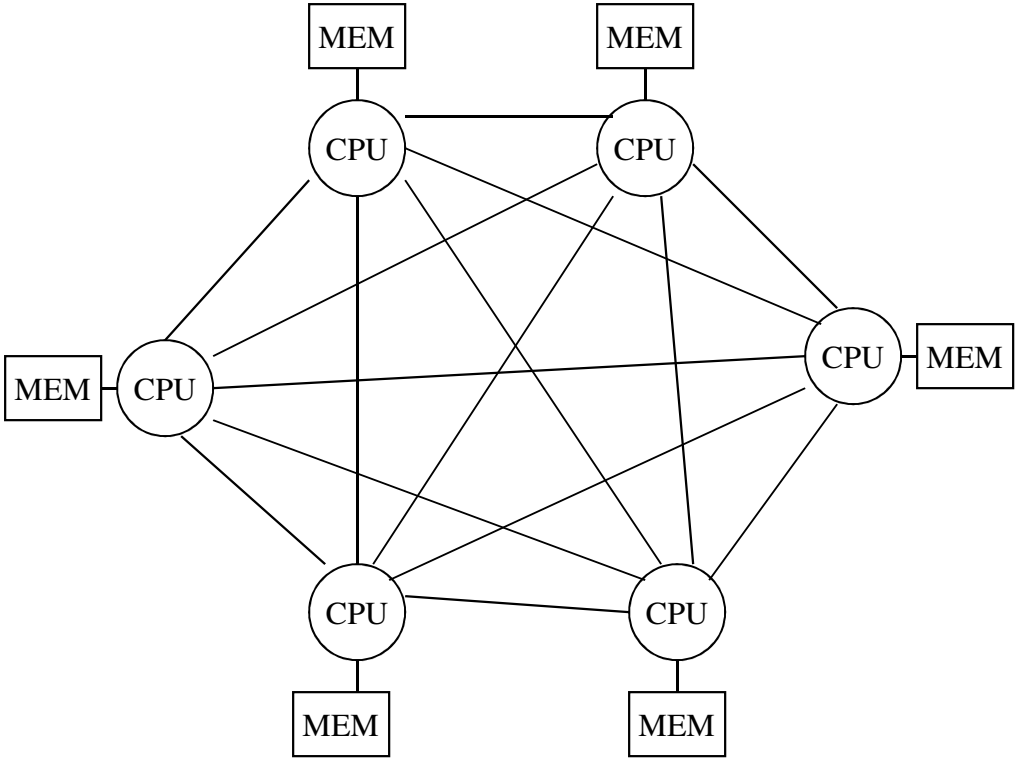
“Distributed Address Space: Network Topologies”

Outline

- Overview of distributed address space
- Performance Metrics
- Topologies
- Routing

Distributed Address Space

- Each processor has its own local address space
- Processors share data via explicit message passing



Distributed Address Space

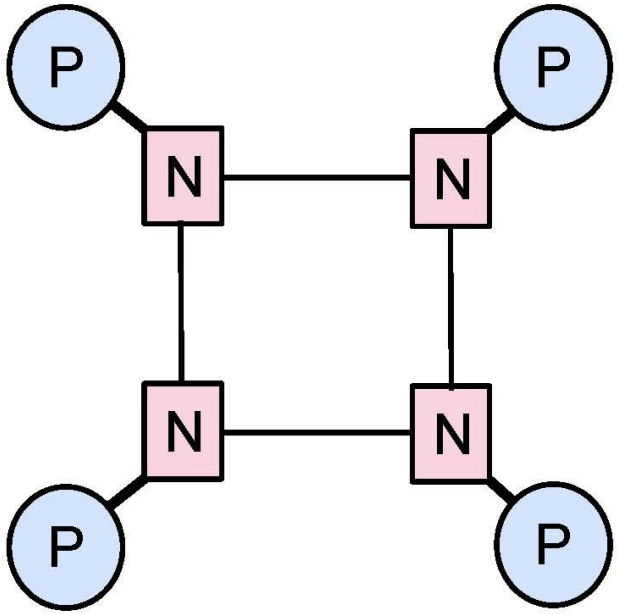
- Important aspects are:
  - Interconnect
  - Message routing mechanism
- These aspects determine:
  - Performance
  - Scalability
  - cost

Interconnect Networks

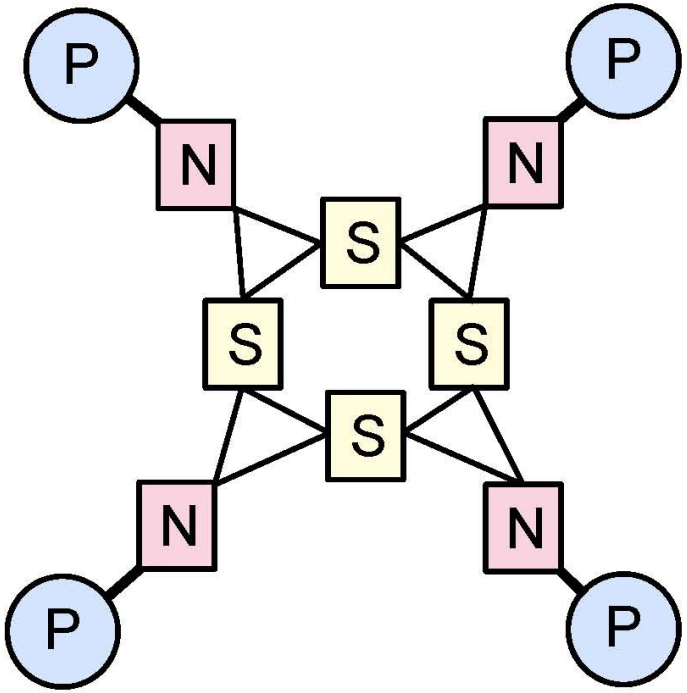
- Carry data between processors
- Interconnect components
  - Switches
  - Links(wires, fiber)
- Interconnect network flavors
  - Static network: point-to-point comm links
    - Direct networks
  - Dynamic networks: switches and comm links
    - Indirect network

