Introduction to Parallel Computing

George Karypis

Parallel Programming Platforms

Elements of a Parallel Computer

- Hardware
  - Multiple Processors
  - Multiple Memories
  - Interconnection Network
- System Software
  - Parallel Operating System
  - Programming Constructs to Express/Orchestrate Concurrency
- Application Software
  - Parallel Algorithms

Goal:

Utilize the Hardware, System, & Application Software to either

- Achieve Speedup: $T_p = T_s/p$
- Solve problems requiring a large amount of memory.

Parallel Computing Platform

- Logical Organization
  - The user's view of the machine as it is being presented via its system software
- Physical Organization
  - The actual hardware architecture

- Physical Architecture is to a large extent independent of the Logical Architecture

Logical Organization Elements

- Control Mechanism
  - SISD/SIMD/MIMD/MISD
    - Single/Multiple Instruction Stream & Single/Multiple Data Stream
  - SPMD:
    Single Program Multiple Data

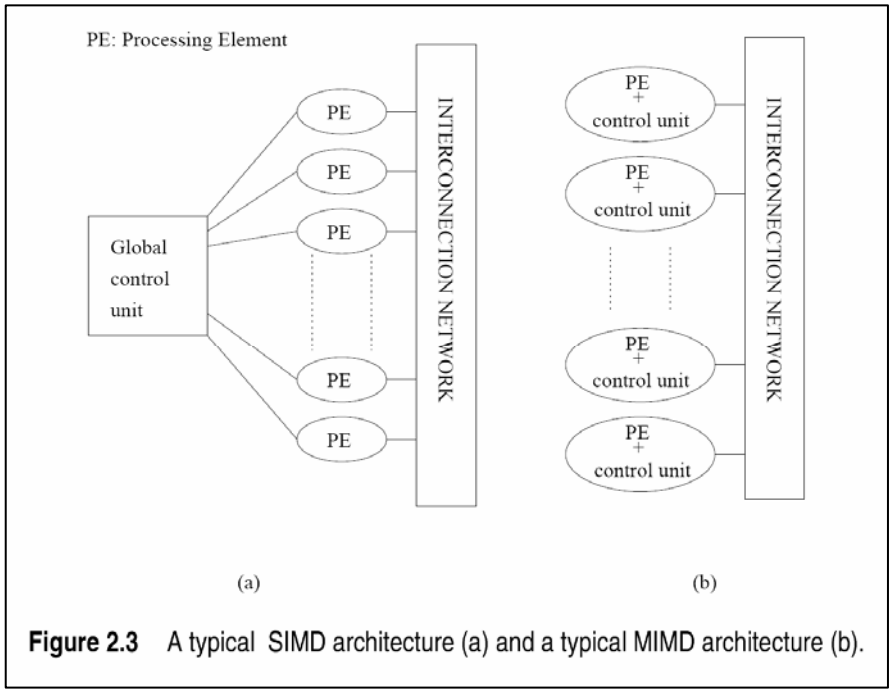


Figure 2.3 A typical SIMD architecture (a) and a typical MIMD architecture (b).

