)

El coste de los conmutadores crece cuadráticamente con el número de entradas. Por tanto, para reducir el coste de la red (para usar menos hardware) es necesario organizar la red de otra manera, aunque con ello no obtengamos la misma capacidad de comunicación y latencia reducida que podemos conseguir con un *crossbar*.

Las redes dinámicas más utilizadas son las denominadas redes multietapa, en las que las conexiones se establecen mediante conmutadores organizados por niveles o etapas, tal y como se muestra en la siguiente figura. Las conexiones entre etapa y etapa se pueden definir de múltiples maneras; en función del esquema o patrón de conexionado se obtienen diferentes tipos de redes: *perfect shuffle*, *butterfly*, *cube connection*, etc.

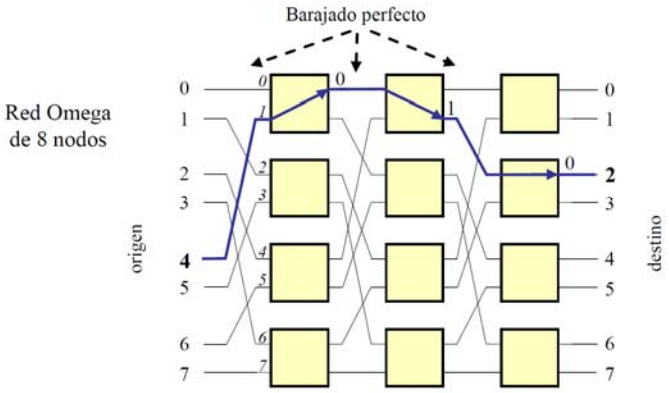


La red del dibujo es unidireccional (de izquierda a derecha); si se necesita que sea bidireccional (p.e., para el caso de conexiones con módulos de memoria) basta con superponer dos redes unidireccionales.

3.1.- La red Omega

Como ejemplo de las redes multietapa veamos una de las más utilizadas: la red Omega. Una red Omega con P entradas consta de $\log_k P$ niveles o etapas de conmutación, cada una con P/k conmutadores. Por tanto, el número de conmutadores de una red Omega de grado k es $(P/k)\ \log_k P$, mucho menor que el de un *crossbar*. Por ejemplo, un *crossbar* de 256 nodos utiliza $256^2 = 65536$ conmutadores de grado 2, mientras que la red Omega correspondiente sólo utiliza $128 \times 8 = 1024$; por contra, la latencia de las conexiones es mayor en la red Omega, ya que hay que superar 8 etapas de conmutación (una sola en el *crossbar*).

Las conexiones entre las etapas de conmutación siguen un esquema denominado barajado perfecto (*perfect shuffle*), tal y como se muestra en la figura (con conmutadores de grado 2).



El esquema de conexionado denominado barajado perfecto es muy simple: para el caso de grado 2, las P entradas (de 0 a P–1) se dividen en dos grupos por la mitad, y se reordenan de la siguiente manera: una de la primera mitad (0), otra de la segunda mitad (P/2); una de la primera mitad (1), otra de la segunda mitad (P/2+1); etc. Es decir, se efectúa la siguiente permutación:

$$[0, 1, 2... P–1] \rightarrow [0, P/2, 1, P/2+1, ... P/2–1, P–1].$$

En el caso general de grado k, se dividen en k grupos y se reordenan de la misma manera que acabamos de comentar. Se puede comprobar que la permutación de barajado perfecto se corresponde con una rotación hacia la izquierda de la dirección binaria del origen. La tabla siguiente muestra dicha rotación para el caso de 8 procesadores.

Barajado perfecto (*perfect shuffle*)

posición antigua			nueva posición	
(0)	000	→	(0)	000
(1)	001	→	(2)	010
(2)	010	→	(4)	100
(3)	011	→	(6)	110
(4)	100	→	(1)	001
(5)	101	→	(3)	011
(6)	110	→	(5)	101
(7)	111	→	(7)	111

Analicemos los parámetros topológicos de una red Omega. La distancia media y el diámetro coinciden: todos los procesadores están a la misma distancia, $\log_k P$, el número de etapas de la red. Por tanto, desde el punto de vista topológico la latencia de todos los mensajes sería la misma: no se puede aprovechar la "localidad" de las comunicaciones (la comunicación suele ser más frecuente con los procesadores que están más "cerca").

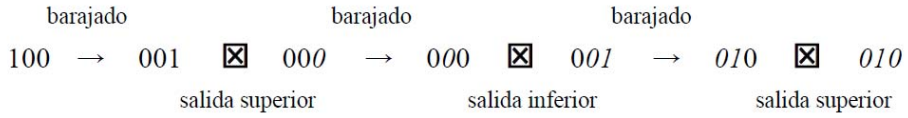
La red es simétrica y regular. El grado de los conmutadores es fijo y el mismo para todos (puede ser 2, como en la figura, pero son más habituales los conmutadores de grado 4). La red no tiene tolerancia a fallos; tal como vamos a ver a continuación, sólo existe un camino que comunique el nodo *i* con el nodo *j*, por lo que si algo falla en dicho camino no es posible establecer esa comunicación. En algunos casos, si no se rompe la red, una comunicación entre *i* y *j* que no se puede establecer puede dividirse en dos fases: se manda un mensaje de *i* a *k*, y se reenvía luego de *k* a *j* (aunque, obviamente, la latencia total será mucho mayor). Si no, la única opción para hacer frente a fallos en la red consiste en replicar el hardware (p.e. duplicar los conmutadores, para que si uno de ellos falla podamos usar el otro).