Static vs. Dynamic Networks

Static/direct network



Dynamic/indirect network



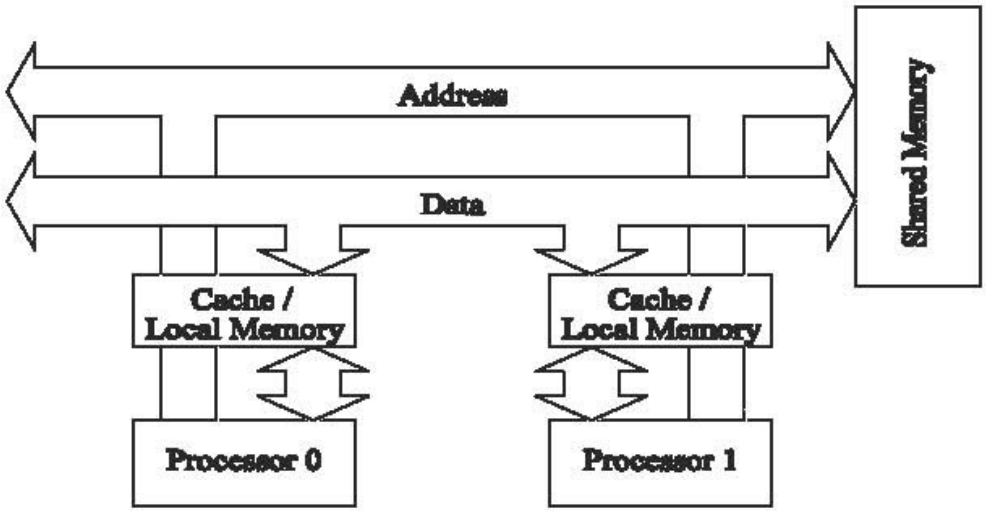
Performance Metrics

- Degree: number of links per node
- Diameter: maximum distance between any two processors.
- Connectivity: the multiplicity of paths between processors.
  - Arc connectivity: the minimum number of links that need to be removed in order to bread the network into two disjoint parts
- Bisection width: the minimum number of links that need to be removed in a network to separate the processor into two halves
- Cost:
  ~ # links and switches

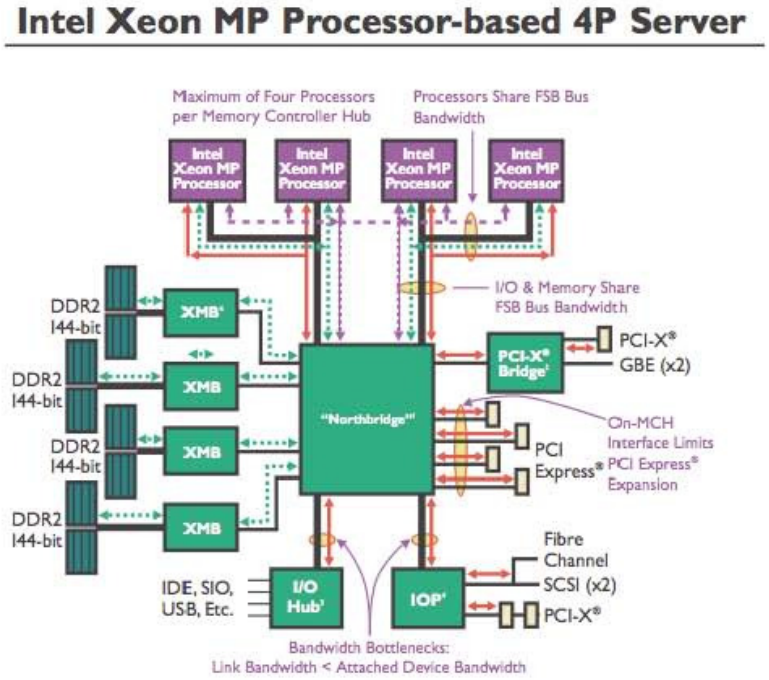
Network Topologies: Bus

- All processors access a common bus for exchanging data
- Used in simples and earliest parallel machines
- Advantages
  - ?
- Disadvantages
  - ?

Bus



Bus-based interconnect with local memory/cache

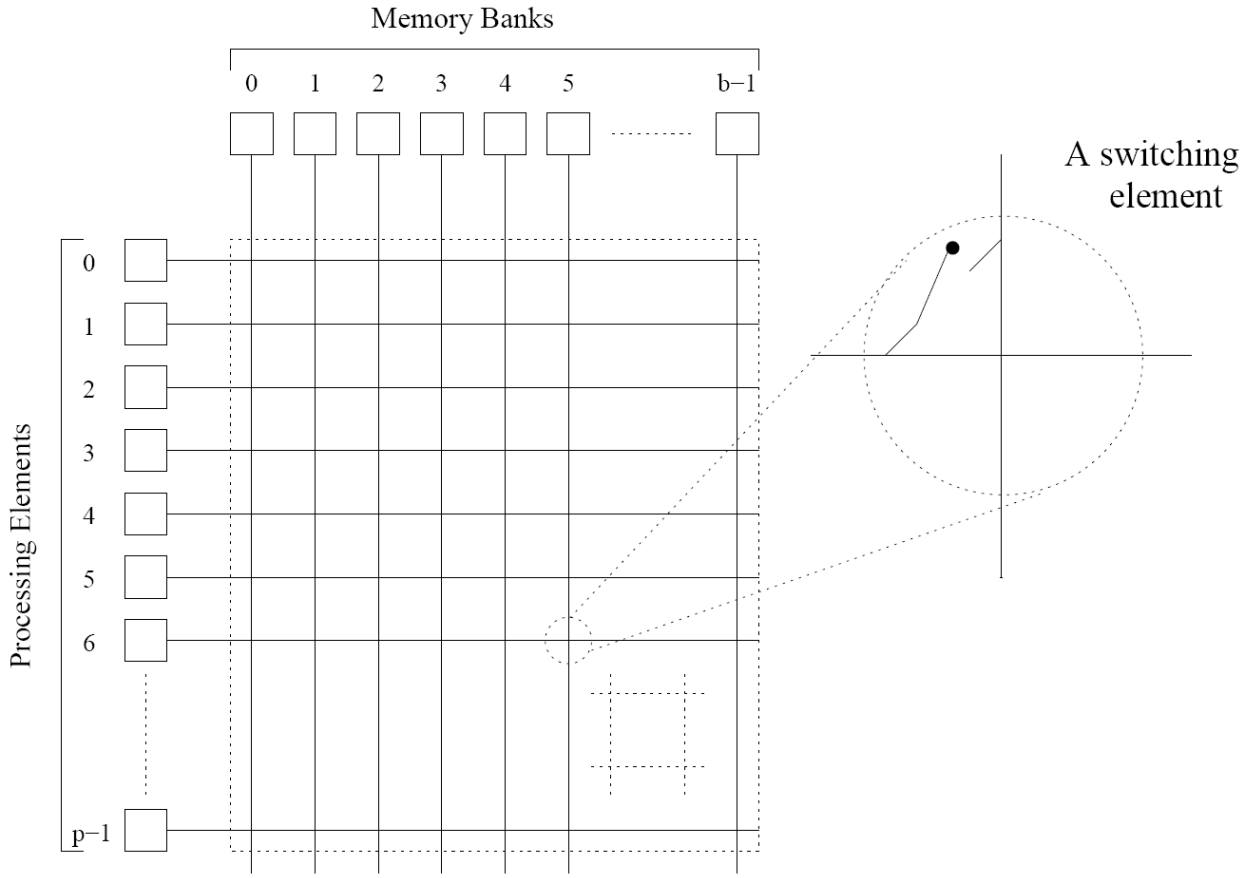


Dual-bus (circa 2005)