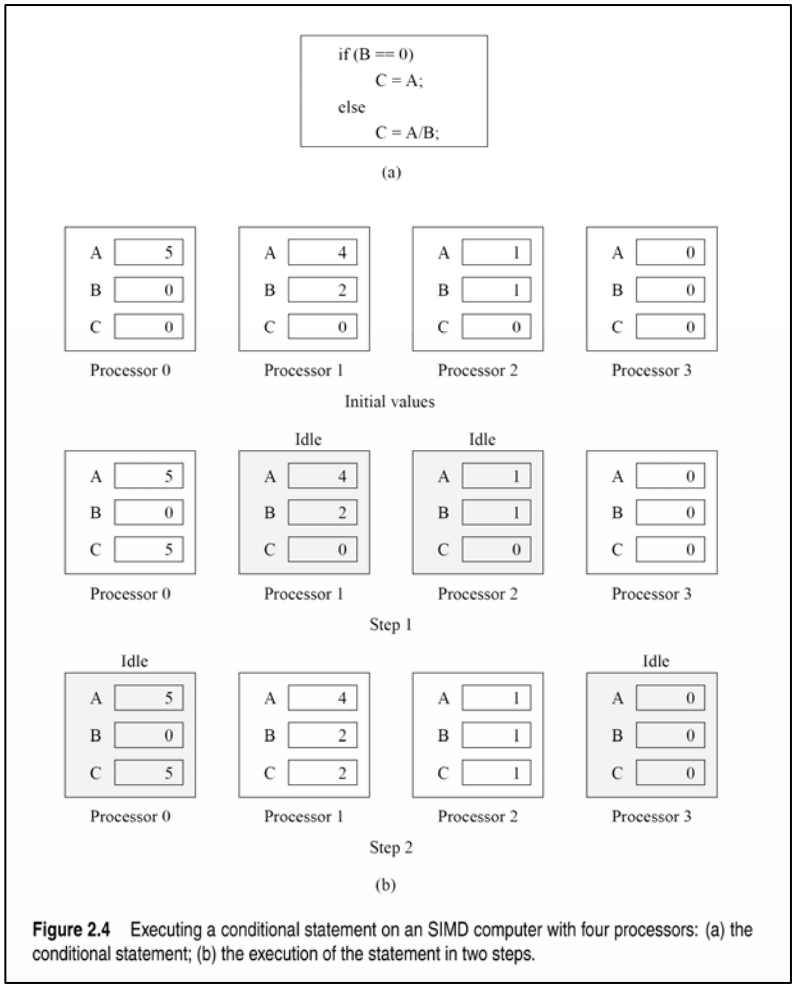


Figure 2.4 Executing a conditional statement on an SIMD computer with four processors: (a) the conditional statement; (b) the execution of the statement in two steps.

Logical Organization Elements

- Communication Model
  - Shared-Address Space
    - UMA/NUMA/ccNUMA
  - Message-Passing

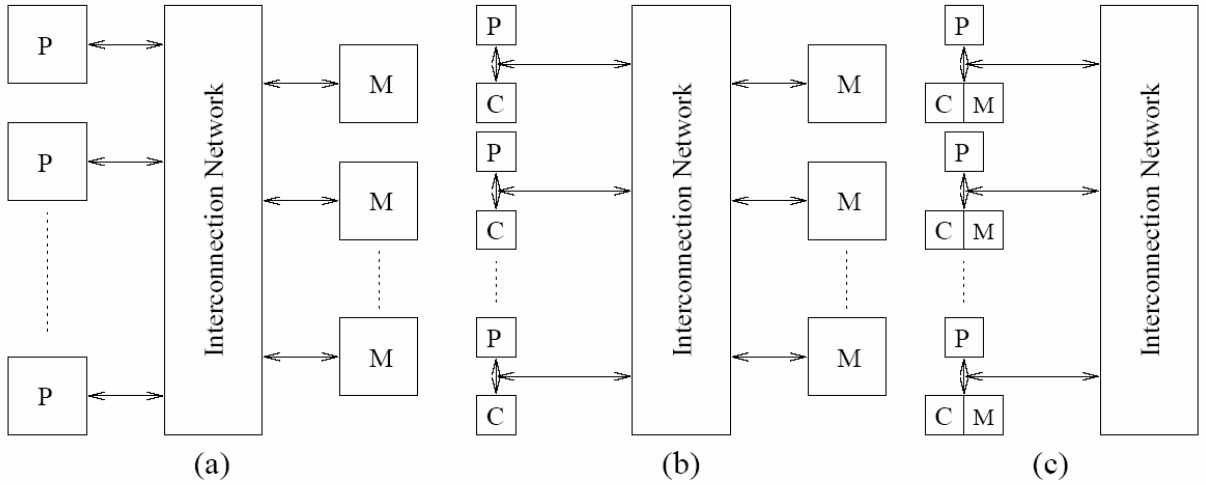


Figure 2.5 Typical shared-address-space architectures: (a) Uniform-memory-access shared-address-space computer; (b) Uniform-memory-access shared-address-space computer with caches and memories; (c) Non-uniform-memory-access shared-address-space computer with local memory only.

Physical Organization

- Ideal Parallel Computer Architecture
  - PRAM: Parallel Random Access Machine
- PRAM Models
  - EREW/ERCW/CREW/CRCW
    - Exclusive/Concurrent Read and/or Write
  - Concurrent Writes are resolved via
    - Common/Arbitrary/Priority/Sum

Physical Organization

- Interconnection Networks (ICNs)
  - Provide processor-to-processor and processor-to-memory connections
  - Networks are classified as:
    - Static
      - Consist of a number of point-to-point links
        - direct network
      - Historically used to link processors-to-processors
        - distributed-memory system
    - Dynamic
      - The network consists of switching elements that the various processors attach to
        - indirect network
      - Historically used to link processors-to-memory
        - shared-memory systems

