

Interconnection Networks

There are two basic classes of network

- Direct (static)
- Indirect (dynamic)

Traditionally direct networks are used for distributed memory machines and indirect networks are used for shared memory. **THERE IS NO REASON FOR THIS OTHER THAN ENGINEERING HISTORY.** In fact, both types of networks are now used for both types of architectures.

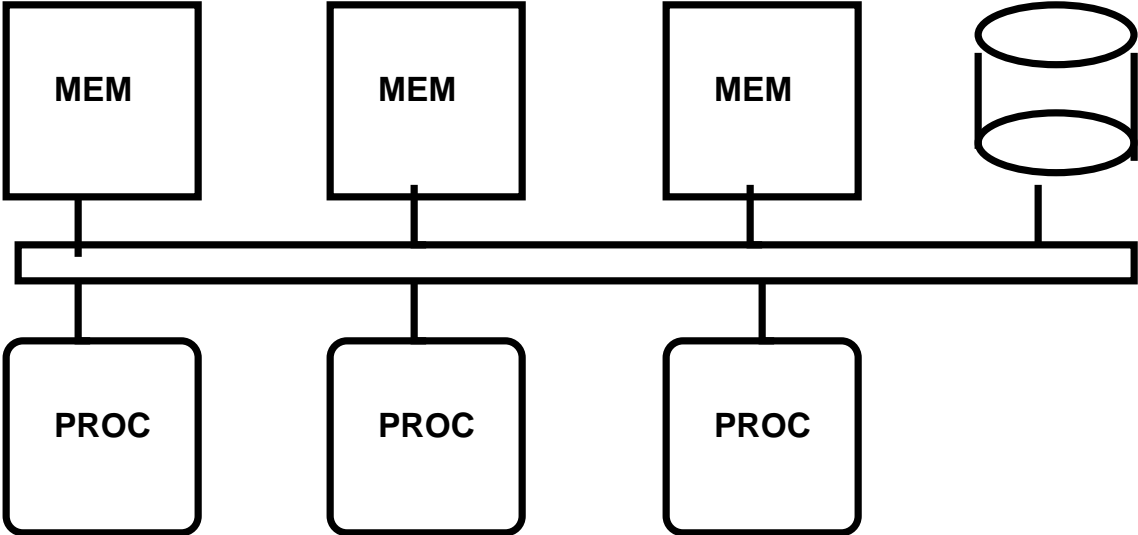
For example, the Cray T3E is a 3-D toroidal mesh with direct memory access (incoherent physically shared memory).

Most networks are designed to address engineering trade-offs and not architectural dogmatic purity so there is still a gap between practice (where details make the decision) and theory (where isomorphism, embedding and theoretical performance are dominant)

Bus interconnection

- simplest and most often used.
- multiple resources (often very heterogeneous) compete to control single communication pathway seen by all.
- broadcast with selective read
- used at many different levels of the system, often as a building block in combination with other interconnects

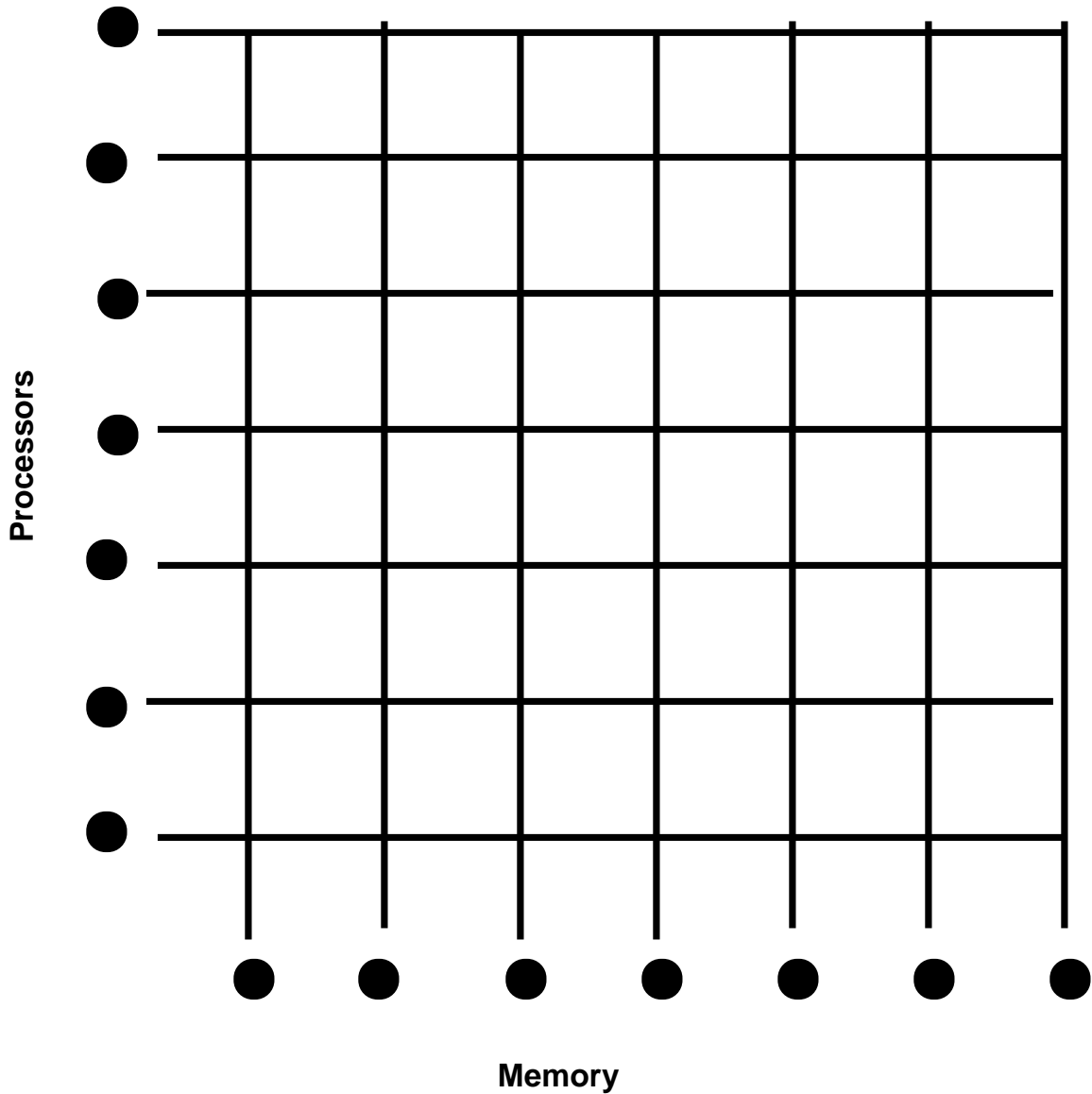
BUS INTERCONNECTION



Crossbar

- a bus is essentially a sequential medium
- to have full connectivity between m sources and n destinations a crossbar can be used
- high cost – $O(mn)$ switches or gates
- very fast, but typically small – n in the biggest crossbar is on the order 400
- very often used as a component in a larger more complex network with other crossbars or busses