Since much of the data accessed by processors is local to the processor, a local memory can improve the performance of bus-based machines

Crossbar Network

A crossbar network uses an $p \times m$ grid of switches to connect p inputs to m outputs in a non-blocking manner



Crossbar Network

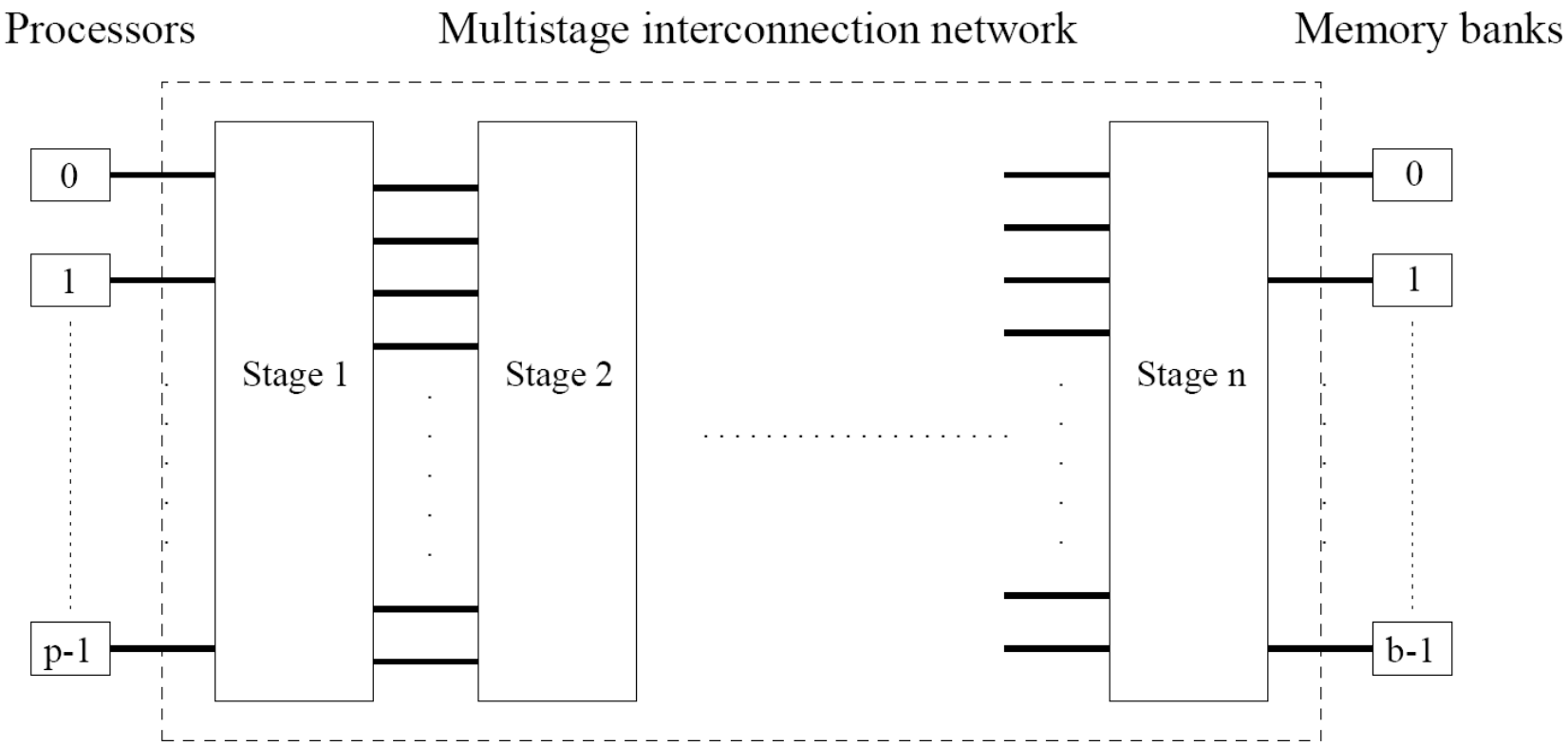
- Suppose a $p \times p$ crossbar, its cost = ?

- Examples:
  - Full crossbar
    - Earth simulator: custom 640-way single-stage crossbar
  - Crossbar as building block
    - Rice RTC (retired in 2008): 16-way crossbar switches in 128-way Clos network

Multistage Interconnect

- Buses
  - Excellent cost scalability
  - Poor performance scalability
- Crossbars
  - Excellent performance scalability
  - Poor cost scalability
- Multistage interconnects
  - Compromise between these extremes

Multistage Network



Schematic of a typical multistage interconnection network (processor-to-memory, e.g., BBN Monarch)