Static & Dynamic ICNs

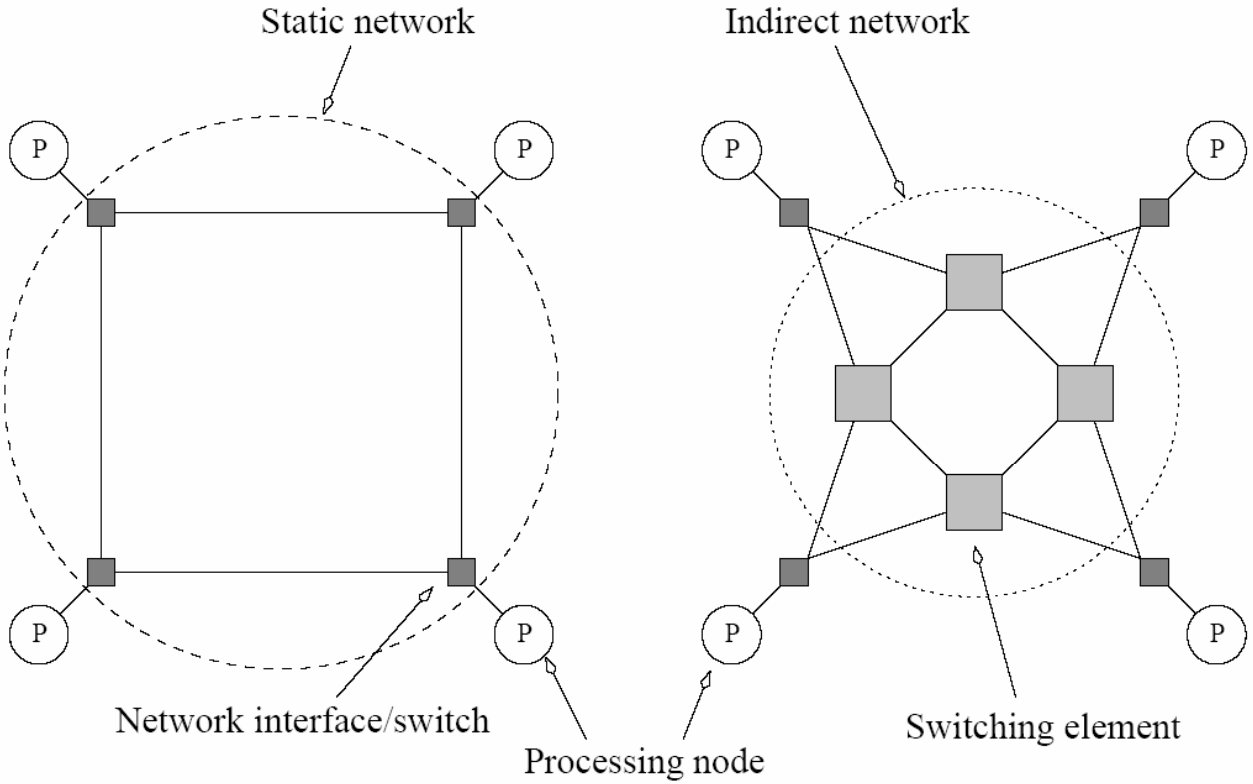


Figure 2.6 Classification of interconnection networks: (a) a static network; and (b) a dynamic network.

Evaluation Metrics for ICNs

- Diameter
  - The maximum distance between any two nodes
    - Smaller the better.
- Connectivity
  - The minimum number of arcs that must be removed to break it into two disconnected networks
    - Larger the better
  - Measures the multiplicity of paths
- Bisection width
  - The minimum number of arcs that must be removed to partition the network into two equal halves.
    - Larger the better
- Bisection bandwidth
  - Applies to networks with weighted arcs—weights correspond to the *link width* (how much data it can transfer)
  - The minimum volume of communication allowed between any two halves of a network
    - Larger the better
- Cost
  - The number of links in the network
    - Smaller the better