CROSSBAR INTERCONNECT



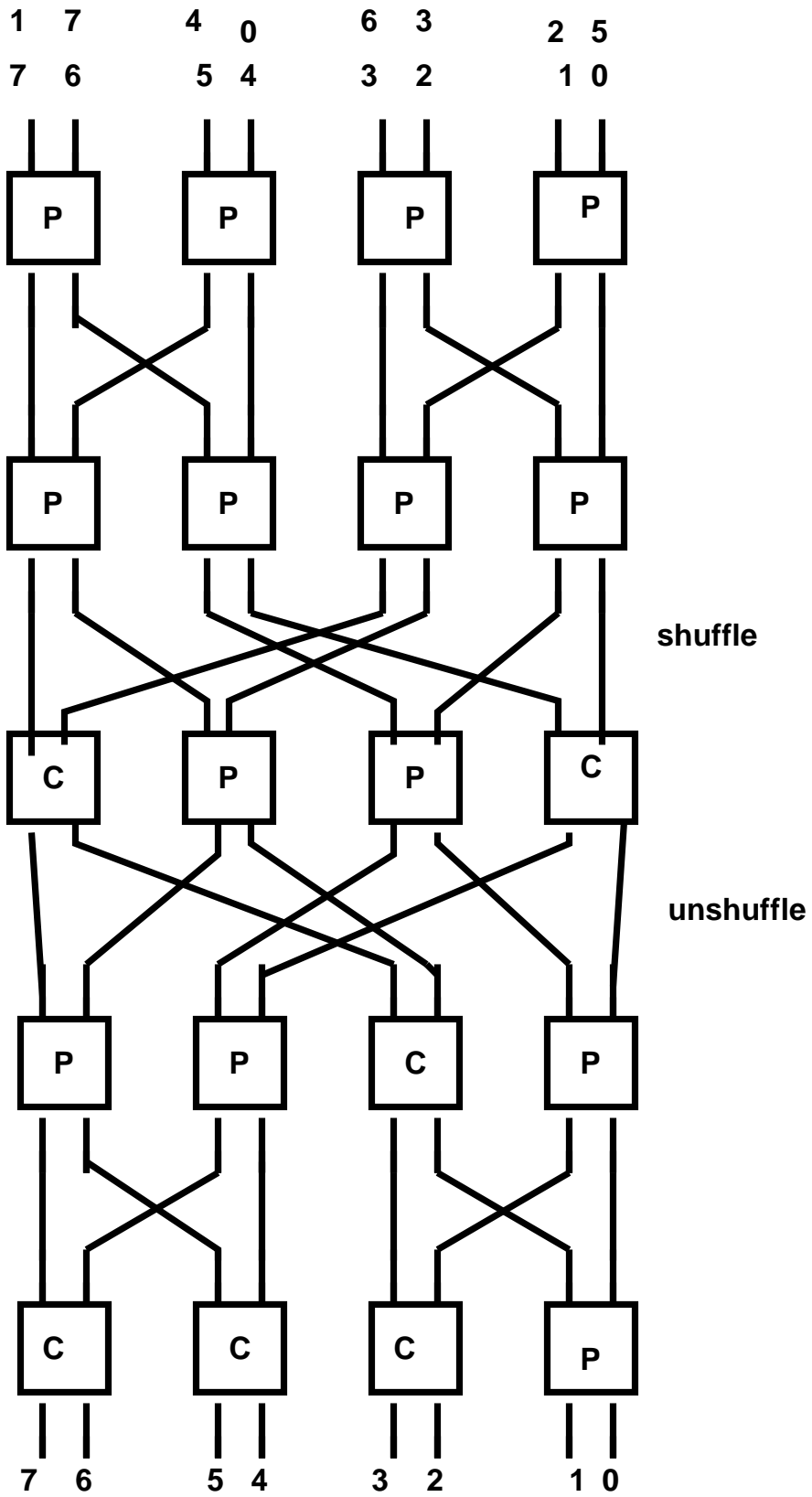
each crosspoint is an independent switch
various designs place restrictions on how
many can be set in each row and column
can also be built out of MUX-DEMUX's
still $O(n^2)$ complexity in gates

MULTISTAGE INTERCONNECTION NETWORKS

- An attempt to get full connectivity (or as near as possible) with less complexity (something close to $O(n \log n)$ in gates.
- substantial work done on networking for commercial reasons e.g. the phone company did most of the interesting stuff
- computer networks for processor memory interconnects require fast local routing and low latency and area
- related to the study of permutations and sorting

- Optimal Rearrangable Alignment Networks (ORAN's)
- many different types
- parameters:
 - gates (area)
 - latency (time, or depth)
 - control cost (setup of switches)
 - connectivity (all or some permutations)
- Clos-Benes networks are best example

Clos-Benes 8 by 8



- note multiple paths from i to j
- compare the identity settings to a nontrivial permutation
- complete permutations
- high control costs $O(n \log n)$ i.e. path for each depends on whole
- complexity $(n/2)(2 \log n - 1)$ 2×2 switches
- time $(2 \log n - 1)$ levels
- not acceptable in high-performance memory

Sorting Networks

- to insure full permutation connectivity consider how to sort in hardware a list of numbers.
- we first consider a special sequence and sorting circuit due to Batcher

Definition: A bitonic sequence is the juxtaposition of two monotonic sequences (one ascending and one descending). It remains bitonic if split and the parts interchanged.