Multistage Omega Network

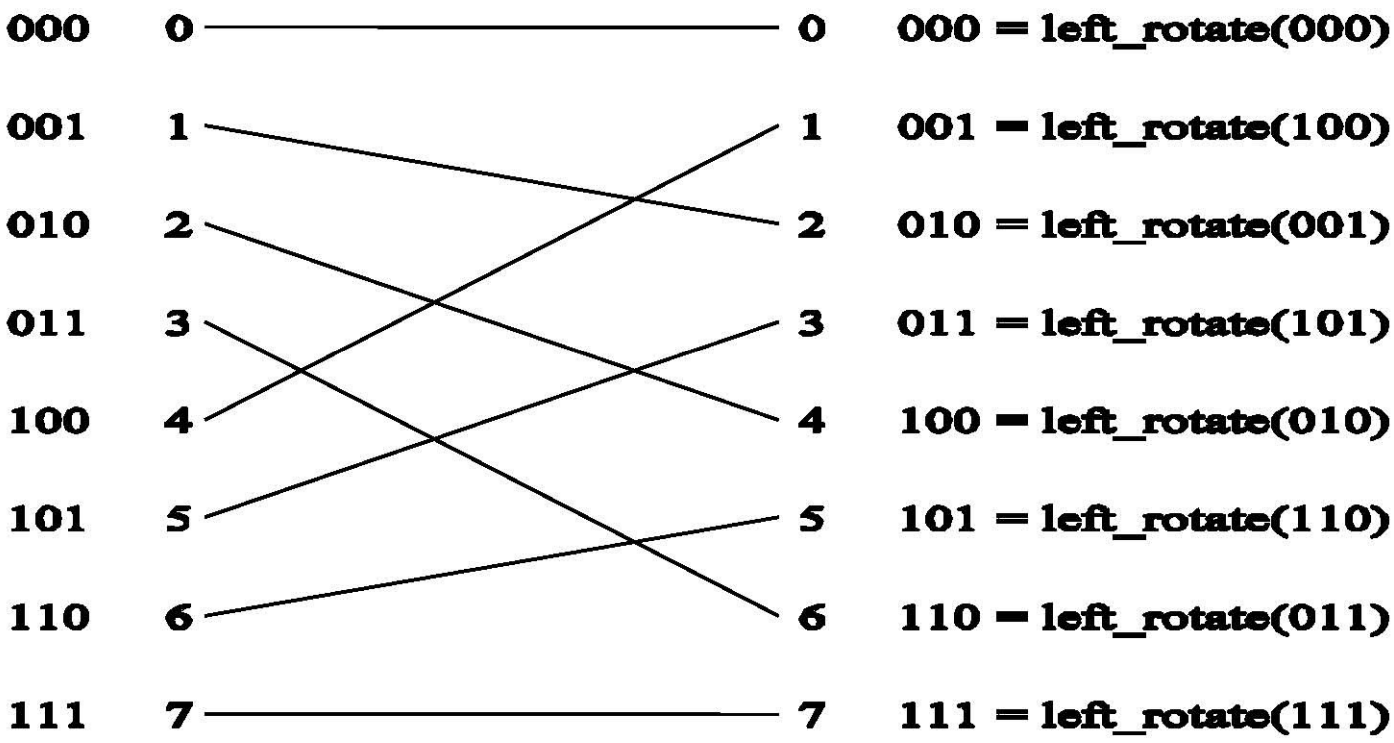
- One of the most commonly used multistage interconnects
- Organization
  - ***log p*** stages
  - ***p*** number of inputs/outputs
- At each stage, input *i* is connected to output *j* if:

$$j = \begin{cases} 2i, & 0 \leq i \leq p/2 - 1 \\ 2i + 1 - p, & p/2 \leq i \leq p - 1 \end{cases}$$

- If $p=2^k$, then j=?

Omega Network Stage

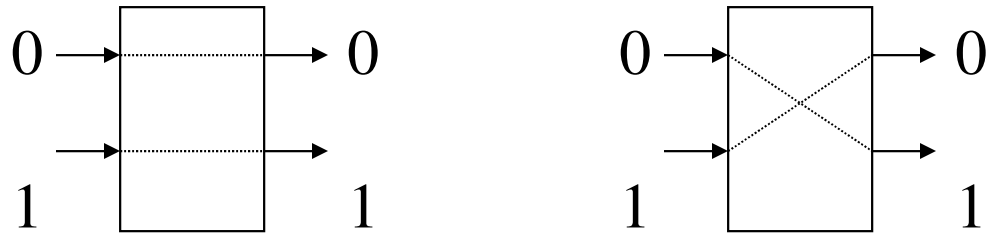
Each Omega stage is connected in a perfect shuffle



A perfect shuffle interconnection for eight inputs and outputs

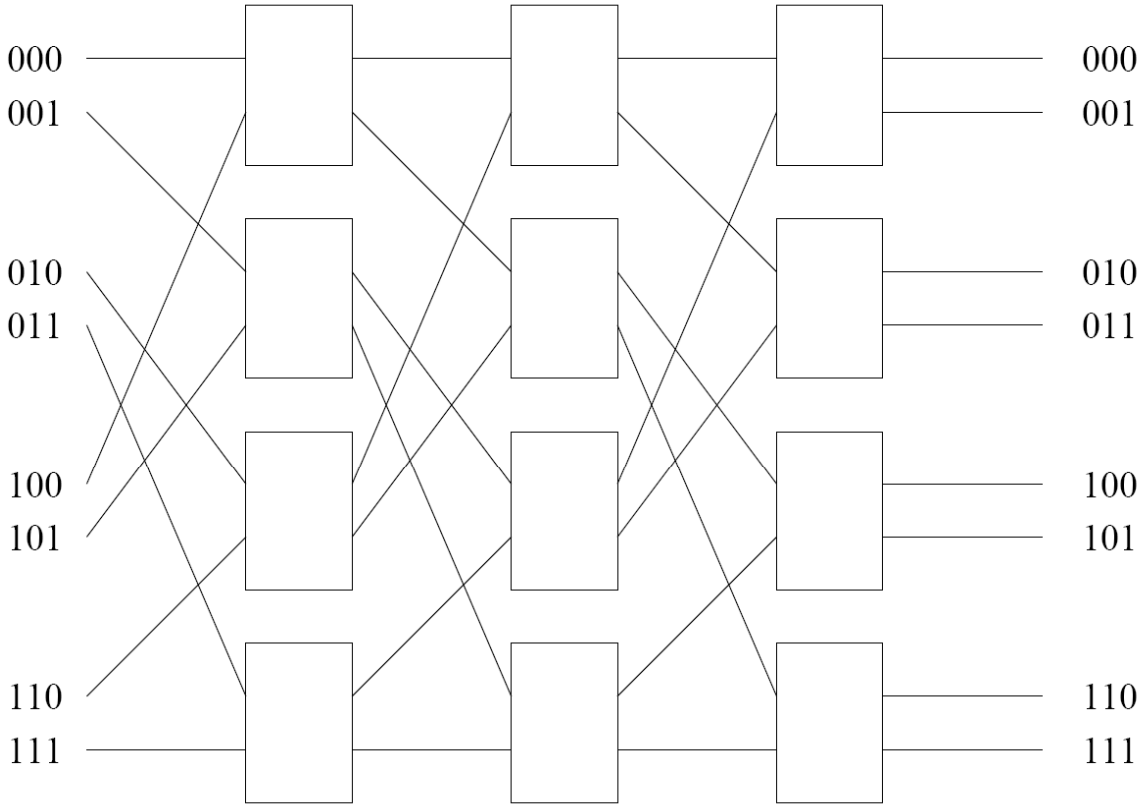
Omega Network Switches

- The perfect shuffle patterns are connected using 2×2 switches
- The switches operate in two modes:
  - Cross-over vs. pass-through.