Metrics and Dynamic Networks

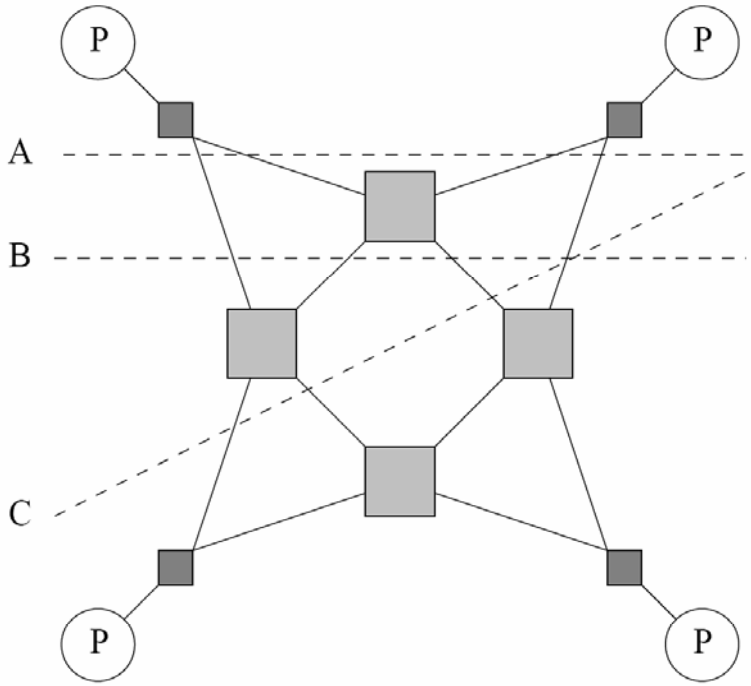
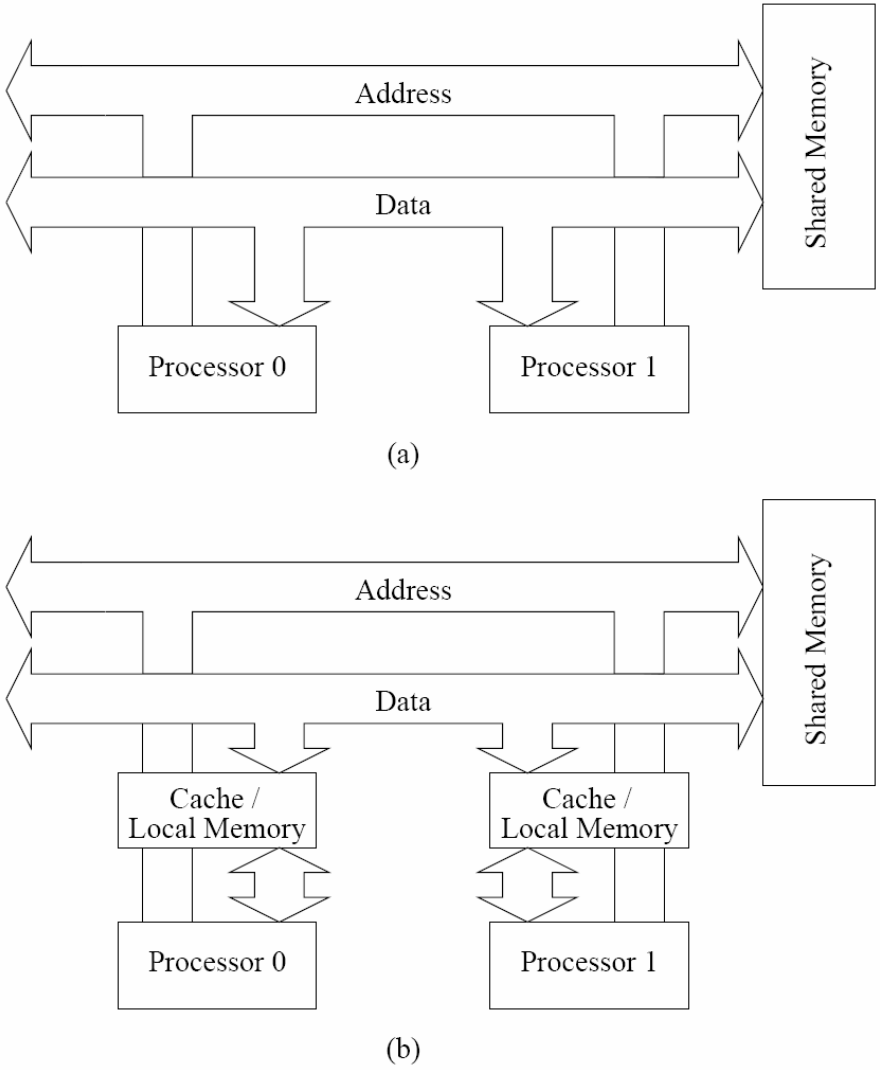


Figure 2.20 Bisection width of a dynamic network is computed by examining various equi-partitions of the processing nodes and selecting the minimum number of edges crossing the partition. In this case, each partition yields an edge cut of four. Therefore, the bisection width of this graph is four.