Essentially if the sequence is viewed in a ring fashion and the magnitudes considered, it has no local minima or maxima only one maximum and one minimum.

$$\begin{aligned}
 a_i &= (1, 3, 5, 6, 12, 18, 17, 13, 11, 5, 4, 3) \\
 1 \leq i \leq 2n \\
 a_i &= (17, 13, 11, 5, 4, 3, 1, 3, 5, 6, 12, 18)
 \end{aligned}$$

Both are bitonic sequences since they are the same ring split in two different places.

Define $d_i = \min(a_i, a_{n+i})$ and $e_i = \max(a_i, a_{n+i})$. It does not matter which splitting you use, d_i and e_i will have the required properties.

$$\begin{aligned}
 d_i &= (1, 3, 5, 5, 4, 3) \\
 a_i &= (17, 13, 11, 5, 4, 3) \\
 a_{n+i} &= (1, 3, 5, 6, 12, 18) \\
 e_i &= (17, 13, 11, 6, 12, 18)
 \end{aligned}$$

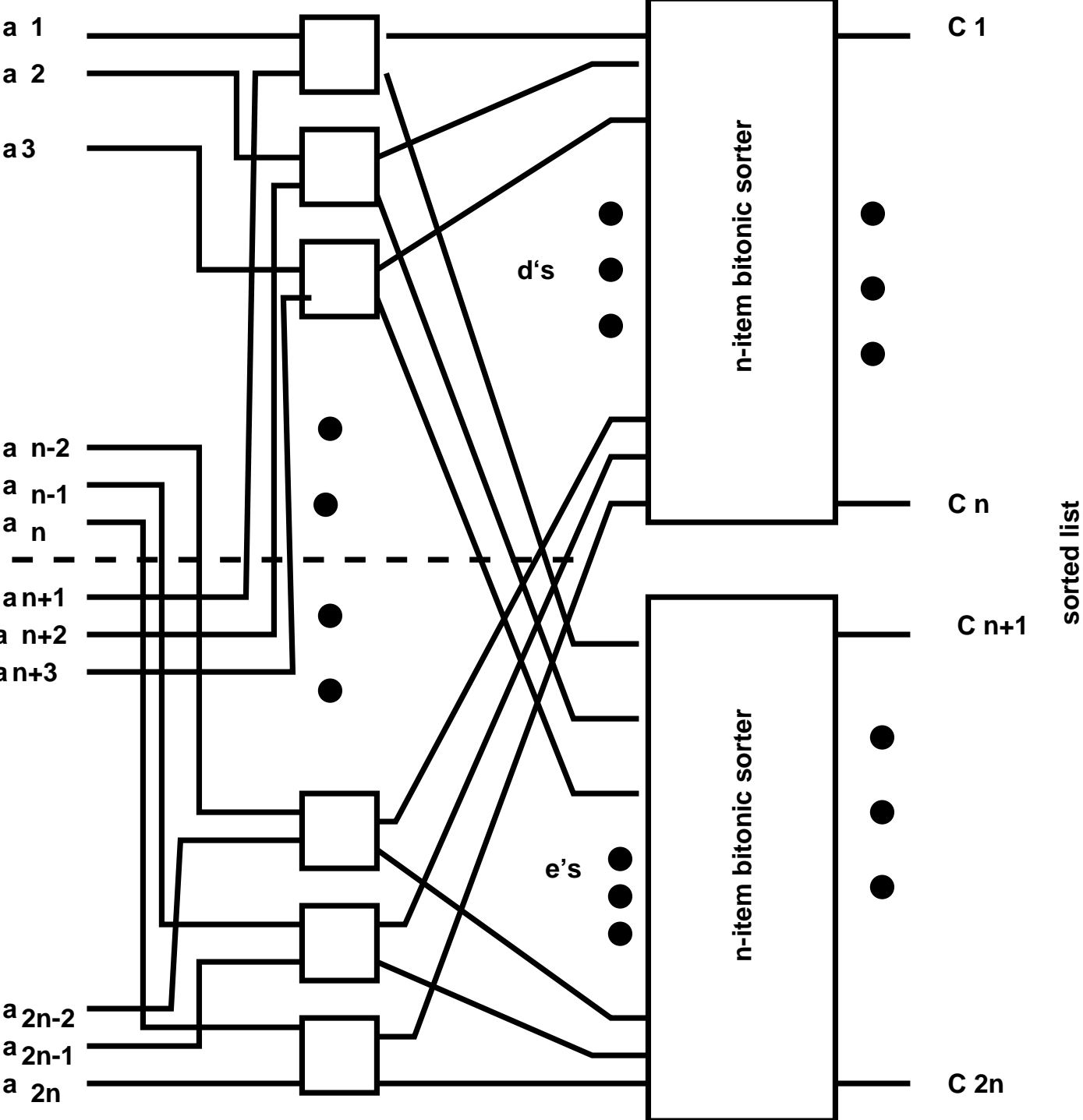
Note that the d_i and e_i generation is easily done via a 2 switch which looks for the leading one position and chooses the parallel or cross setting to output the two input values (a_i, a_{n+i}) in sorted order.

So what? The following theorem allows the construction of a bitonic sorter.