(a) pass-through (b) cross-over

Multistage Omega Network



A complete omega network connecting eight inputs and eight outputs.

Cost: *??*

Omega Network Routing

- Let
  - s= binary representation of the source processor
  - d= binary representation of the destination processor or memory
- The data traverses the link to the first switching node

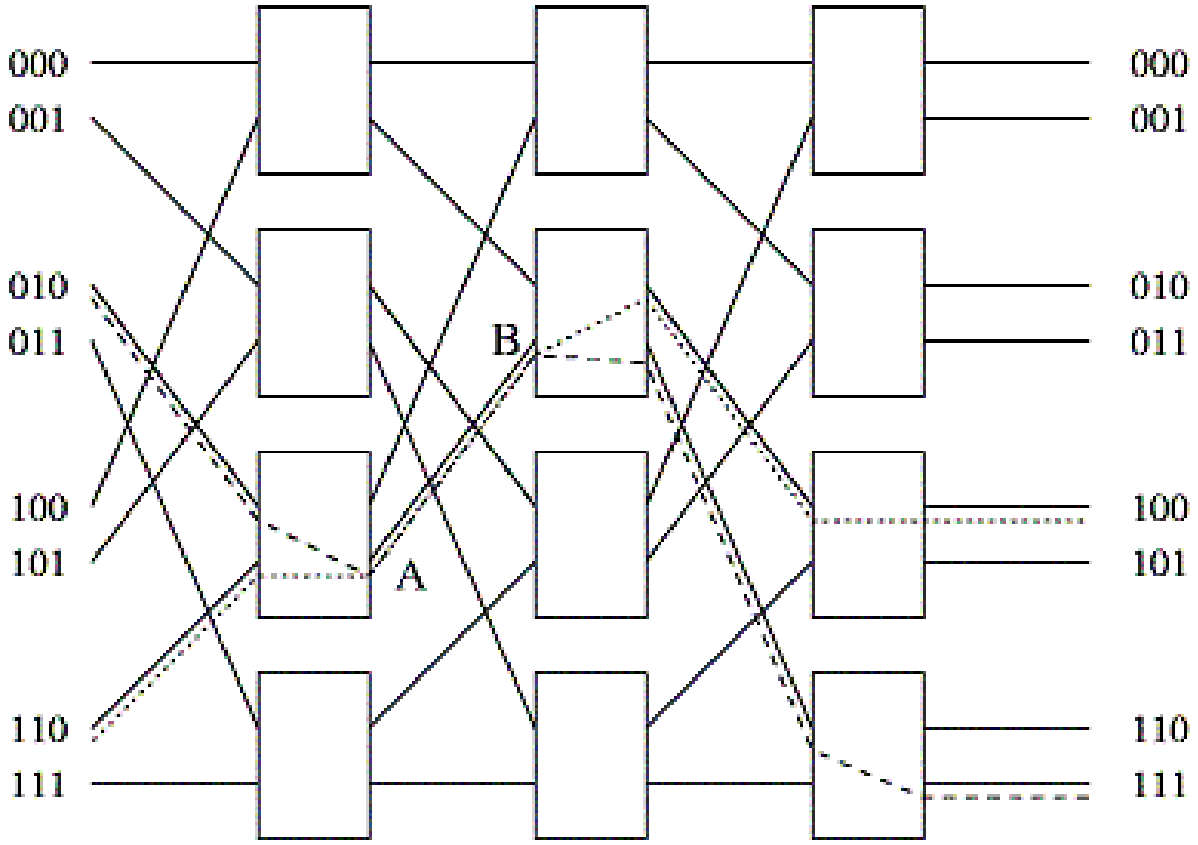
  if the most significant bit of s and d are the same

    pass-through

  else

    cross-over
- Strip off leftmost bit of s and d
- Repeat for each of the log(p) switching stages

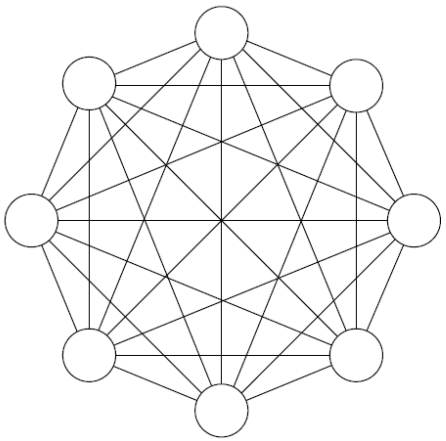
Omega Network Routing

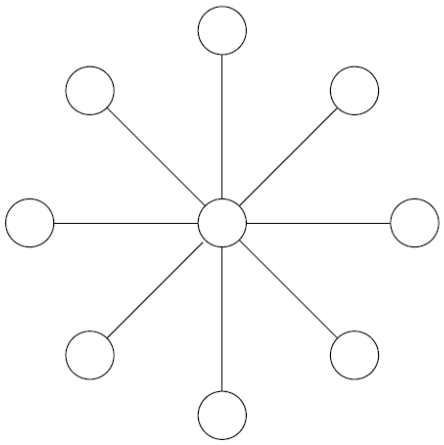


An example of blocking in omega network: one of the messages (010 to 111 or 110 to 100) is blocked at link AB

Star & Complete Connected Network

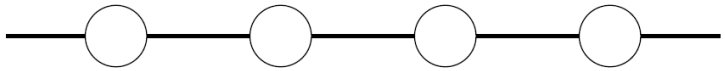
- Star
  - Static counterpart of bus
  - Every node connected only to a common node at the center
  - Distance between any pair of nodes is O(1)
- Complete connected network
  - Static counterpart of crossbar
  - Each processor is connected to every other processor.
  - The number of links in the network scales as $O(p^2)$