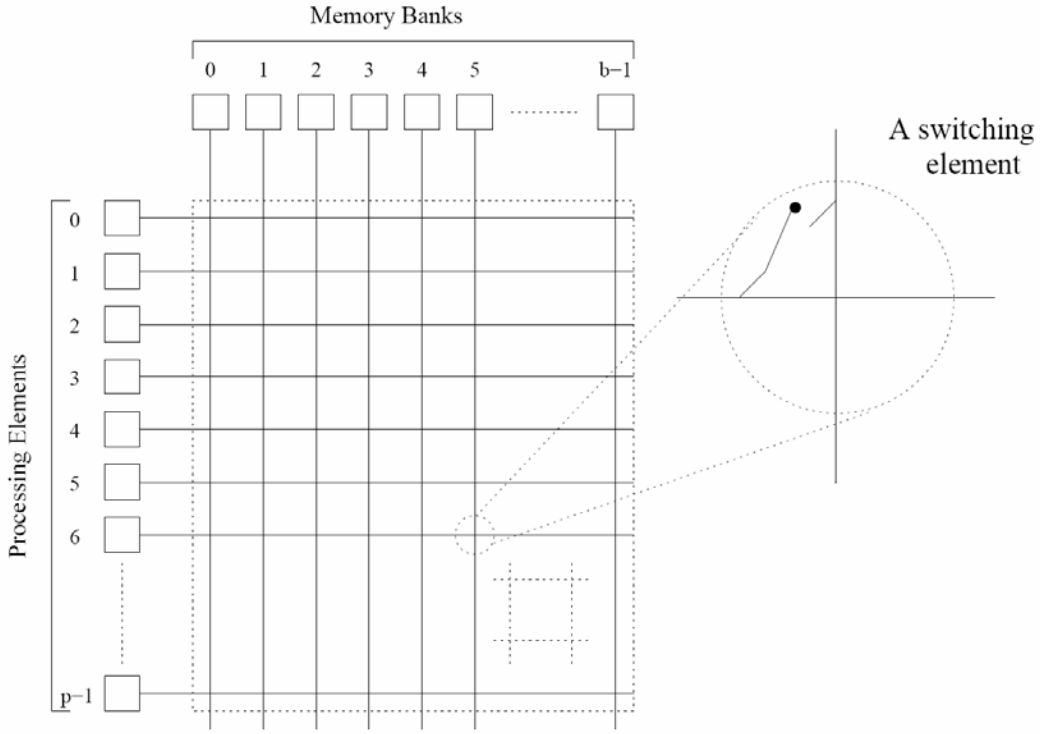
Network Topologies

- Bus-Based Networks
  - Shared medium
  - Information is being broadcasted
  - Evaluation:
    - Diameter: O(1)
    - Connectivity: O(1)
    - Bisection width: O(1)
    - Cost: O(p)



Network Topologies

- Crossbar Networks
  - Switch-based network
  - Supports simultaneous connections
  - Evaluation:
    - Diameter: O(1)
    - Connectivity: O(1)?
    - Bisection width: O(p)?
    - Cost: $O(p^2)$