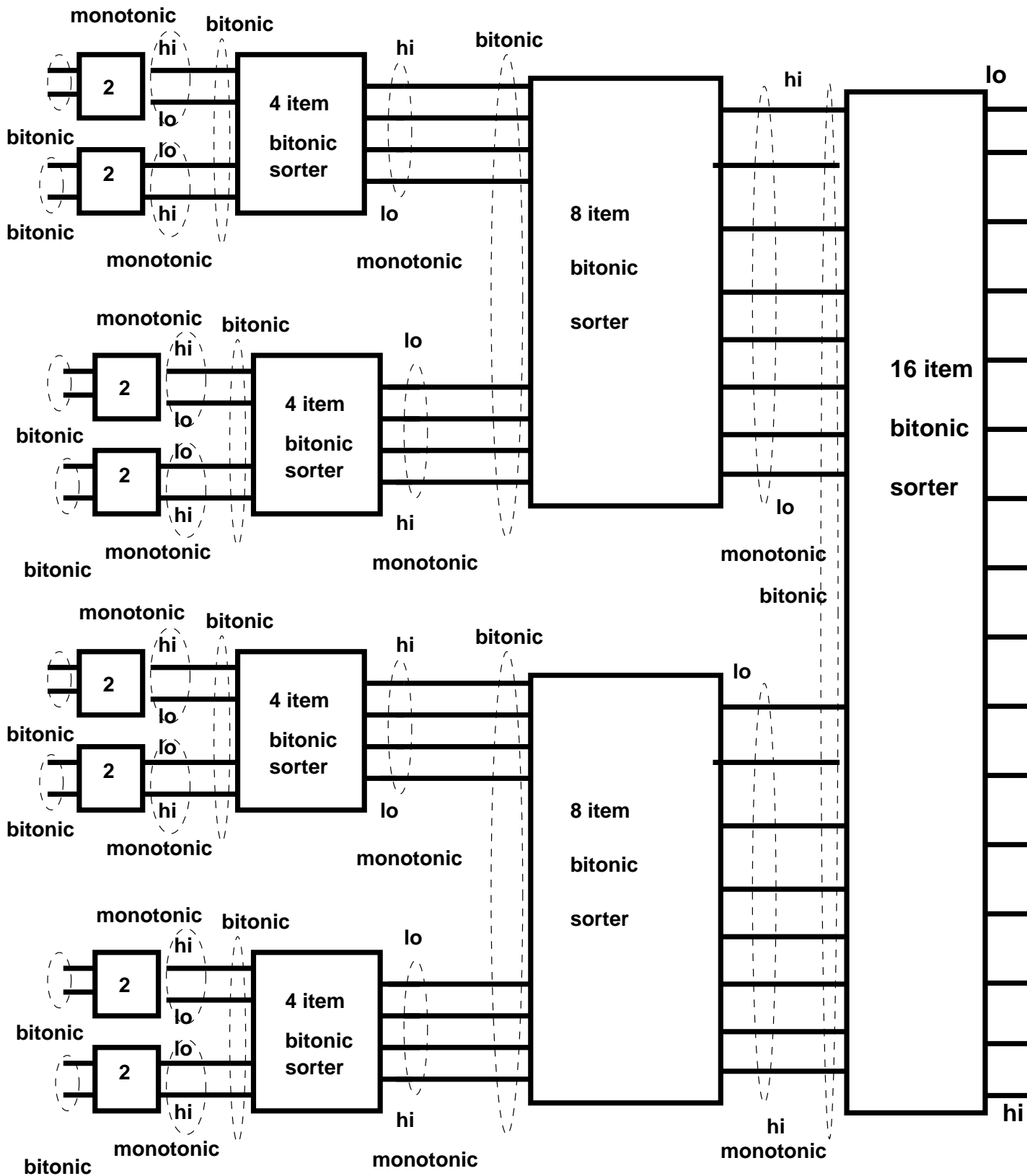
Definition: If $a_i, 1 \leq i \leq 2n$, is bitonic then so are d_i and $e_i, 1 \leq i \leq n$. Furthermore, $\max(d_1, \dots, d_n) \leq \min(e_1, \dots, e_n)$.

So we can split a_i into two sequences that are bounded appropriately to sort separately and that have the bitonic property so we can apply the theorem recursively $\log n$ times to get a completely sorted sequence.

2*n Bitonic Sequence Sorter



- A bitonic sorter on a sequence of size $m = 2n = 2^p$
- time (depth) is $\log m$
- gates (area) is $(m/2) \log m$
- can be used to get a general sorter

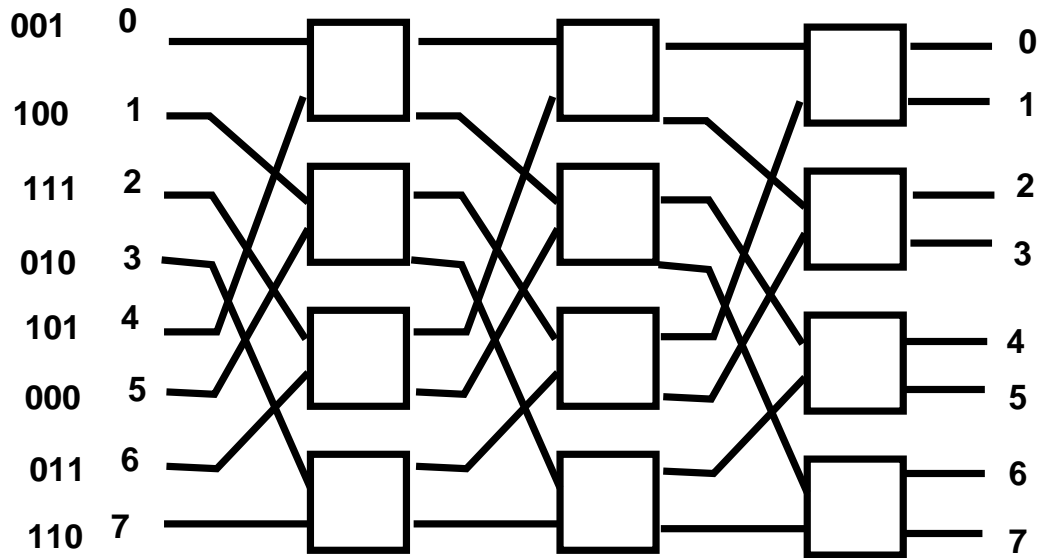


- NOTE THE ALTERNATION IN ASCENDING AND DESCENDING SEQUENCES (THIS ENTAILS SIMPLE ALTERATIONS IN CONNECTIVITY)
- gates (area) $O(m \log^2 m)$
- time (depth) $O(\log^2 m)$
- control is local since it is essentially leading ones detection on two inputs – $O(\log^2 m)$
- to do better we must sacrifice some connectivity

Omega Networks

- supports all main permutations – shifts, reversals etc. – but not all
- as a result BLOCKING POSSIBLE
- less delay $O(\log m)$
- acceptable area $O(m \log m)$
- easy local routing decision
- very popular with higher degree crossbars as basic switch
- simple pattern of repeated shuffles