3.2.- Encaminamiento en la red Omega

La finalidad de una red de comunicación es, evidentemente, permitir la comunicación entre los procesadores. Por ello, es importante que se pueda conectar el origen y el destino de una manera fácil. Pero, ¿cómo se construye una conexión (un camino) entre dos nodos de la red? A la construcción (o elección) de un camino determinado se le denomina encaminamiento (routing). En las redes Omega el encaminamiento es bastante simple.

Si analizamos el funcionamiento de los conmutadores (k = 2) y de la red Omega, se puede ver que cuando se elige una salida en cada conmutador se modifica el último bit de la dirección (posición): a 0, si se elige la salida de arriba, o a 1, si se elige la de abajo. Además, en cada uno de los barajados que se hace entre las etapas de conmutación se rota un bit de la dirección. Por ello, para llegar al destino basta tener en cuenta la dirección de destino: la salida escogida en cada nivel de conmutación se va correspondiendo con los bits de la dirección de destino, comenzando por el bit de más peso.

Veamos un ejemplo (el de la figura anterior). Para ir desde el procesador 4 (**100**) al 2 (**010**), hay que elegir el siguiente camino en los conmutadores: arriba (0), abajo (1) y arriba (0).



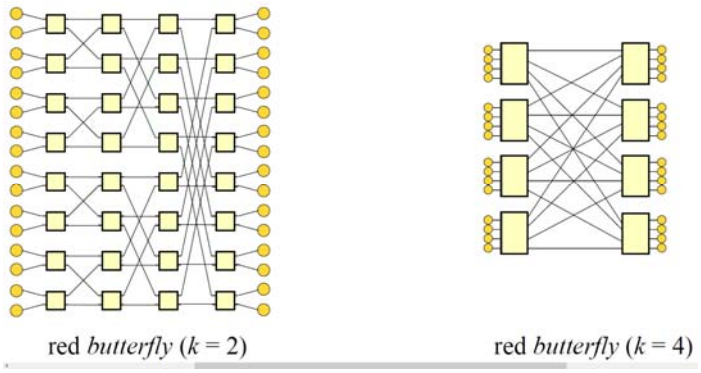
Otra alternativa para seleccionar el camino es calcular la función `xor` entre las direcciones del origen y del destino. El resultado, denominado *routing record* o registro de encaminamiento (*RE*), se debe utilizar de la siguiente manera: en cada etapa, se procesa un bit del registro de encaminamiento, comenzando por el de más peso; si es 0, se prosigue sin cruzar; en cambio, si es 1, se cruza dentro del conmutador, eligiendo la salida contraria.

Para el ejemplo anterior, 4 → 2: *RE* = 100 `xor` 010 = 110. Por tanto, en las dos primeras etapas cruzar, y en la tercera no cruzar.

Sea utilizando la dirección de destino o el registro de encaminamiento, es muy sencillo encontrar el camino para ir del nodo *i* al nodo *j* en una red Omega. Sin embargo, existe sólo un camino para ir de un nodo a otro, lo cual no resulta muy favorable si se quiere poder hacer frente a fallos de funcionamiento de la red (o a situaciones de tráfico elevado).

3.3.- Otras redes

Podríamos enumerar muchos otros ejemplos de redes multietapa que se han usado en multiprocesadores. Entre las más conocidas está la denominada red *butterfly* (mariposa). En la figura se presenta un esquema de una red *butterfly* de 16 nodos, con conmutadores de grado 2 o de grado 4.



Las redes *butterfly* y Omega son muy similares; por ejemplo, puede comprobarse fácilmente que puede usarse en ambas el mismo algoritmo de encaminamiento, el que utiliza la función `xor` para generar el registro de encaminamiento.

Las redes Omega y *butterfly*, son redes bloqueantes; por su parte, un *crossbar* es una red no bloqueante (admite cualquier comunicación simultánea), pero su coste es muy elevado. En todo caso, es posible construir redes no bloqueantes con un coste menor que el del *crossbar*; ese tipo de redes se conoce, de manera general, como **redes de *Clos***. Entre las redes no bloqueantes se encuentran las redes "reorganizables" (*rearrangeable*), en las que siempre es posible encontrar un camino para cualquier comunicación, aunque en algunas ocasiones es necesario reorganizar los caminos existentes en un momento dado. Un ejemplo de ese tipo de redes son las **redes de *Benes***, que permiten cualquier permutación si se conoce de antemano cuál es (para elegir los caminos adecuados), ya que hay muchos caminos para conectar origen y destino.

La siguiente figura presenta una red de *Benes* de 16 nodos, que utiliza 7 etapas de conmutación (en las redes de *Clos* el número de etapas de conmutación es siempre un número impar). La red permite construir múltiples caminos para unir dos nodos; por ejemplo, hemos dibujado dos caminos diferentes para ir del nodo P3 al P11. Como es obvio, el coste de esta red es mayor que el de una red Omega, y la latencia de los paquetes será también mayor, ya que tienen que superar más etapas de comunicación.

Si "doblamos" una red de Benes sobre sí misma, la red que se consigue se denomina árbol (*fat tree*), en la que los paquetes van hacia adelante (hasta la raíz) y luego vuelven hacia atrás para llegar al destino (es decir, los enlaces y puertos son bidireccionales).