Network Topologies

- Multistage Interconnection Networks

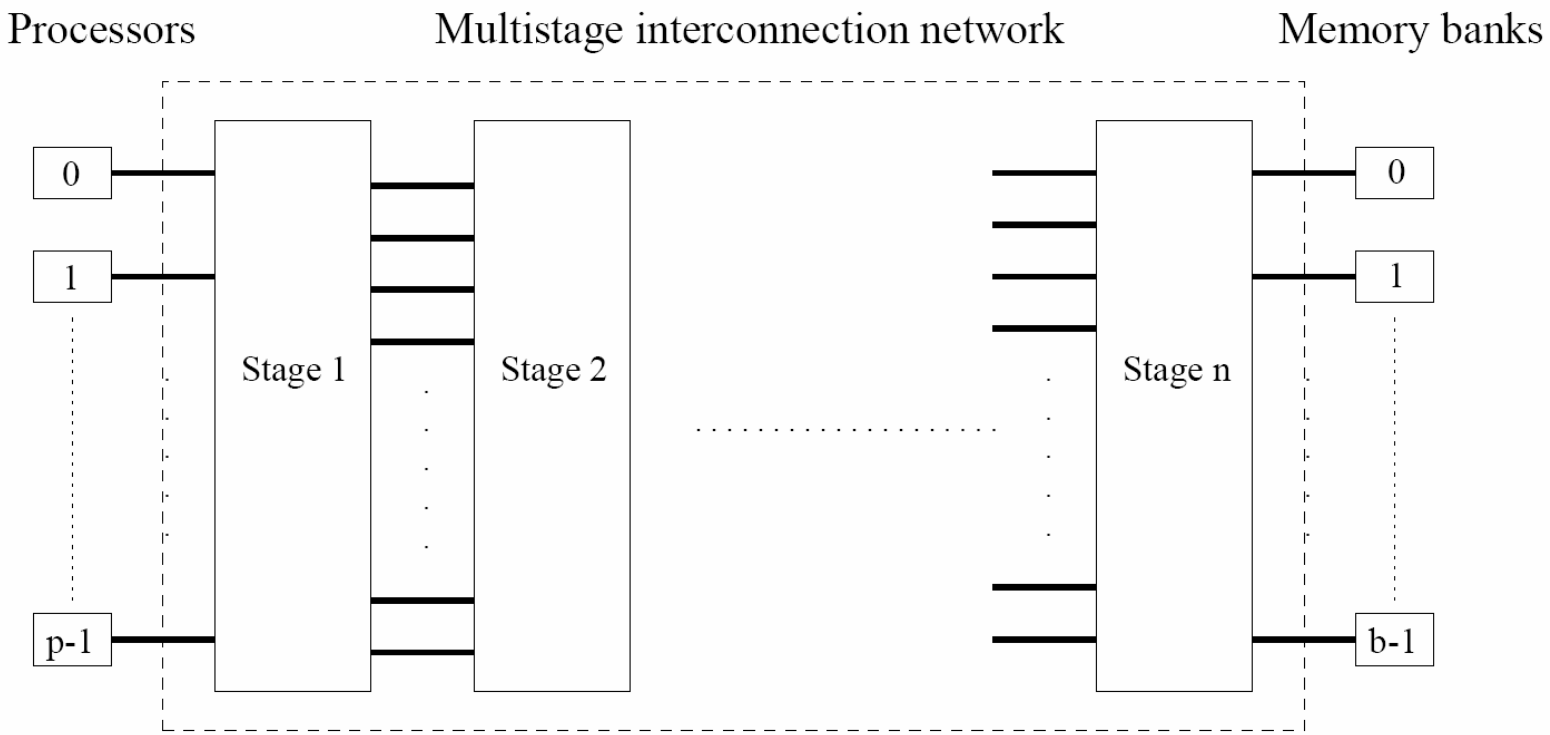
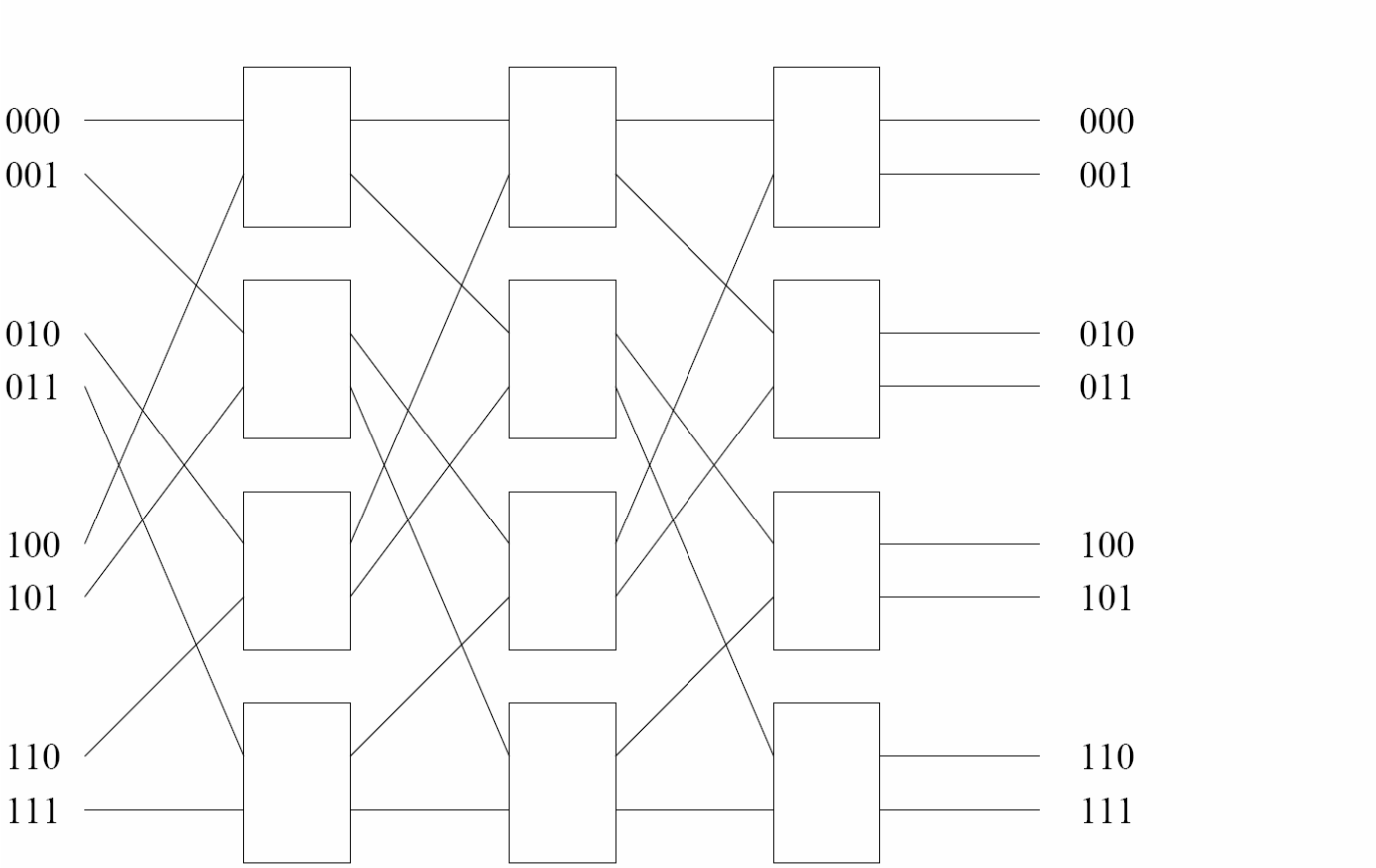
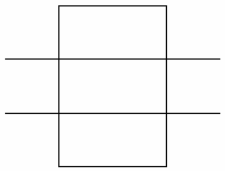


Figure 2.9 The schematic of a typical multistage interconnection network.