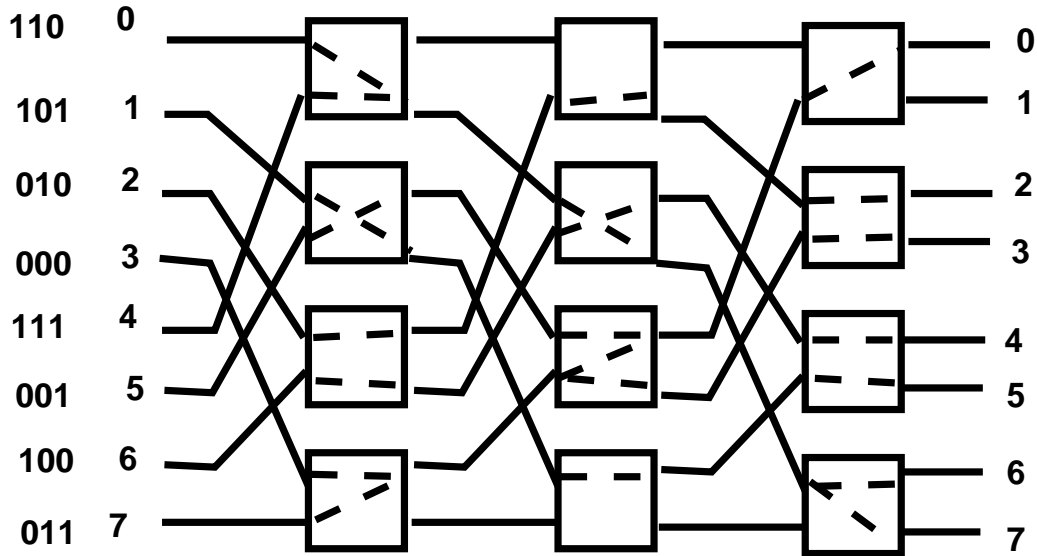
8 by 8 Omega network using 2 by 2 crossbars



Routing algorithm uses binary pattern of destination:

use bit i on stage i, (leftmost bit first)

8 by 8 Omega network using 2 by 2 crossbars



Blocking is possible as seen on this permutation which is blocked at marked gates

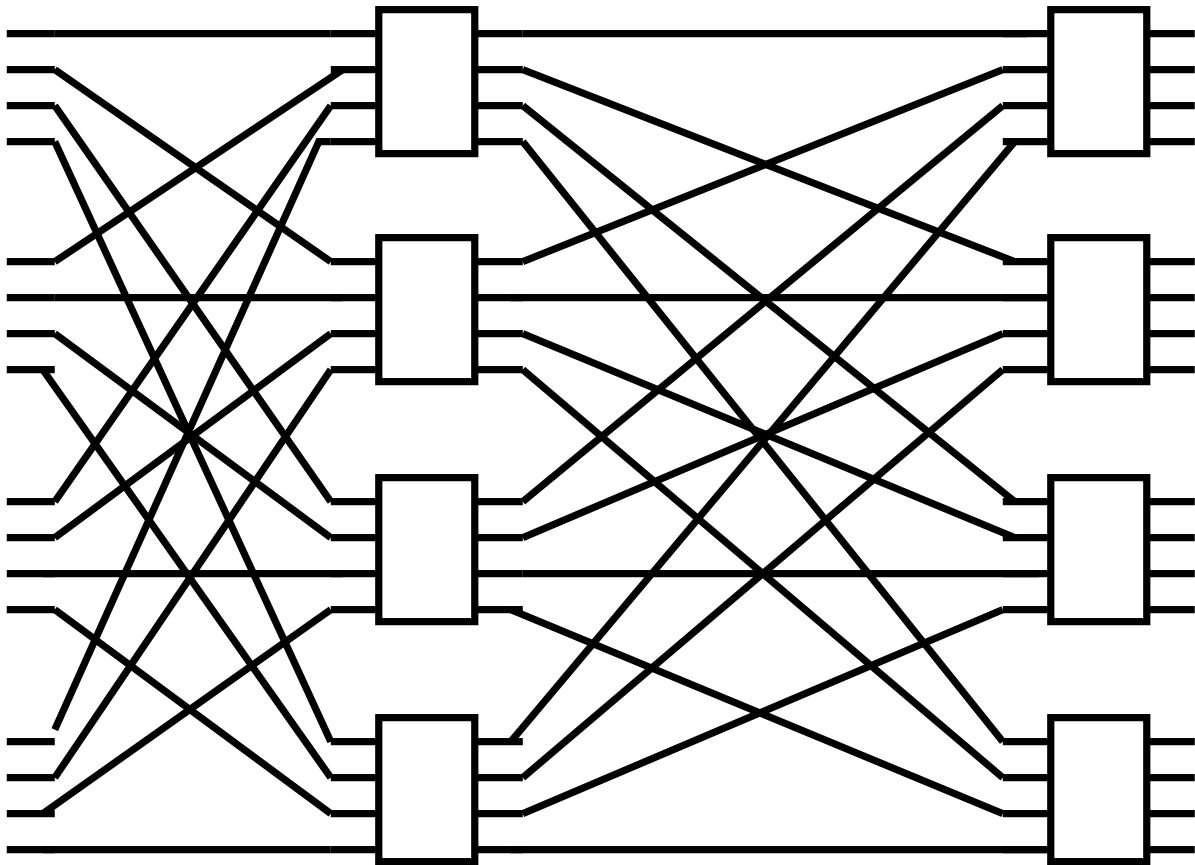
- Higher order switches may be used – requires the use of b -way shuffles where b is the order of the switch.
- In reality, the system does not operate in lock step with permutations
- each stage must have queues, throttle mechanisms, and arbitration strategies to handle blocks and traffic flow
- shared memory traffic tends to assume small packet sizes are normal – address on forward network and address and data on reverse network for a read, address and data on forward network and address (acknowledge) on reverse network for a write.