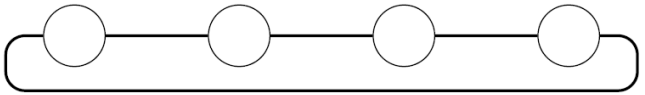


Linear Array

- Each node has two neighbors: left & right

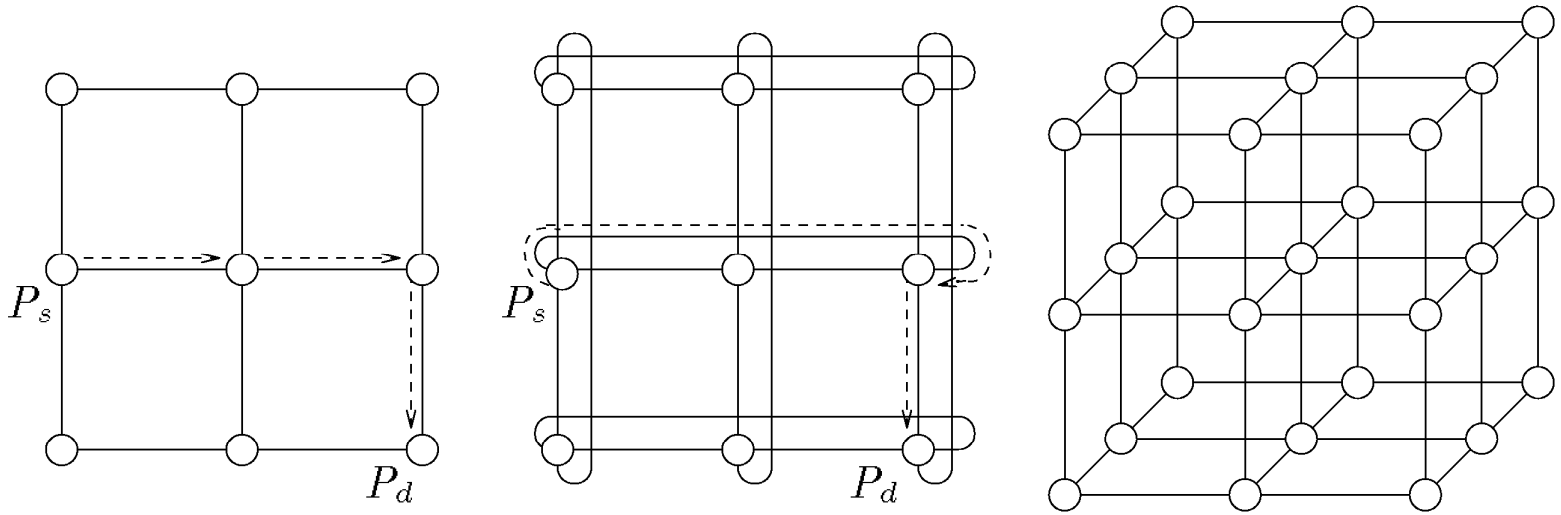


- If connection between nodes at ends: 1D torus (ring)
  - Diameter of a ring is ?
  - Arc connectivity of a ring is ?
  - Bisection width of a ring is ?



Multidimensional Meshes

- Mesh: generalization of linear array to 2D or 3D
- Mesh with wrap around: torus
- K-dimensional mesh
  - Node has 2k neighbors



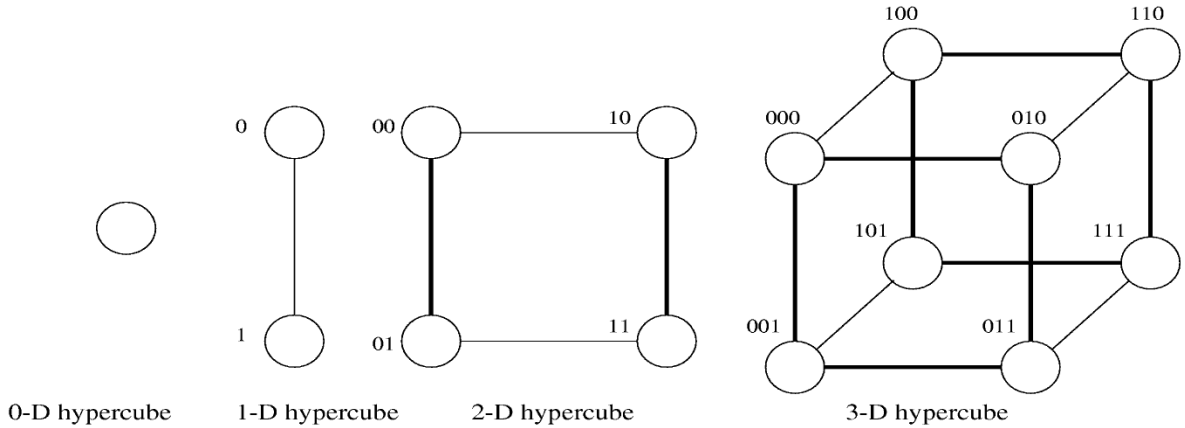
Multidimensional Meshes

- Diameter of a 2D mesh is ? ; for torus it is ?
- Arc connectivity of a 2D mesh is ?; for torus it is ?
- Bisection width of a 2D mesh is ? ; for torus it is ?