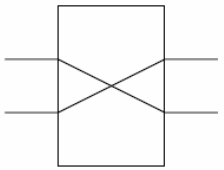
Multistage Switch Architecture



A complete omega network connecting eight inputs and eight outputs.



Pass-through



Cross-over

Connecting the Various Stages

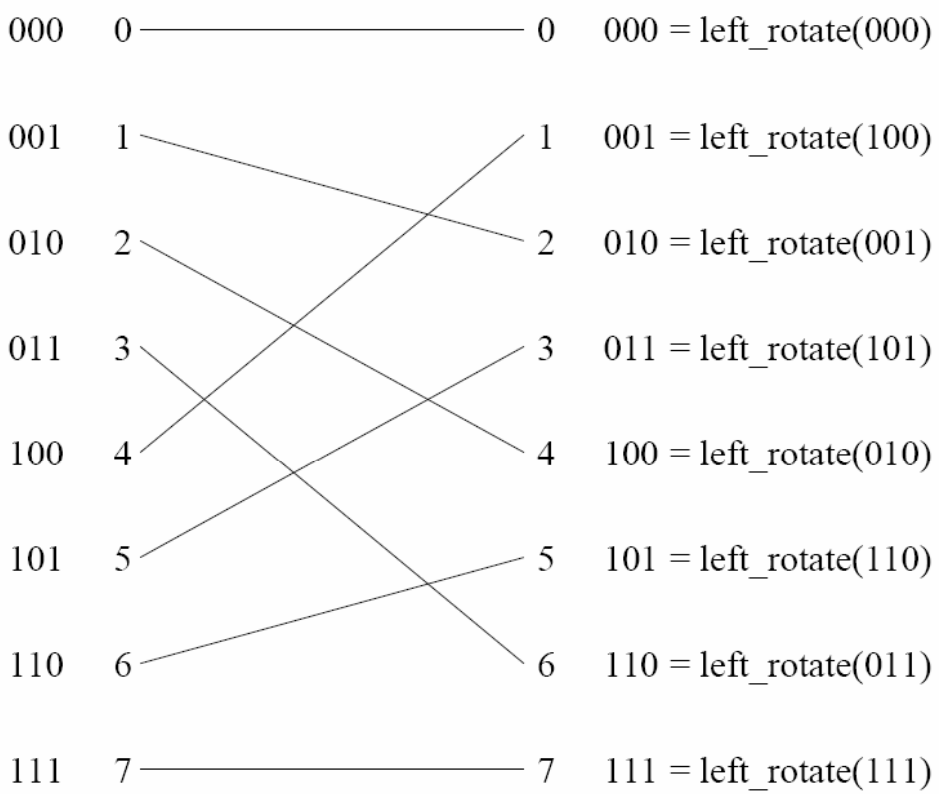


Figure 2.10 A perfect shuffle interconnection for eight inputs and outputs.

Blocking in a Multistage Switch

Routing is done by comparing the bit-level representation of source and destination addresses.
-match goes via pass-through
-mismatch goes via cross-over