- sometimes multiple words are taken from each bank to exploit spatial locality
- modeling of contention and latency and their effects on performance are key basis for algorithm design and data layout
- key model issue: number of hops versus wire length dominating model
- latency mitigation is the main design issue (prefetch and multiple outstanding requests to memory)
- deadlock is not possible since there is always someone taking traffic off the network, i.e., simple source to drain model

16 by 16 Omega network using

4 by 4 crossbars

two 4way whuffles



Direct or Static Networks

- Typically associated with distributed memory machines.
- Operational Characteristics:
 - Topology - assumed static here
 - Timing protocol - synchronous or asynchronous
 - Switching method - circuit or packet
 - Control strategy - centralized or distributed
- Performance criteria:
 - Functionality - support for various operations
 - Network latency - worst case time for unit message
 - Bandwidth - maximum data transfer rate
 - Hardware complexity - cost of implementation
 - Scalability - ease of expansion

Interconnects: Evaluation

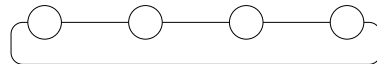
- Diameter: maximum distance between any two processors. The distance is the least number of links (hops) that need to be traversed between processors
- Arc connectivity: connectivity is a measure of the multiplicity of paths between processors. Arc connectivity is the minimum number of links that need to be removed in order to break the network into two disjoint parts; it is a measure of connectivity. High connectivity (and thus arc connectivity) is desirable for avoiding contention
- Bisection width: the minimum number of links that need to be removed in a network to separate the processors into two halves. Bisection width is a measure of the volume of traffic that can be handled by the network

Interconnects: Ring

- Each processor connected to 2 other processors
- Diameter of a ring is $\lfloor \frac{p}{2} \rfloor$
- Arc connectivity of a ring is 2
- Bisection width of a ring is 2



(a)

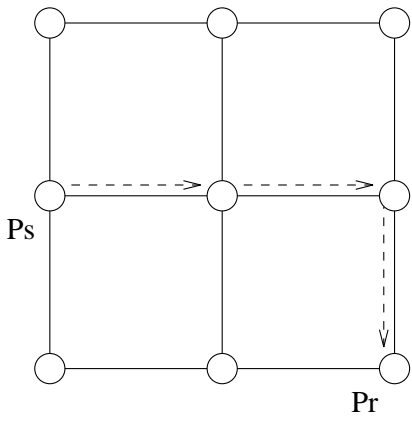


(b)

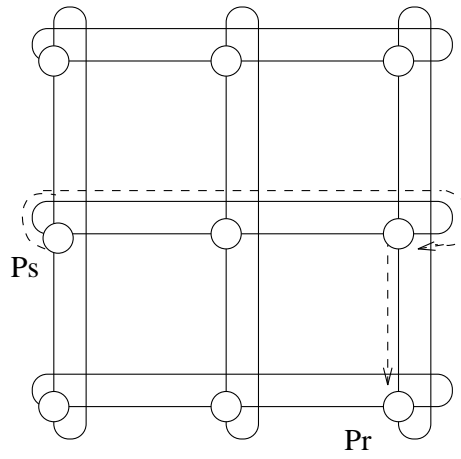
Interconnects: Multidimensional Meshes

- Each processor in an d dimensional mesh is connected to $2d$ other processors except for the corner processors
- In practice, only 2 or 3 dimensional meshes are constructed
- Mesh with wrap around - torus
- Processors can be increased without increasing dimension of mesh

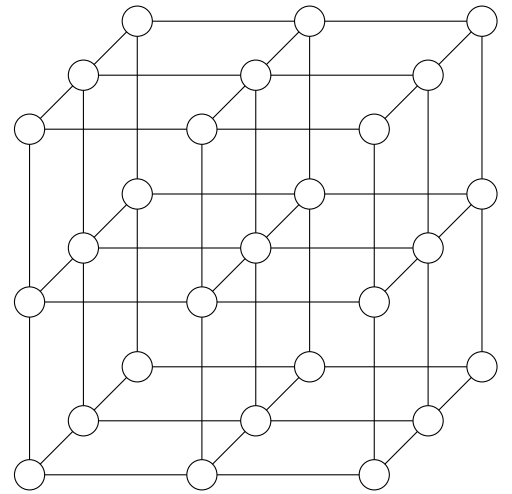
Multi-dimensional Meshes



(a)



(b)



(c)

Interconnects: Multidimensional Meshes

- Diameter of a 2 dimensional mesh is $2(\sqrt{p} - 1)$; diameter of a 2 dimensional torus is $2\lfloor \frac{\sqrt{p}}{2} \rfloor$
- Arc connectivity of a 2 dimensional mesh is 2 and for a torus it is 4
- Bisection width of a 2 dimensional mesh is \sqrt{p} ; for a torus it is $2\sqrt{p}$

