Lecture 5

Interconnection Networks for Parallel Computers

Communication Costs in Parallel Machines

adapted from…

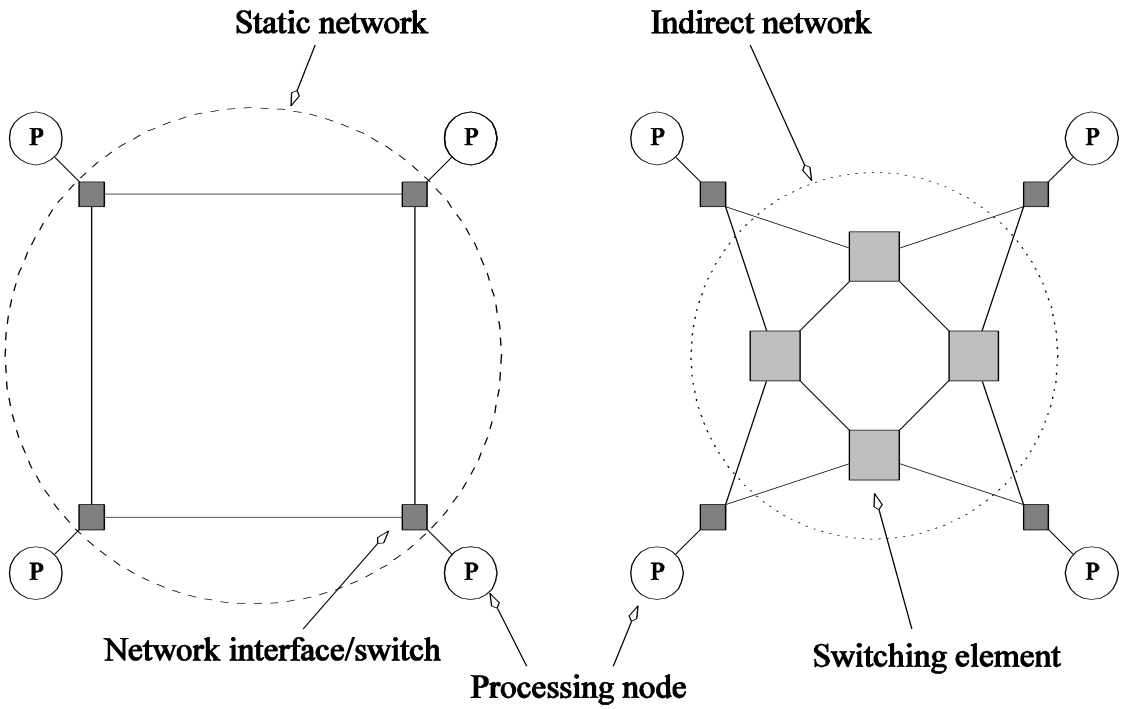
Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar

``Introduction to Parallel Computing",
Addison Wesley, 2003

Interconnection Networks for Parallel Computers

- Interconnection networks carry data between processors and to memory.
- Interconnects are made of
  - *switches* and
  - *links* (wires, fiber).
- Interconnects are classified as static or dynamic.
  - ***Static networks*** consist of point-to-point communication *links* among processing nodes and are also referred to as *direct* networks.
  - ***Dynamic networks*** are built using *switches* and communication *links*. Dynamic networks are also referred to as *indirect* networks.

Static and Dynamic Interconnection Networks



Classification of interconnection networks:
(a) a static network; and (b) a dynamic network.

What Characterizes an Interconnection Network?

- Topology (what)
  - Interconnection structure of the network graph
- Routing Algorithm (which)
  - Restricts the set of paths that messages may follow
  - Many algorithms with different properties
- Switching Strategy (how)
  - How data in a message traverses a route
  - *circuit switching* vs. *packet switching*
- Flow Control Mechanism (when)
  - When a message or portions of it traverse a route
  - What happens when traffic is encountered?

Topological Properties

- Routing distance
  - Number of links on route from source to destination
- Diameter
  - Maximum routing distance
- Average distance
- Partitioned network – bisection width
  - Removal of links resulting in 2 disconnected graphs
  - Minimal cut
- Scaling increment
  - What is needed to grow the network to next valid degree

Interconnection Networks

- Switches map a fixed number of inputs to outputs.
- The total number of ports on a switch is the *degree* of the switch.
- The cost of a switch grows as
  - the square of the degree of the switch,
  - the peripheral hardware linearly as the degree, and
  - the packaging costs linearly as the number of pins (many pins on a port… how many? 2/5/9/25…).

Interconnection Networks: Network Interfaces

- Processors interact to the network via a network interface.
- The network interface may hang off the I/O bus or the memory bus.
- In a physical sense, this distinguishes a cluster from a tightly coupled multicomputer.
- The relative speeds of the I/O and memory buses impact the performance of the network.

Network Topologies

- A variety of network topologies have been proposed and implemented.
- These topologies tradeoff performance for cost.
- Commercial machines often implement hybrids of multiple topologies for reasons of packaging, cost, and available components.

Scalability

- **scalability** is the ability of a system, network, or process,

to handle growing amounts of work

  - informal definition=> in a “graceful” manner
- For example, it can refer to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added.

  - **Ability to be enlarged to accommodate** growing amounts of work.

    [André B. Bondi, 'Characteristics of scalability and their impact on performance', *Proceedings of the 2nd international workshop on Software and performance,* Ottawa, Ontario, Canada, 2000, pages 195 - 203]

Bisection bandwidth

- The sum of the capacities of links between two partitions of a network is called the bandwidth of the partition.
- Bisection is the division of a network(graph) into two equal partitions.
- The bisection width is minimum number of links one have to cut to partition the network in two equal parts.

- The bisection bandwidth of a network topology is the bandwidth available between the two partitions.

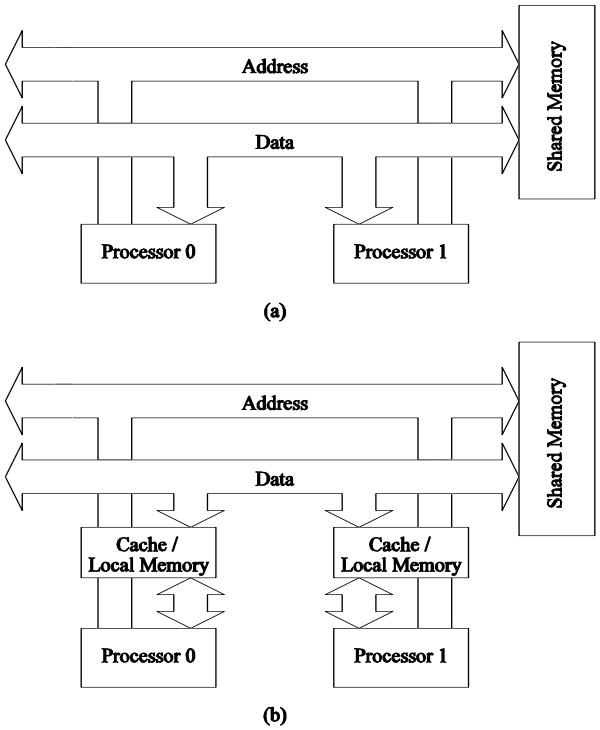
- Bisection bandwidth accounts for the ***bottleneck bandwidth*** of the entire network.

  ->bisection bandwidth represents a very good metric for bandwidth characteristics of the network

Network Topologies: Buses

- Some of the simplest and earliest parallel machines used buses.
- All processors access a common bus for exchanging data.
- The distance between any two nodes is *O(1)* in a bus. The bus also provides a convenient broadcast media.
- However, the bandwidth of the shared bus is a major bottleneck.
- Typical bus based machines are limited to dozens of nodes. Sun Enterprise servers and Intel Pentium based shared-bus multiprocessors are examples of such architectures.