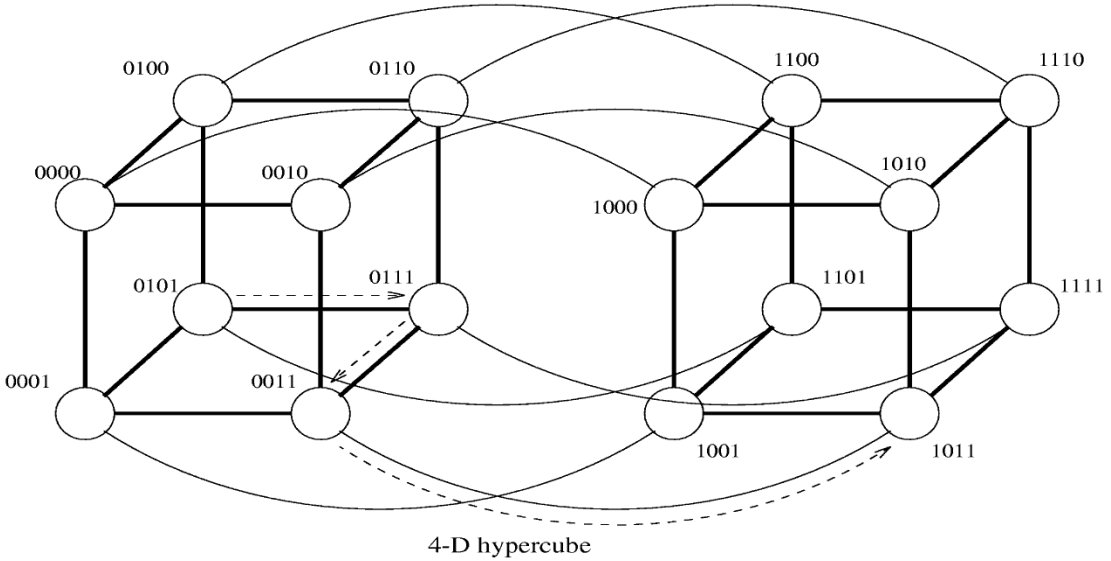
Hypercube

- 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

Hypercube



0-D hypercube 1-D hypercube 2-D hypercube 3-D hypercube



4-D hypercube

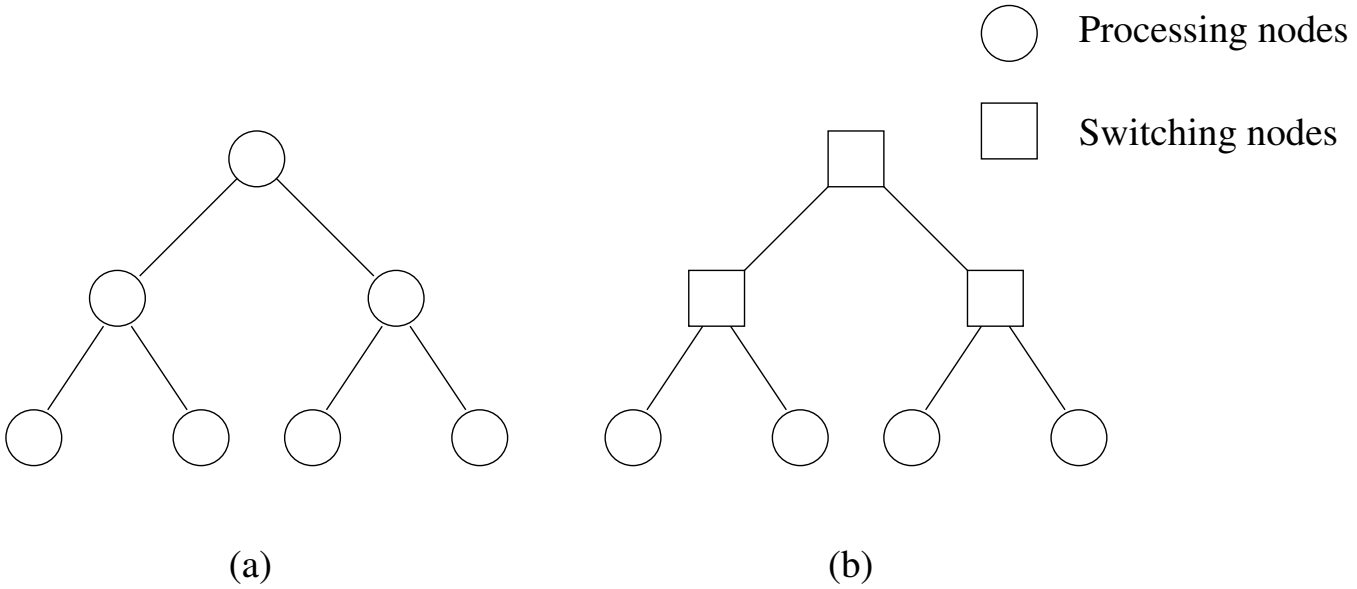
Hypercube

- Hypercubes can be constructed recursively
- Two processors are connected if ?
- Consider two processors with labels s and t, what is the Hamming distance for the two processors?

Hypercube

- Diameter of a hypercube is ?
- Arc connectivity of a hypercube is ?
- Bisection width of a hypercube is ?

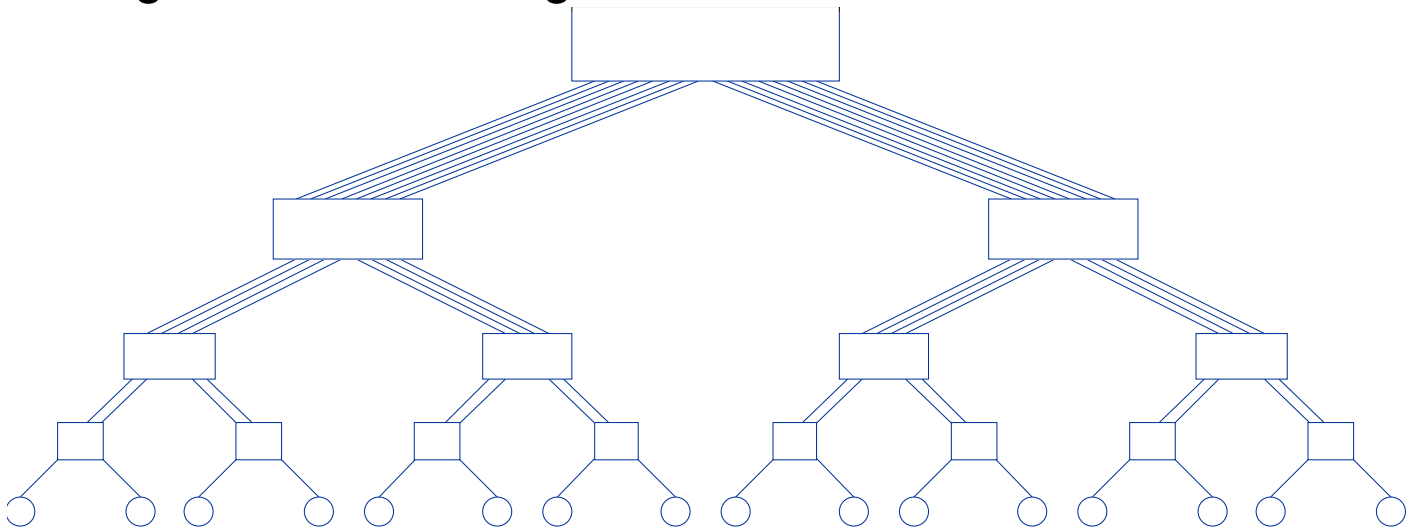
Trees



(a) Static tree network (b) dynamic tree network

Tree Properties

- Distance between any two nodes is no more than *2logp*
- Problem:
  - Links higher up the tree potentially carry more traffic than those at the lower levels
- Solution: fat tree
  - Using wider links at higher levels in the tree



Fat tree network for 16 processors

Tree

- Diameter of a tree is ?
- Arc connectivity is ?
- Bisection width is ?