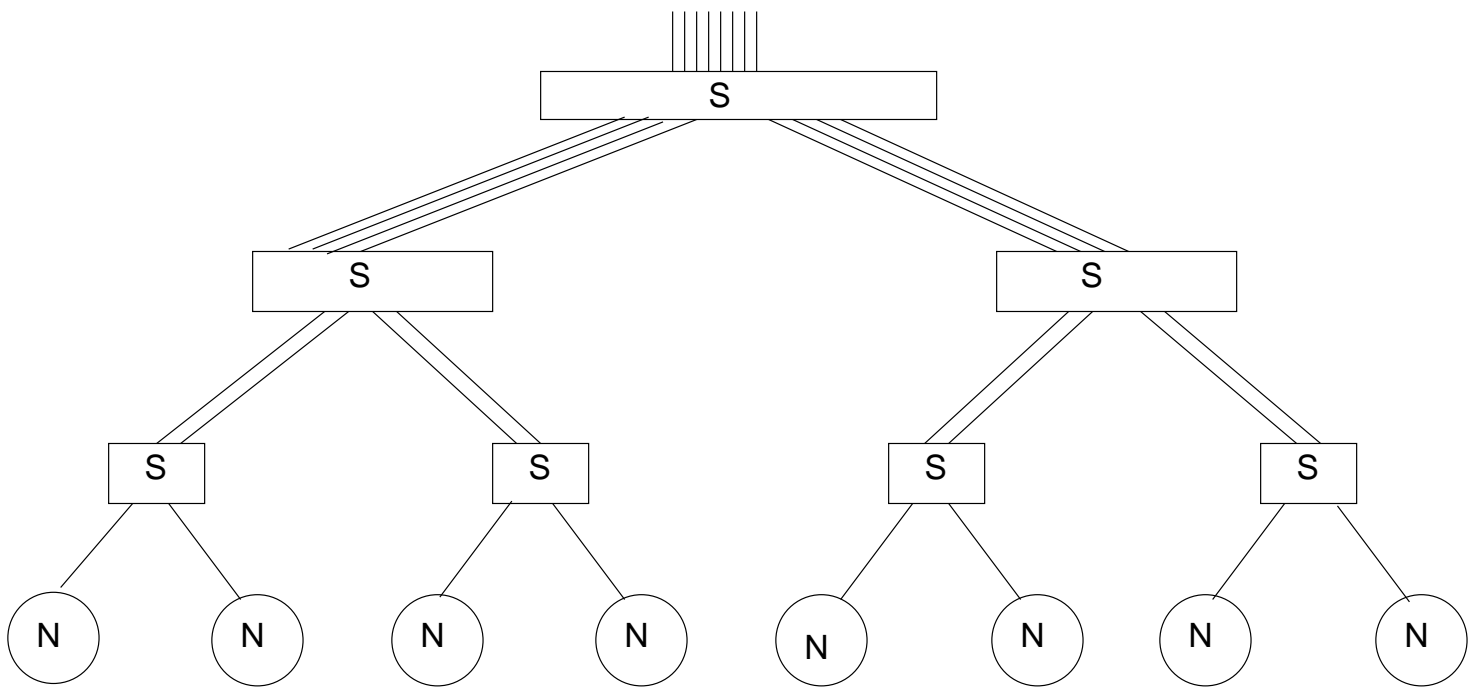
Interconnects: Fat Tree

- A tree network is one where there is exactly one path between a pair of processors
- In a simple tree, the bandwidth on higher level links is shared among all processors below creating bottlenecks
- A fat tree solves the problem by using wider links at higher levels in the tree
- Cost and performance of increasing complexity switches is main problem

Interconnects: Fat Tree

- Diameter of a fat tree is $2 \log((p + 1)/2)$
- Arc connectivity is 1
- Bisection width is 1 for a tree $p/2$ for a fat tree.

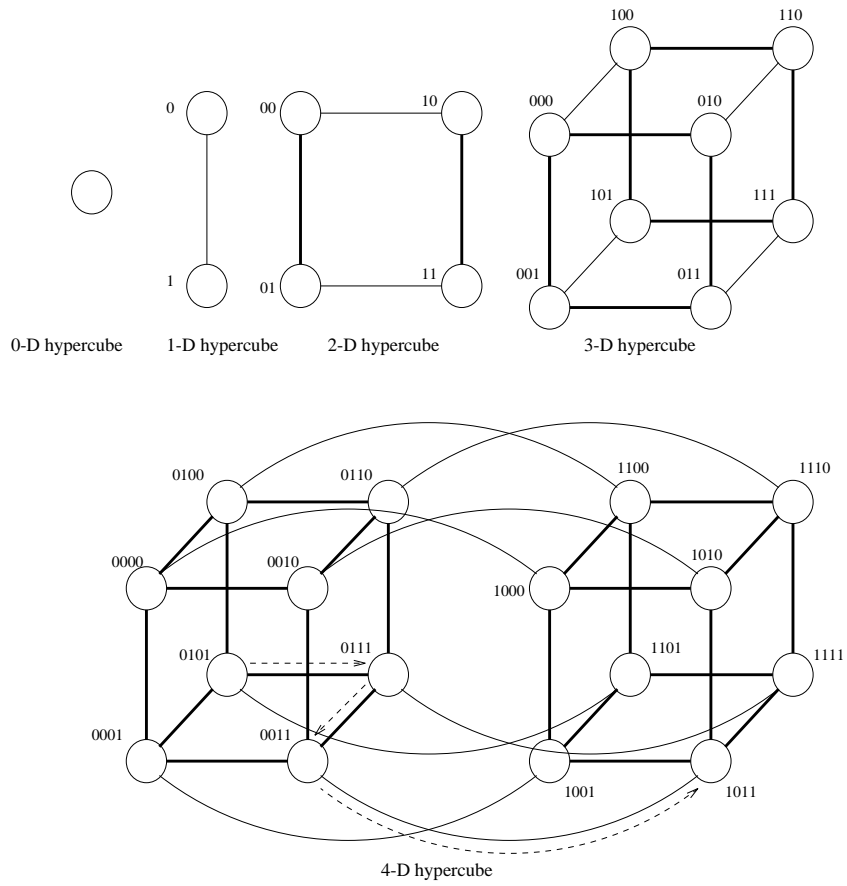
Interconnects: Fat Tree



Interconnects: Hypercube

- A hypercube is a multidimensional mesh with exactly two processors in each dimension
- In a d dimensional hypercube, each processor is connected with d other processors
- Hypercubes can be constructed recursively; when two d dimensional hypercubes are used to construct a $d + 1$ dimensional hypercube, the labels on the nodes of the original hypercubes are prefixed with a 0 or a 1 for the new hypercube