Network Topologies: Buses

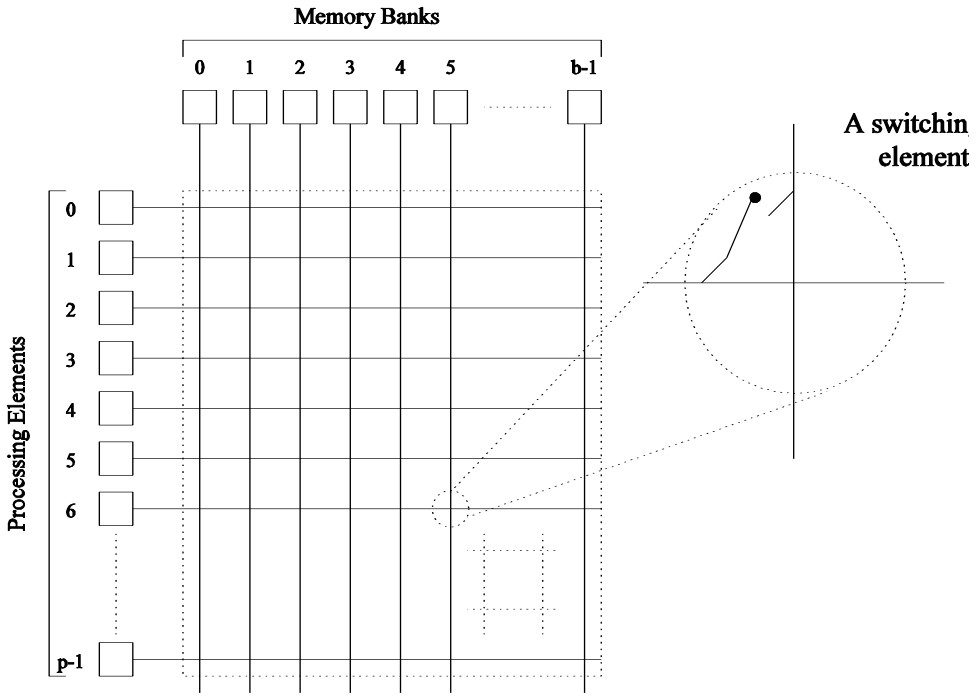


Bus-based interconnects (a) with no local caches; (b) with local memory/caches.

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

Network Topologies: Crossbars

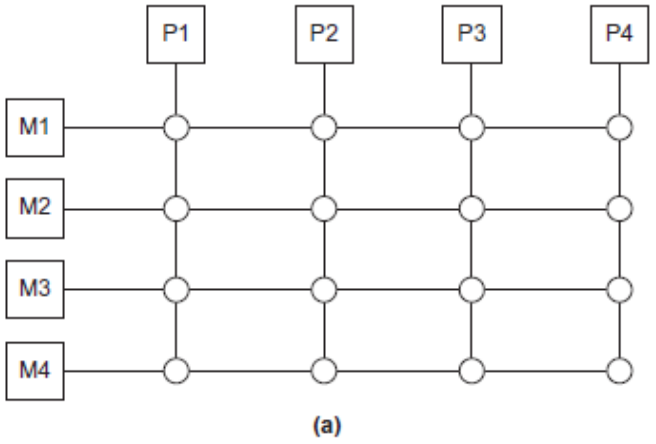
A crossbar network uses an $p \times m$ grid of switches to connect p inputs to m outputs in a non-blocking manner.



A completely non-blocking crossbar network connecting p processors to b memory banks.

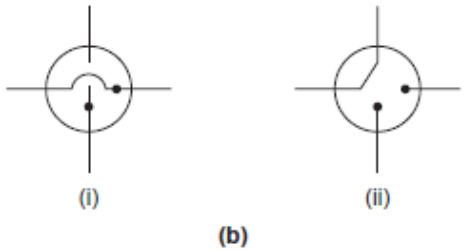
(a)

A crossbar switch connecting 4 processors (P_i) and 4 memory modules (M_j)

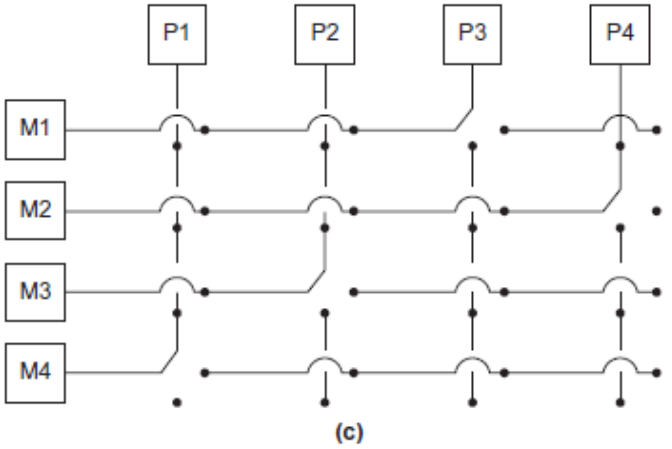


(b)

Configuration of internal switches in a crossbar



(c) Simultaneous memory accesses by the processors



Network Topologies: Crossbars

- The cost of a crossbar of p processors grows as $O(p^2)$.
- This is generally difficult to scale for large values of p.
- Examples of machines that employ crossbars include the Sun Ultra HPC 10000 and the Fujitsu VPP500.

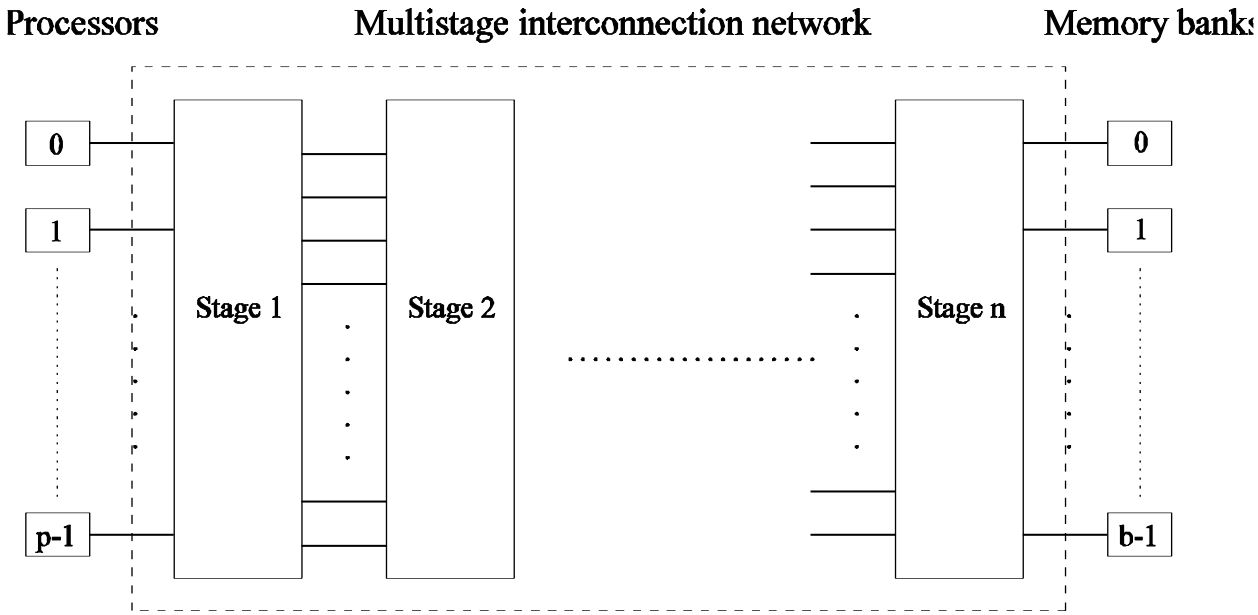
Fujitsu VPP500 Supercomputer Cabinets.

http://www.computerhistory.org/collections/catalog/102685357

Network Topologies: Multistage Networks

- Crossbars have excellent performance scalability but poor cost scalability.
- Buses have excellent cost scalability, but poor performance scalability.
- Multistage interconnects strike a compromise between these extremes.