Metrics for Static Interconnects

Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Completely-connected	1	$p^2/4$	$p-1$	$p(p-1)/2$
Star	2	1	1	$p-1$
Complete binary tree	$2\log((p+1)/2)$	1	1	$p-1$
Linear array	$p-1$	1	1	$p-1$
2-D mesh, no wraparound	$2(\sqrt{p}-1)$	$\sqrt{p}$	2	$2(p-\sqrt{p})$
2-D wraparound mesh	$2\lfloor\sqrt{p}/2\rfloor$	$2\sqrt{p}$	4	$2p$
Hypercube	$\log p$	$p/2$	$\log p$	$(p\log p)/2$
Wraparound *k*-ary *d*-cube	$d\lfloor k/2\rfloor$	$2k^{d-1}$	$2d$	dp

Metrics for Dynamic Interconnects

Network	Diameter	Bisection Width	Arc Connectivity	Cost (No. of links)
Crossbar	1	p	1	p^2
Omega Network	$\log p$	$p/2$	2	$p/2$
Dynamic Tree	$2 \log p$	1	2	$p - 1$

Message Passing Costs

- *Startup time* (t_s)
  - Time spent at sending and receiving nodes
  - Including?
- *Per-hop time* (t_h)
  - A function of number of hops
  - Including?
- *Per-word transfer time* (t_w)
  - Including?

Store-and-Forward Routing

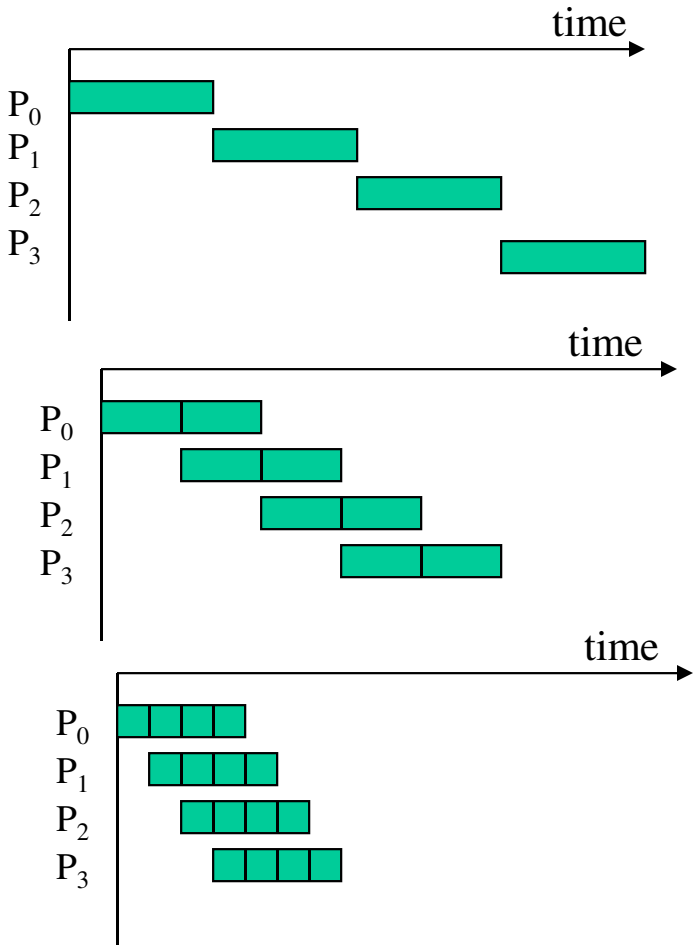
- Definition:
  - Message traversing multiple hops is completely received at an intermediate hop before being forwarded to the next hop
- The total communication cost for a message of size *m* words to traverse *l* communication links is

$$t_{comm} = t_s + (mt_w - t_h)l.$$

- Typically, t_h is small
- Thus, approximate store-and-forward message cost

$$t_{comm} = t_s + mlt_w.$$

Routing Techniques



(1) A single message sent over A store-and-forward network

(2) The same message broken into two parts and send over the network

(3) The same message broken into four parts and sent over the network

Packet Routing

- Store-and-forward makes poor use of comm resources
- Packet routing
  - Break messages into ***packets***
  - **Pipeline** them through the network
- Packets may take different paths, thus each packet must carry
  - Routing information
  - Error checking
  - Sequencing information
- The total communication time for packet routing is

$$t_{comm} = t_s + t_h l + t_w m.$$

- t_w accounts for overheads in packet headers

Cut-Through Routing

- Packet routing in the extreme
  - Divide messages into basic units called ***flits***
  - Flits are typically small -> header info must be small
- What are the differences between cut-through and packet routing?

- Used in today's infiniband networks

Cut-Through Routing

- Communication time for cut-through routing is

$$t_{comm} = t_s + t_h l + t_w m.$$

- This is identical to packet routing; however, t_w is typically much smaller
- Typically, t_h smaller than t_s and t_w
- Approximate communication time using cut-through routing is

$$t_{comm} = t_s + t_w m.$$

Simplified Cost Model for Messages

- Valid for only uncongested networks
- If a link takes multiple messages
  - corresponding t_w term must be scaled up by # messages
- Network congestion varies by
  - Communication pattern
  - Match between pattern and network topology
- Communication models must account for congestion

Routing Variants

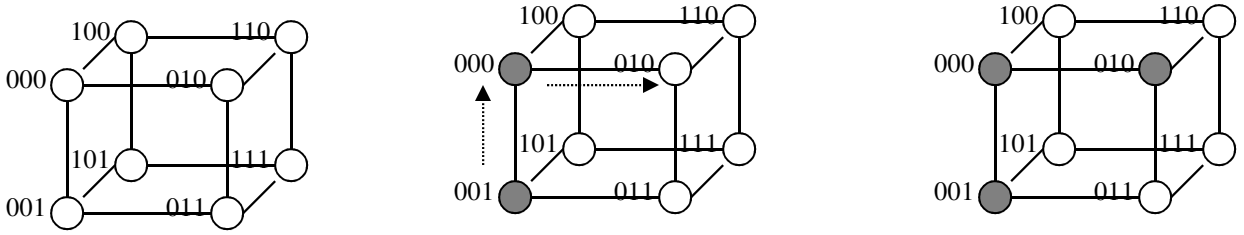
- Routing mechanism determines the path a message takes through the network when going from a source to a destination processor
- Minimal or non-minimal routing
- Deterministic or adaptive

Routing Challenges

- Routing must prevent deadlocks
  - Use dimension-ordered or e-cube routing
- Routing must avoid hot-spts
  - Can use two-step routing
  - Message from s to d: first sent to a randomly chosen intermediate processor i; then forward from i to destination d

Example: E-cube Routing in Hypercube

- 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
- Continues this process at each successive processor until P_d is reached



Example: XY Routing in 2D Mesh

- Message sent along X dimension until destination's X coordinate is reached
- Then message is sent along Y dimension until it reaches destination processor
- Called dimension-order routing

Summary

- Overview of distributed address space
- Performance Metrics
- Topologies
- Routing

- Reading material: Kumar – Chpt 2