Interconnects: Hypercube



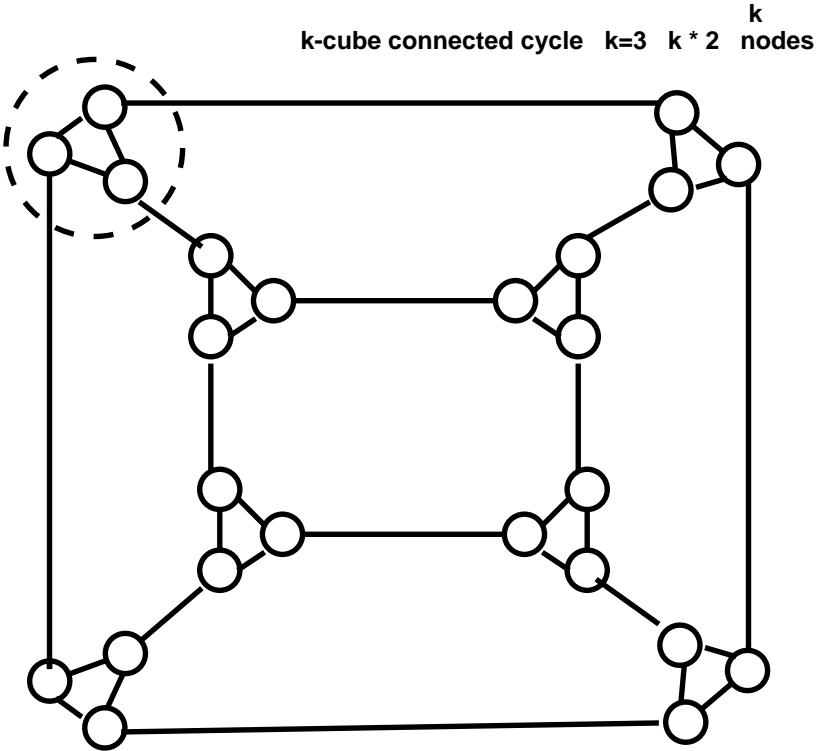
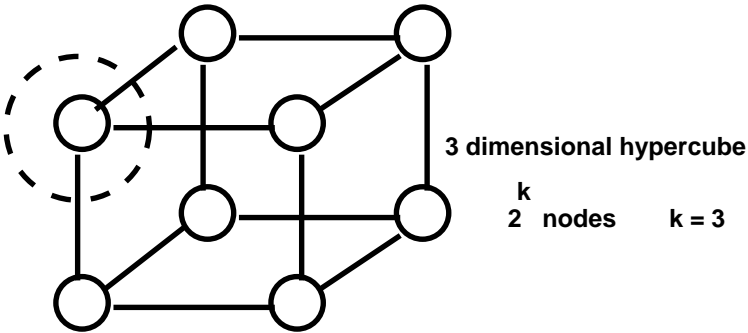
Interconnects: Hypercube

- Two processors are connected if their labels differ in exactly one bit position
- In a d dimensional hypercube, each processor is connected to exactly d other processors
- Consider two processors with labels s and t , the Hamming distance for the two processors is defined to be the number of positions in which their labels differ
- For example if $s = 001$ and $t = 101$, Hamming distance is 1
- The Hamming distance represents the shortest path between two processors

Interconnects: Hypercube

- Diameter of a hypercube is $\log(p)$ (smaller than others)
- Arc connectivity of a hypercube is $\log(p)$ (scaling problem as $p \rightarrow \infty$)
- Bisection width of a hypercube is $\frac{p}{2}$

In order to avoid the growing degree of the node as a function of the dimension k of the hypercube a ring of k nodes can replace each node in the hypercube to create a k -cube connected cycle



Effects of Interconnects on Programming

- Congestion
 - Traffic patterns on the network can create hot spots
 - Deadlock can occur in general, needs to be addressed by a combination of hardware and software
- Latency hiding
 - Do some other work while a data transfer is in progress
- Latency amortization
 - Try to fetch large amounts of data at the same time
- Note that these are all similar to shared memory issues.

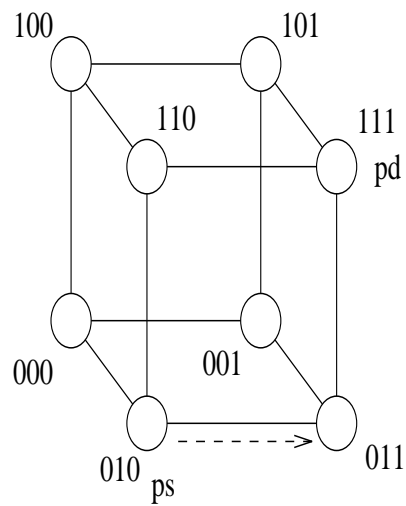
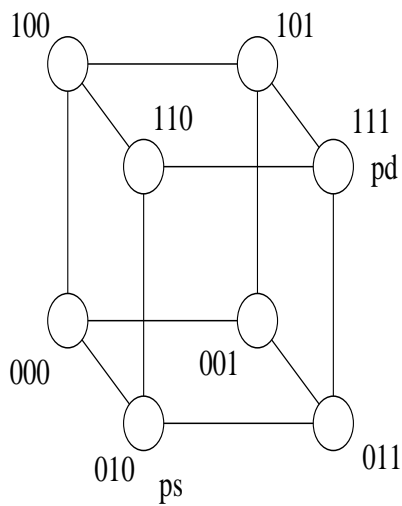
Message Routing Mechanisms

- Routing mechanism determines the path a message takes through the network when going from a source to a destination processor
- Minimal or non-minimal routing:
 - Minimal always takes the shortest path (maybe several min. paths)
 - Non-minimal may sometimes take a longer path to avoid congestion
- Deterministic or adaptive:
 - Deterministic routine always takes the same path between processors (typical of shared memory, and older distributed memory)
 - Adaptive routing can take different paths depending on congestion (becoming more common)

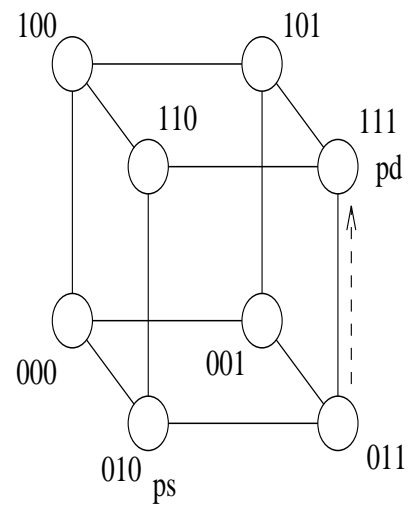
Example: E-cube Routing in Hypercubes

- If the message is at processor P_i and needs to go to P_d , compute $s = P_i \oplus P_d$. Send message along dimension k from P_i , where k is the least significant non-zero bit in s
- Continue this process at each successive processor until P_d is reached
- s gives the bits that have to be switched, any order can be used to get a deterministic minimal path

Example: E-cube Routing in Hypercubes



Step 1 (010 \rightarrow 011)



Step 2 (011 \rightarrow 111)

Example: XY Routing in 2d Mesh

- Message sent along X dimension till destination's X coordinate is reached
- Now message is sent along Y dimension till it reaches destination processor
- Scheme is minimal and deterministic