Network Topologies: Multistage Networks



The schematic of a typical multistage interconnection network.

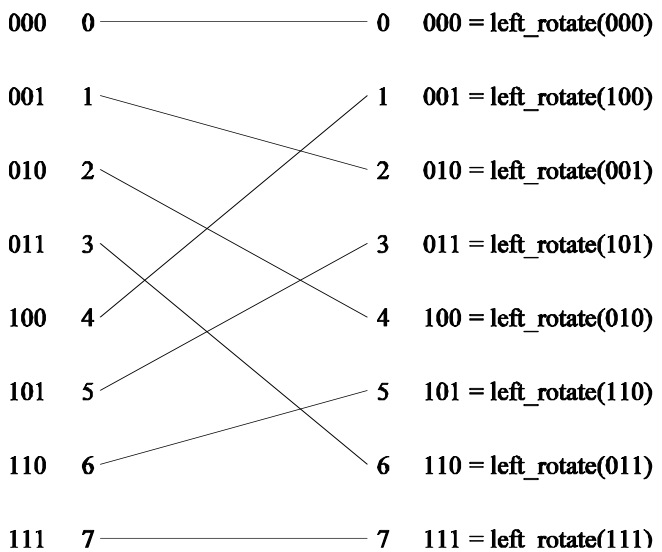
Network Topologies: Multistage Omega Network

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

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

Network Topologies: Multistage Omega Network

Each stage of the Omega network implements a perfect shuffle as follows:



A perfect shuffle interconnection for eight inputs and outputs.

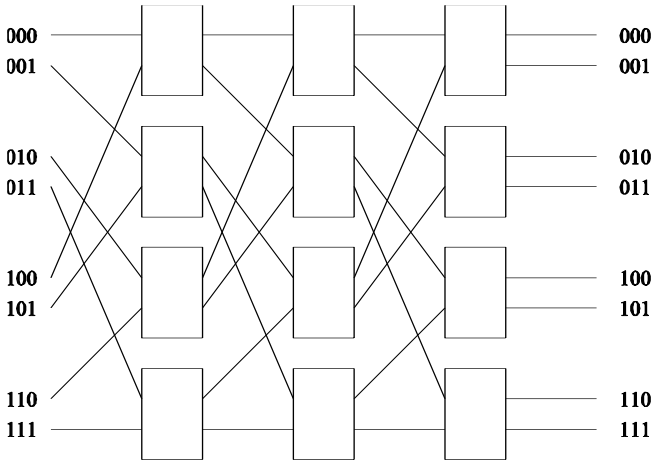
Network Topologies: Multistage Omega Network

- The perfect shuffle patterns are connected using 2×2 switches.
- The switches operate in two modes – crossover or passthrough.

Two switching configurations of the 2 × 2 switch:
(a) Pass-through; (b) Cross-over.

Network Topologies: Multistage Omega Network

A complete Omega network with the perfect shuffle interconnects and switches can now be illustrated:



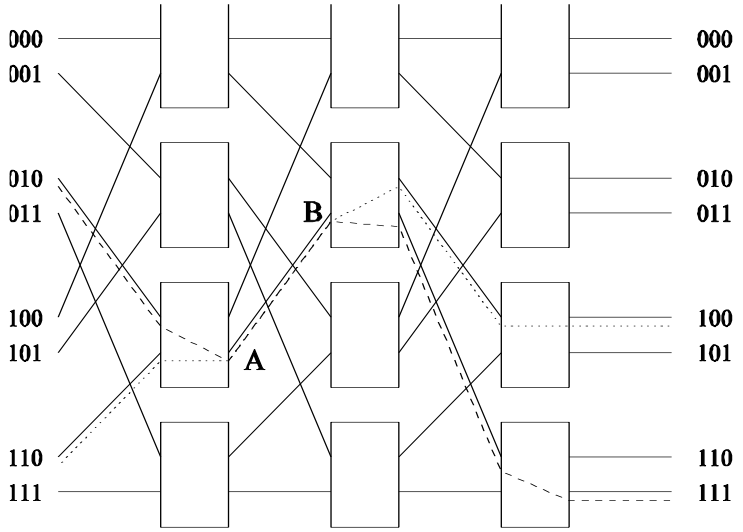
A complete omega network connecting eight inputs and eight outputs.

An omega network has $p/2 \times \log p$ switching nodes, and the cost of such a network grows as $(p \log p)$.

Network Topologies: Multistage Omega Network – Routing

- Let *s* be the binary representation of the source and *d* be that of the destination processor.
- The data traverses the link to the first switching node.
  - ***If the most significant bits of s and d are the same, then the data is routed in pass-through mode by the switch; else, it switches to crossover.***
- This process is repeated for each of the *log p* switching stages.
- Note that this is not a non-blocking switch.

Network Topologies: Multistage 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.

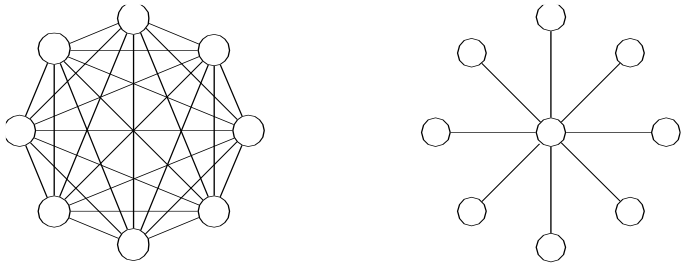
Static Interconnection Networks

Network Topologies: Completely Connected Network

- Each processor is connected to every other processor.
- The number of links in the network scales as $O(p^2)$.
- While the performance scales very well, the hardware complexity is not realizable for large values of p.
- In this sense, ***these networks are static counterparts of crossbars.***

Network Topologies: Completely Connected and Star Connected Networks

Example of an 8-node completely connected network.



(a) A completely-connected network of eight nodes;
(b) a star connected network of nine nodes.

Network Topologies:
Star Connected Network

- Every node is connected only to a common node at the center.
- Distance between any pair of nodes is *O(1).*
  - However, ***the central node becomes a bottleneck***.
- In this sense,
  - ***star connected networks are static counterparts of buses.***

Network Topologies: Linear Arrays, Meshes, and *k-d* Meshes

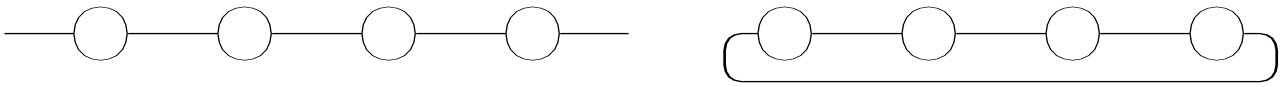
- In a linear array, each node has two neighbors, one to its left and one to its right.
  - If the nodes at either end are connected, we refer to it as a 1-D torus or a ring.

- A generalization to 2 dimensions has nodes with 4 neighbors, to the north, south, east, and west.

- A further generalization to *d* dimensions has nodes with *2d* neighbors.

- A special case of a *d*-dimensional mesh is a hypercube.
  - Here, $d = \log p$, where p is the total number of nodes.

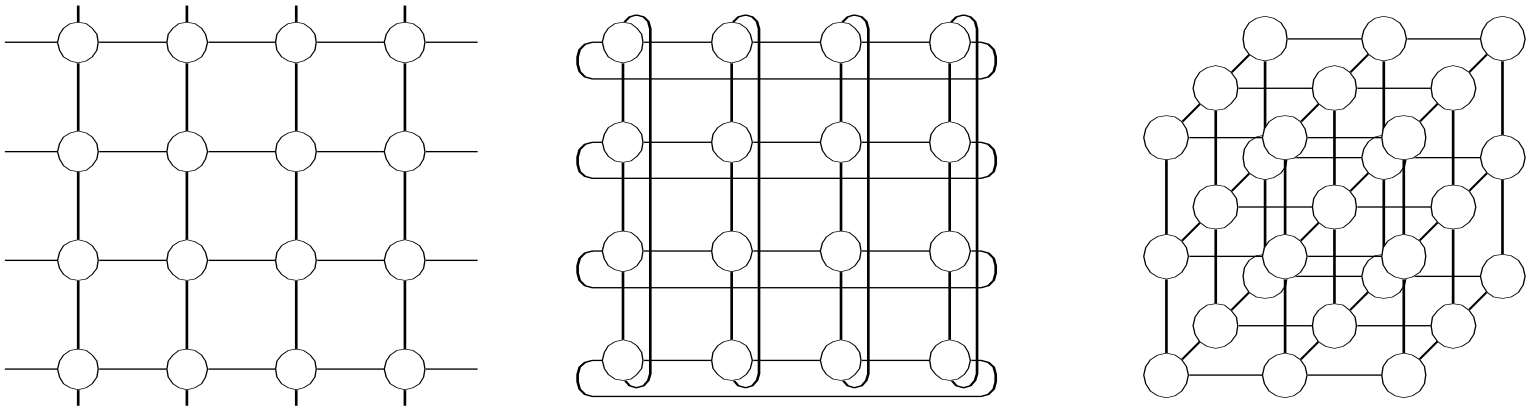
Network Topologies: Linear Arrays



(a) (b)

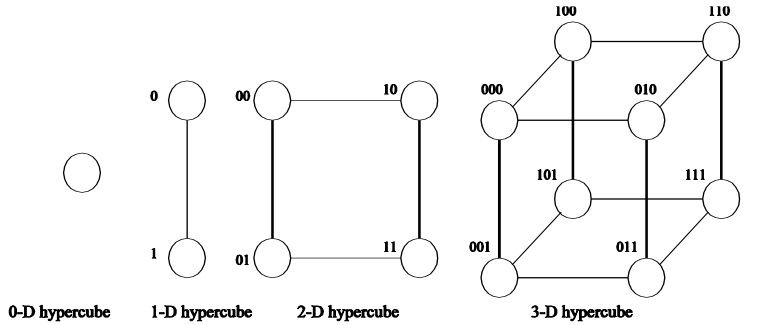
Linear arrays: (a) with no wraparound links; (b) with wraparound link.

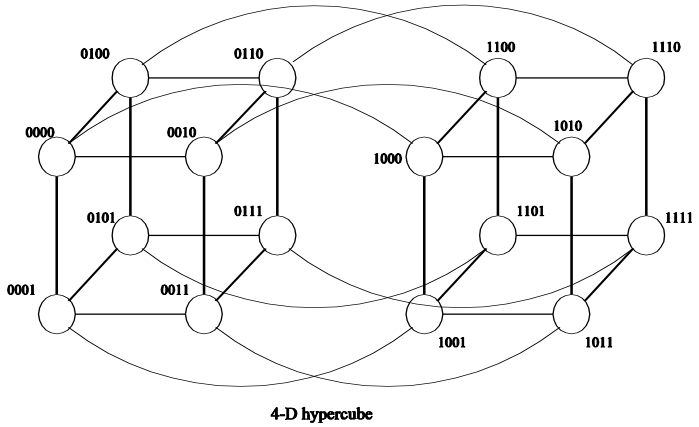
Network Topologies:
Two- and Three Dimensional Meshes



Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Network Topologies: Hypercubes and their Construction



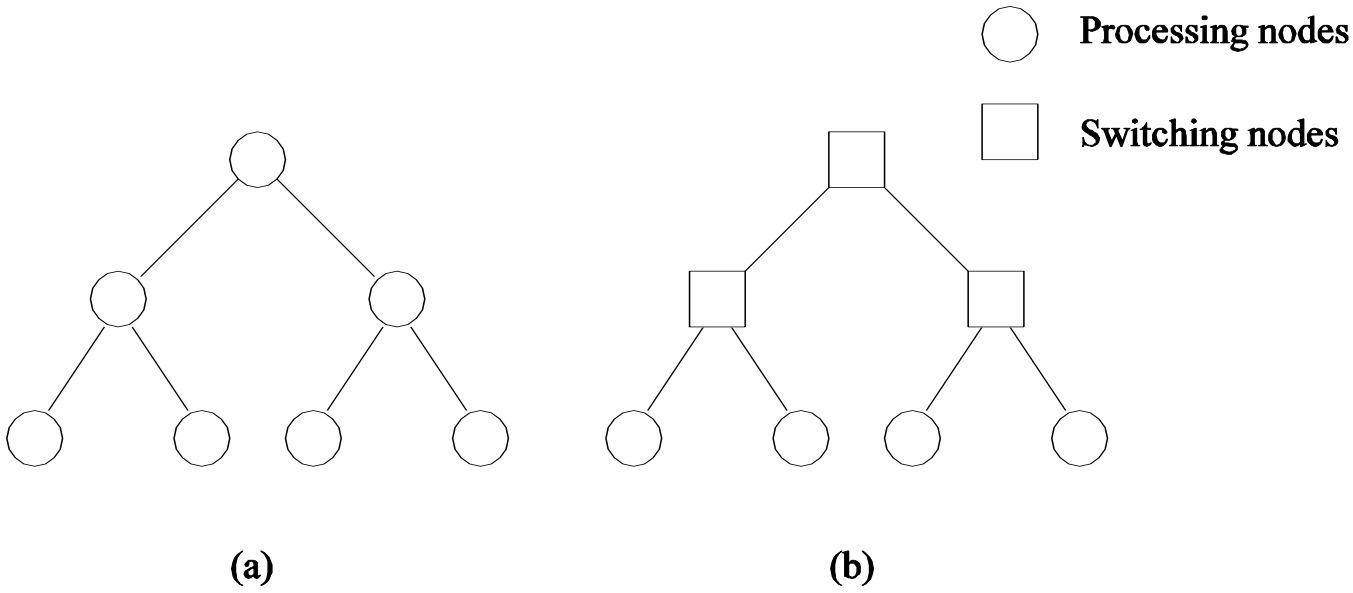


Construction of hypercubes from hypercubes of lower dimension.

Network Topologies: Properties of Hypercubes

- The distance between any two nodes is at most *log p*.
- Each node has *log p* neighbors.
- ***The distance between two nodes is given by the number of bit positions at which the two nodes differ.***

Network Topologies: Tree-Based Networks

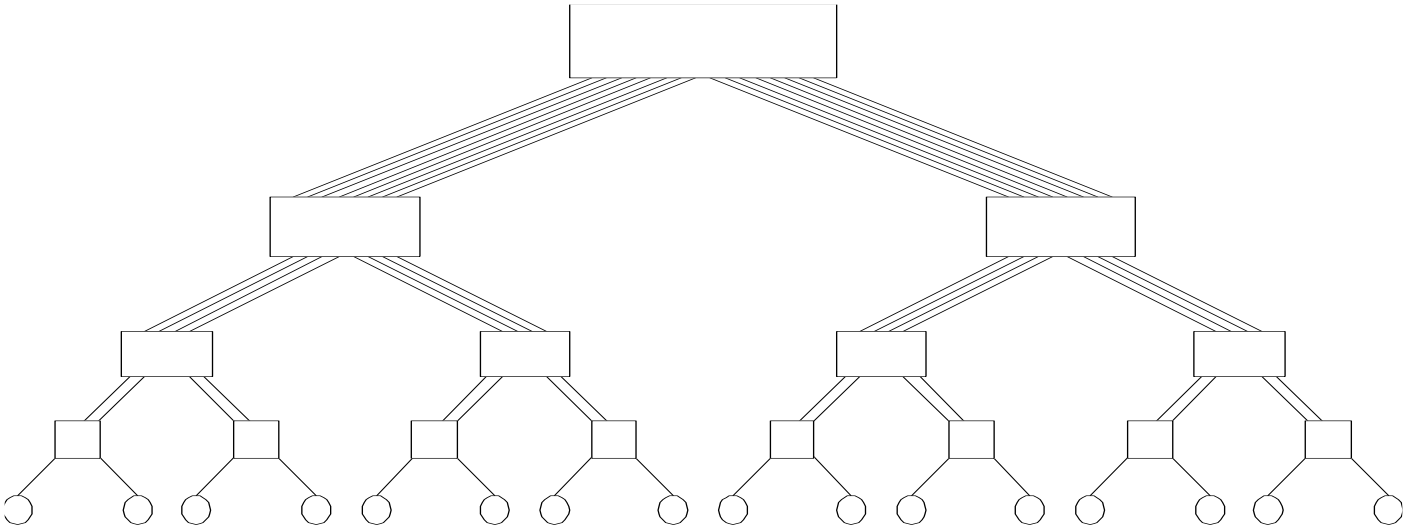


Complete binary tree networks: (a) a static tree network; and (b) a dynamic tree network.

Network Topologies: Tree Properties

- The distance between any two nodes is no more than *2logp*.
- Links higher up the tree potentially carry more traffic than those at the lower levels.
- For this reason, a variant called a fat-tree, fattens the links as we go up the tree.
- Trees can be laid out in 2D with no wire crossings. This is an attractive property of trees.

Network Topologies: Fat Trees



A fat tree network of 16 processing nodes.

Evaluating Static Interconnection Networks

- ***Diameter:*** The distance between the farthest two nodes in the network. The diameter of a linear array is $p - 1$, that of a mesh is $2(\sqrt{p} - 1)$, that of a tree and hypercube is $\log p$, and that of a completely connected network is $O(1)$.
- ***Bisection Width:*** The minimum number of wires you must cut to divide the network into two equal parts. The bisection width of a linear array and tree is 1, that of a mesh is $\sqrt{p}$, that of a hypercube is $p/2$ and that of a completely connected network is $p^2/4$.
- ***Cost:*** The number of links or switches (whichever is asymptotically higher) is a meaningful measure of the cost.
  - However, a number of other factors, such as the ability to layout the network, the length of wires, etc., also factor into the cost.

Evaluating Static Interconnection Networks

Network	Diameter	BisectionWidth	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

Evaluating Dynamic Interconnection Networks

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	$p - 1$

A graph is k arc connected if
k directed disjoint paths between any two nodes u and v exist.

Connection to design and programming

Mapping Techniques for Graphs

- Often, we need to embed a known communication pattern into a given interconnection topology.
- We may have an algorithm designed for one network, which we are porting to another topology.

For these reasons, it is useful to understand mapping between graphs.

Mapping Techniques for Graphs: Metrics

- When mapping a graph *G(V,E)* into *G'(V',E'),* the following metrics are important:
  - The maximum number of edges mapped onto any edge in *E'* is called the *congestion* of the mapping.
  - The maximum number of links in *E'* that any edge in *E* is mapped onto is called the *dilation* of the mapping.
  - The ratio of the number of nodes in the set *V'* to that in set *V* is called the *expansion* of the mapping.

Embedding a Linear Array into a Hypercube

- A linear array (or a ring) composed of 2^d nodes (labeled 0 through $2^d - 1$) can be embedded into a *d*-dimensional hypercube by mapping node *i* of the linear array onto node *G(i, d)* of the hypercube.
  The function *G(i, x)* is defined as follows:

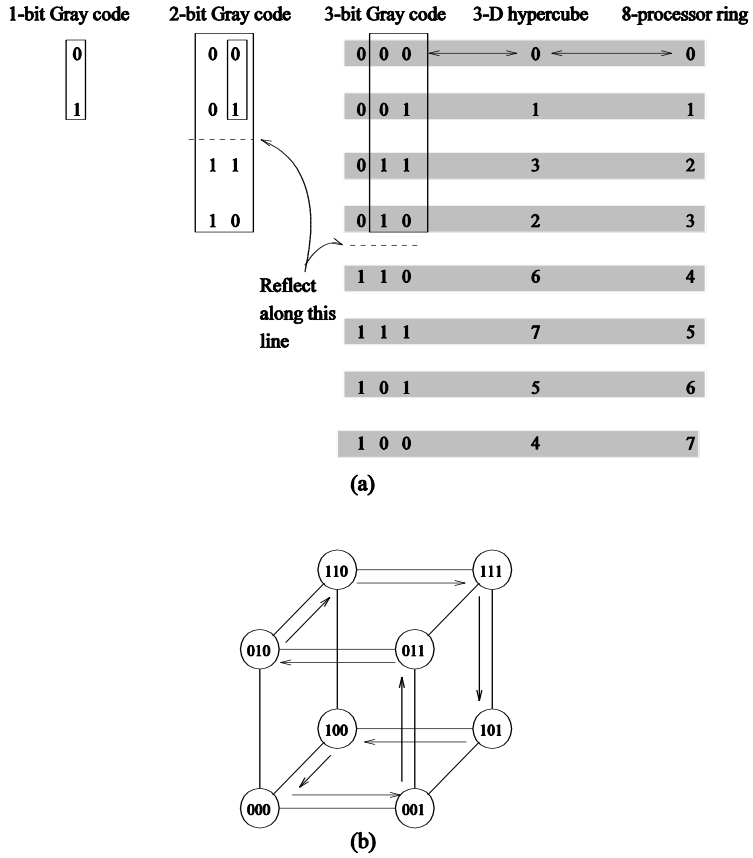
$$
\begin{aligned}
G(0,1) &= 0 \\
G(1,1) &= 1 \\
G(i,x+1) &= \begin{cases} G(i,x), & i < 2^x \\ 2^x + G(2^{x+1} - 1 - i, x), & i \geq 2^x \end{cases}
\end{aligned}
$$

Embedding a Linear Array into a Hypercube

The function *G* is called the *binary reflected Gray code* (RGC).

Since adjoining entries (*G(i, d)* and *G(i + 1, d)*) differ from each other at only one bit position, corresponding processors are mapped to neighbors in a hypercube. Therefore, the congestion, dilation, and expansion of the mapping are all 1.

Embedding a Linear Array into a Hypercube: Example



(a) A three-bit reflected Gray code ring; and (b) its embedding into a three-dimensional hypercube.

Embedding a Mesh into a Hypercube

- A $2^r \times 2^s$ wraparound mesh can be mapped to a 2^{r+s}-node hypercube by :

mapping node

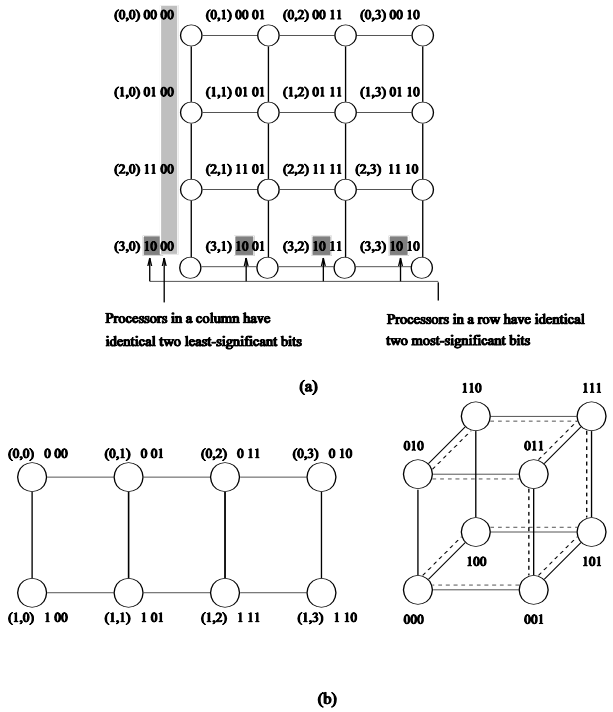
(i, j) of the mesh

onto node

$G(i, r-1) \,||\, G(j, s-1)$ of the hypercube

(where || denotes concatenation of the two Gray codes).

Embedding a Mesh into a Hypercube



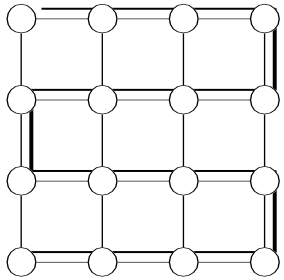
(a) A 4 × 4 mesh illustrating the mapping of mesh nodes to the nodes in a four-dimensional hypercube; and (b) a 2 × 4 mesh embedded into a three-dimensional hypercube.

Once again, the congestion, dilation, and expansion of the mapping is 1.

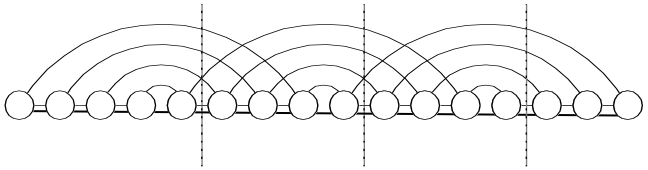
Embedding a Mesh into a Linear Array

- Since a mesh has more edges than a linear array, we will not have an optimal congestion/dilation mapping.
- We first examine the mapping of a linear array into a mesh and then invert this mapping.
- This gives us an optimal mapping (in terms of congestion).

Embedding a Mesh into a Linear Array: Example



(a) Mapping a linear array into a 2D mesh (congestion 1).



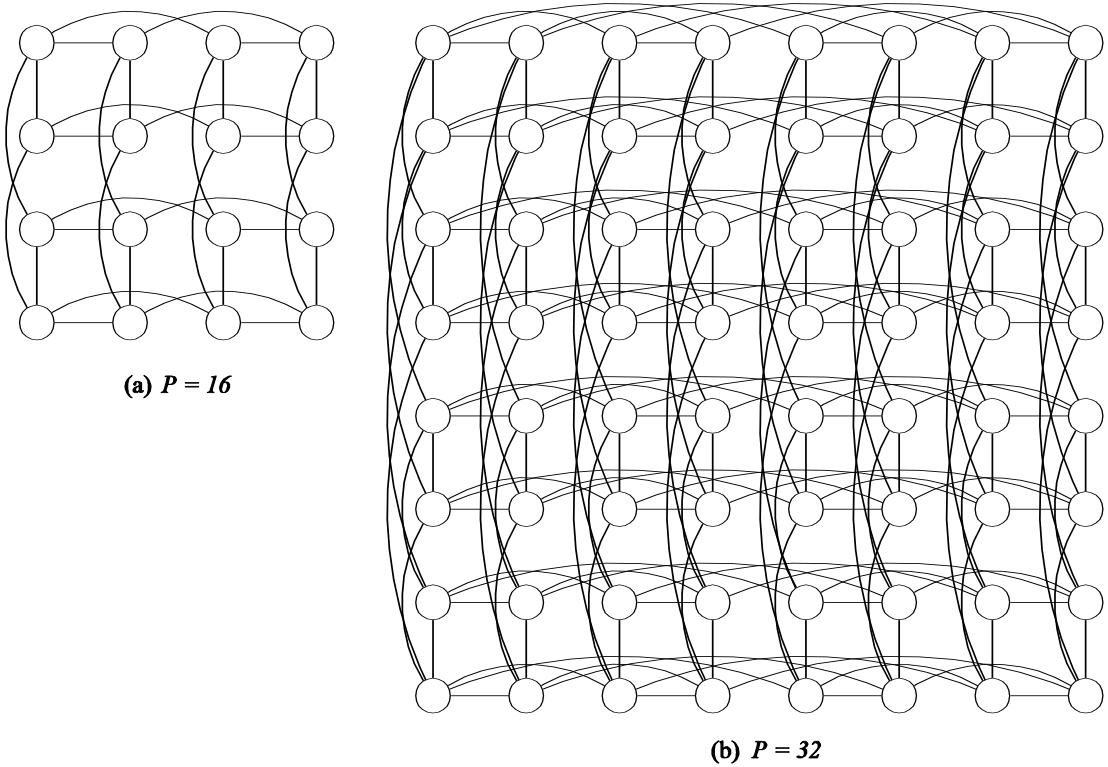
(b) Inverting the mapping - mapping a 2D mesh into a linear array (congestion 5)

(a) Embedding a 16 node linear array into a 2-D mesh; and (b) the inverse of the mapping. Solid lines correspond to links in the linear array and normal lines to links in the mesh.

Embedding a Hypercube into a 2-D Mesh

- Each $\sqrt{p}$ node subcube of the hypercube is mapped to a $\sqrt{p}$ node row of the mesh.
- This is done by inverting the linear-array to hypercube mapping.
- This can be shown to be an optimal mapping.

Embedding a Hypercube into a 2-D Mesh: Example

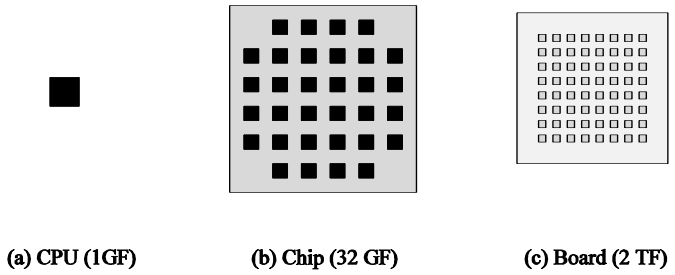


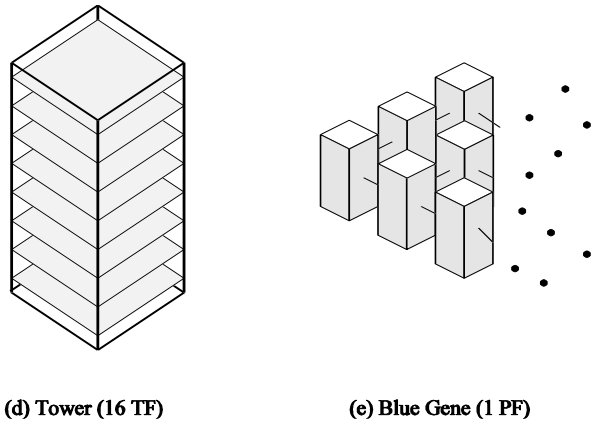
(a) $P = 16$

(b) $P = 32$

Embedding a hypercube into a 2-D mesh.

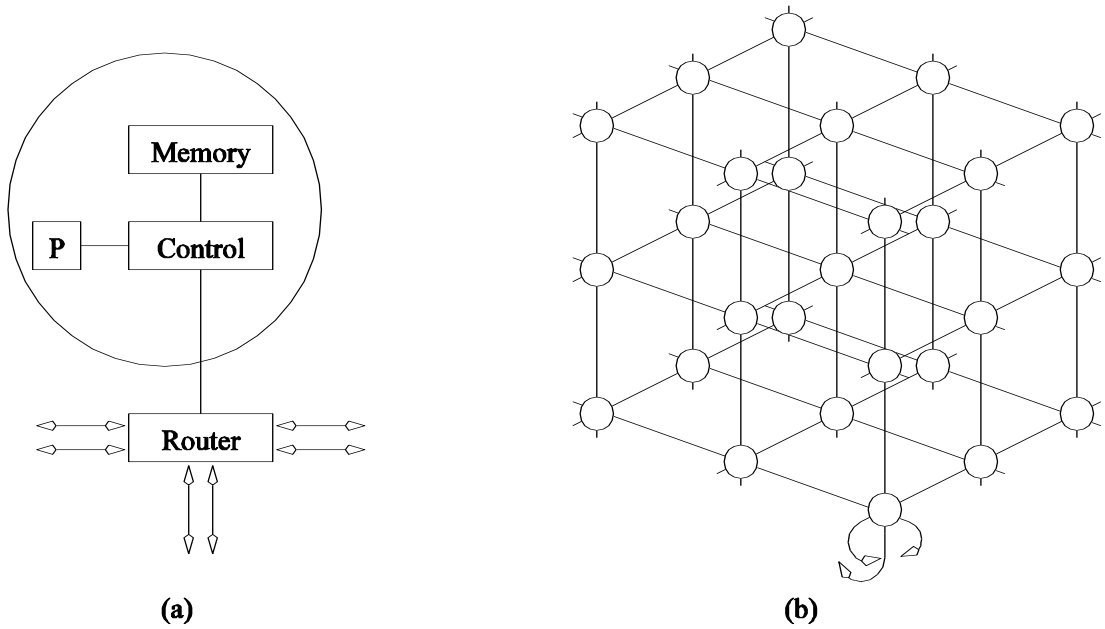
Case Studies:
The IBM Blue-Gene Architecture





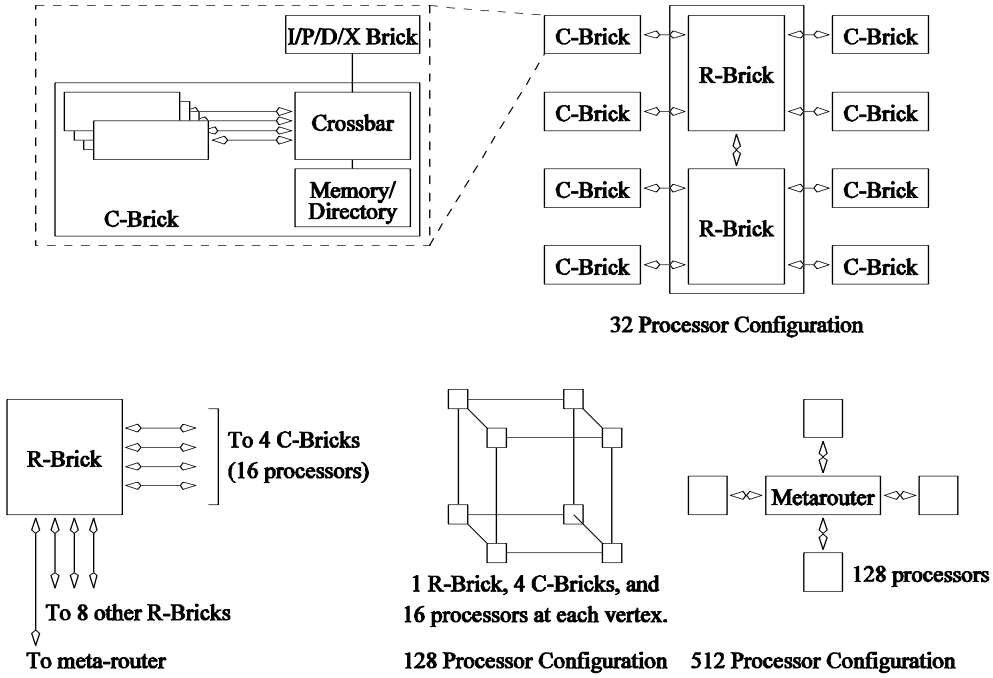
The hierarchical architecture of Blue Gene.

Case Studies:
The Cray T3E Architecture



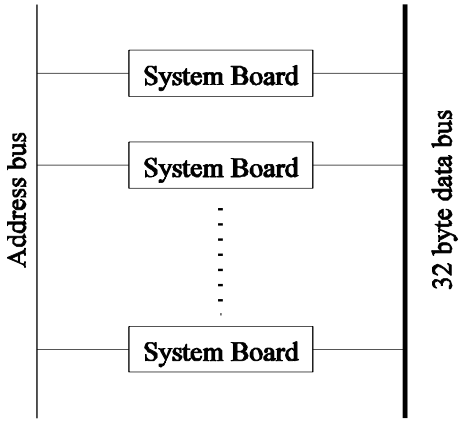
Interconnection network of the Cray T3E:
(a) node architecture; (b) network topology.

Case Studies:
The SGI Origin 3000 Architecture

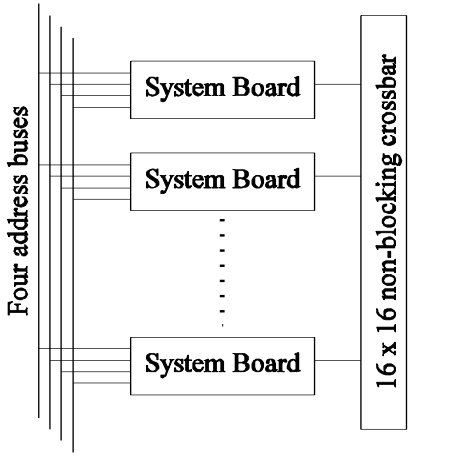


Architecture of the SGI Origin 3000 family of servers.

Case Studies:
The Sun HPC Server Architecture



Sun Ultra 6000 (6 - 30 processors)



Starfire Ultra 1000 (up to 64 processors)

Architecture of the Sun Enterprise family of servers.

Communication Costs in Parallel Machines

- Along with idling and contention, communication is a major overhead in parallel programs.
- The cost of communication is dependent on a variety of features including the programming model semantics, the network topology, data handling and routing, and associated software protocols.

Message Passing Costs in Parallel Computers

- The total time to transfer a message over a network comprises of the following:
  - *Startup time* (t_s): Time spent at sending and receiving nodes (executing the routing algorithm, programming routers, etc.).
  - *Per-hop time* (t_h): This time is a function of number of hops and includes factors such as switch latencies, network delays, etc.
  - *Per-word transfer time* (t_w): This time includes all overheads that are determined by the length of the message.
    - This includes bandwidth of links, error checking and correction, etc.

Store-and-Forward Routing

- A message traversing multiple hops is completely received at an intermediate hop before being forwarded to the next hop.

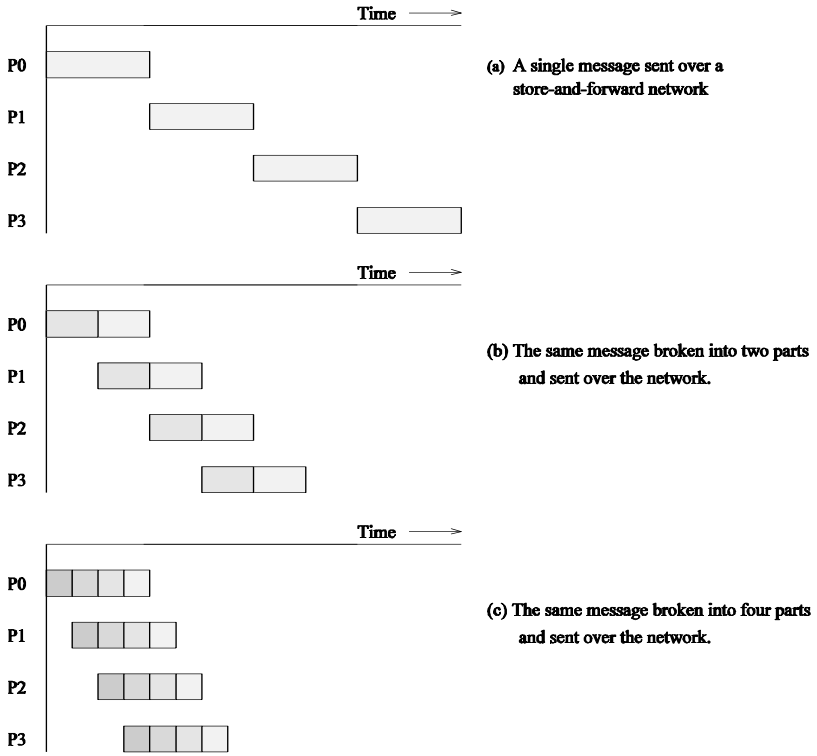
- The total communication cost for a message of size *m* words to traverse *l* communication links is

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

- In most platforms, t_h is small and the above expression can be approximated by

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

Routing Techniques



Passing a message from node P_0 to P_3 (a) through a ***store-and-forward*** communication network; (b) and (c) extending the concept to ***cut-through routing***. The shaded regions represent the time that the message is in transit. The startup time associated with this message transfer is assumed to be zero.

Packet Routing

- Store-and-forward makes poor use of communication resources.
- Packet routing breaks messages into packets and pipelines them through the network.
- Since ***packets may take different paths***, each packet must carry routing information, error checking, sequencing, and other related header information.
- The total communication time for packet routing is approximated by:

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

- The factor t_w accounts for overheads in packet headers.

Cut-Through Routing

- Takes the concept of packet routing to an extreme by further dividing messages into basic units called ***flits***.
- Since flits are typically small, the header information must be minimized.
- This is done by ***forcing all flits to take the same path***, in sequence.
- A tracer message first programs all intermediate routers.
  All flits then take the same route.
- ***Error checks are performed on the entire message***, as opposed to flits.
- No sequence numbers are needed.

Cut-Through Routing

- The total communication time for cut-through routing is approximated by:

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

- This is identical to packet routing, however, t_w is typically much smaller.

Simplified Cost Model for Communicating Messages

- The cost of communicating a message between two nodes *l* hops away using cut-through routing is given by

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

- In this expression, t_h is typically smaller than t_s and t_w. For this reason, the second term does not show, particularly, when *m* is large.
- Furthermore, it is often not possible to control routing and placement of tasks.
- For these reasons, we can approximate the cost of message transfer by

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

Simplified Cost Model for Communicating Messages =>congestion problem

- It is important to note that the original expression for communication time is ***valid for only uncongested networks***.
- If a link takes multiple messages, the corresponding t_w term must be scaled up by the number of messages.
- Different communication patterns congest different networks to varying extents.
- It is important to understand and account for this in the communication time accordingly.

Cost Models for Shared Address Space Machines

- While the basic messaging cost applies to these machines as well, a number of other factors make accurate cost modeling more difficult.

- Memory layout is typically determined by the system.
- Finite cache sizes can result in cache thrashing.
- Overheads associated with invalidate and update operations are difficult to quantify.
- Spatial locality is difficult to model.
- Prefetching can play a role in reducing the overhead associated with data access.
- False sharing and contention are difficult to model.

Routing algorithms

How does one compute the route that a message takes from source to destination?

1) Arithmetic

2) Based on source selection

3) With a table (*table lookup*)

4) Adaptive: The route is determined by the state of the network (taking into account the contention)

Routing

- Given a current node and a destination node, the routing algorithm chooses the next port and channel on which to send out the message.
- A routing algorithm is a function R: N x N -> C.
  - A switch usually uses one of first three mechanisms to determine the output channel from info in the packet header.
  - A switch needs to route a packet every few cycles, so it needs to be fast.
  - In regular topologies, simple arithmetic suffices.

Example

routing in a grid.

Δx =d.x-s.x, Δ y=d.y-s.y

- West (–x)
- East (+x)
- South (–y)
- North (+y)
- Processor (destination) Δ x = 0, Δ y = 0
- To accomplish this routing, the switch needs to test the address in the header and increment or decrement one routing field.
- Usually, routing is done in dimension order—first across the x dimension, then the y-dimension, then the z-dimension (if any), etc.
- .

- In a binary cube, the switch determines the most significant bit where the destination node number differs from the current node number
- Sometimes a packet header has a relative address embedded in it.

Example:

- If the source node is 001010 and the destination node is 100101, what would be the relative address embedded at the source node?

=>the switch just looks for the first non-zero bit and routes accordingly.

Routing in dimension order.

- In general, in a mesh or cube, routing is done by moving from lowest (x) dimension to the highest.
- In a hypercube, it is called e-cube routing.

Source-based routing.

- Generally, the source builds a header consisting of the output port # for each switch along the route.
- Each switch just strips off the port number from the front of the message and sends it on.
- All of the intelligence is at the source node.
- Disadvantage: Header is large, of variable size.

Table-driven routing

Associates a small routing table at each switch.

- It allows for a small fixed-size header.
- The packet header contains a routing field i.
- The output port is R[i], where R is the routing table.
- Usually the table also contains the routing field for the next step in the route.

Disadvantage: The switch must contain quite a bit of routing state. Fairly large tables are needed even for simple routing algorithms.

This approach was used by ATM and HPPI switches, but isn't too practical for multiprocessors, because of the large number of routing patterns that they must support.

One important difference between network routers and multiprocessor switches: Time constraints.

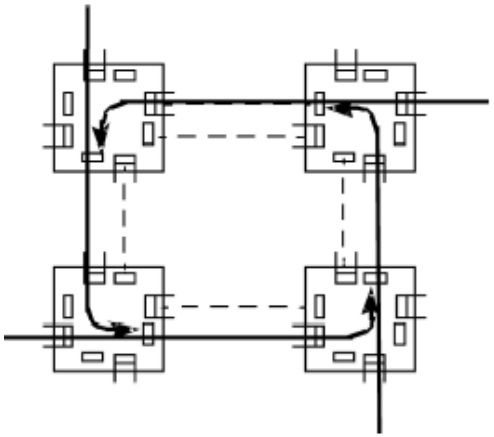
Requirements

Routing should:

- prevent deadlocks - for this reason it coul be used
  - dimension-ordered (Common approach to routing for *k*-ary *n*-cubes or
  - e-cube routing could be used.
- avoid hot-spots - for this reason, ***two-step routing*** is often used.
  - In this case, a message from source *s* to destination *d* is first sent to a randomly chosen intermediate processor *i* and then forwarded to destination *d*.

Deadlock-free routing

- Deadlock occurs when two or more packets are “circularly” waiting for resources that are held by other packets in the group.



- The diagram at the right illustrates how this can occur with 2-hop messages.
- Each packet is waiting for a link occupied by another packet.

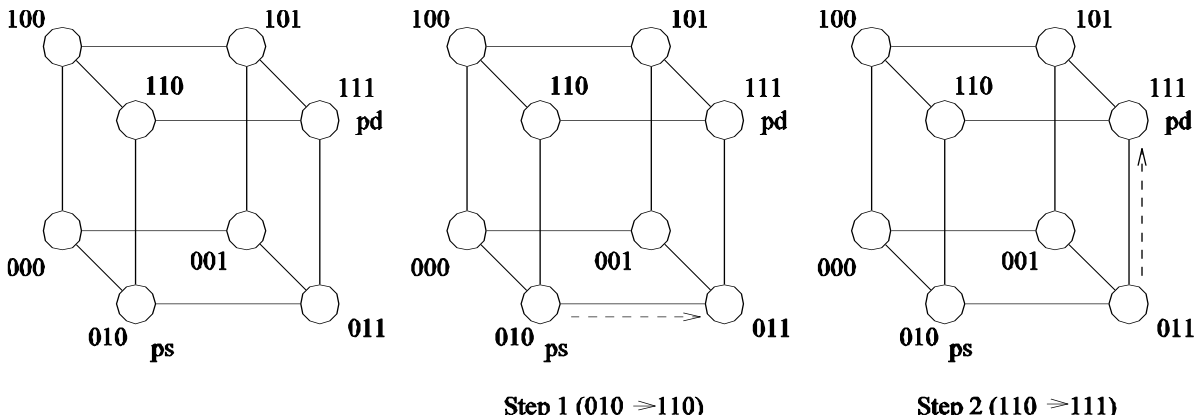
Deadlock

What conditions are necessary for deadlock to occur?

- a shared resource
- that is incrementally allocated and
- non-preemptible.

A channel is a shared resource, and channels are acquired incrementally, as a route is built up.

Routing Mechanisms for Interconnection Networks



Routing a message from node P_s (010) to node P_d (111) in a three-dimensional hypercube