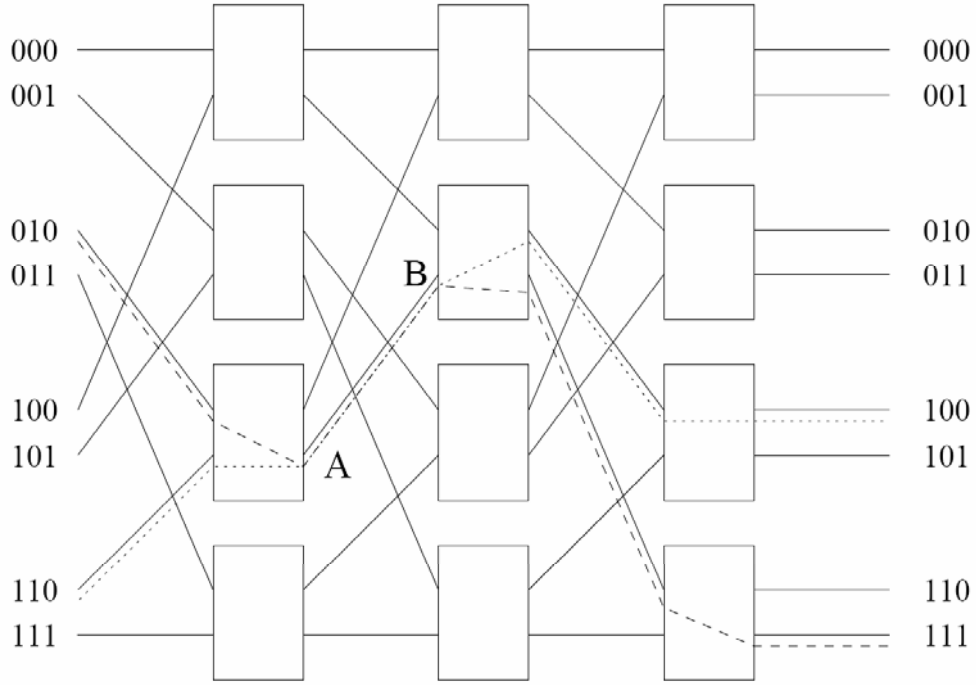


Figure 2.13 An example of blocking in omega network: one of the messages (010 to 111 or 110 to 100) is blocked at link AB.

Network Topologies

- Complete and star-connected networks.

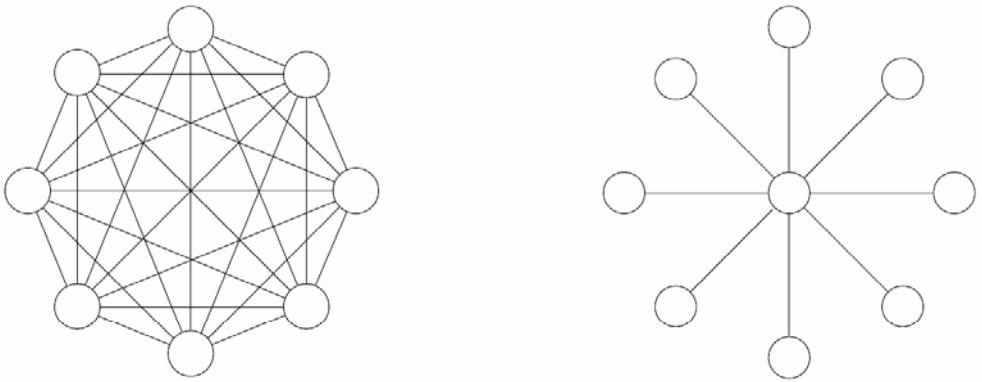


Figure 2.14 (a) A completely-connected network of eight nodes; (b) a Star connected network of nine nodes.

Network Topologies

- Cartesian Topologies

Figure 2.15 Linear arrays: (a) with no wraparound links; (b) with wraparound link.

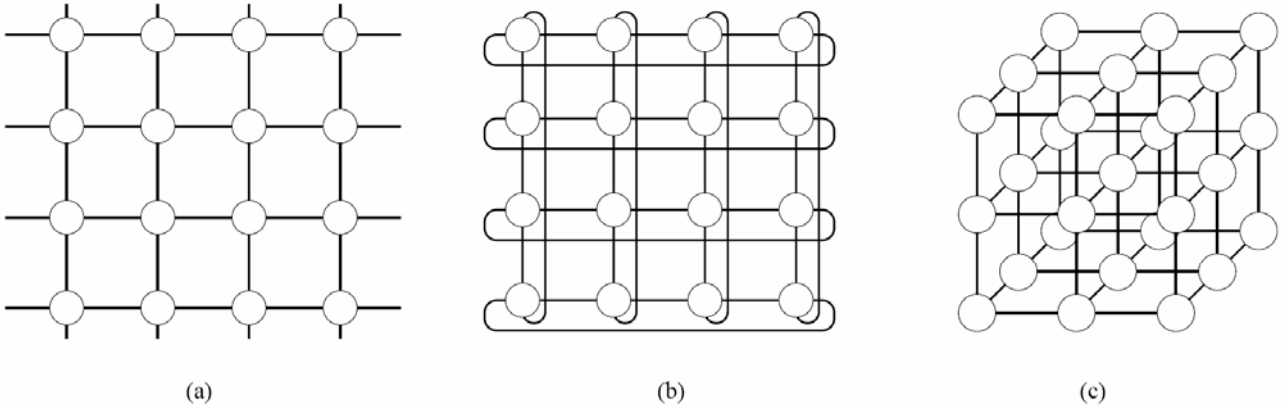


Figure 2.16 Two and three dimensional meshes: (a) 2-D mesh with no wraparound; (b) 2-D mesh with wraparound link (2-D torus); and (c) a 3-D mesh with no wraparound.

Network Topologies

- Hypercubes

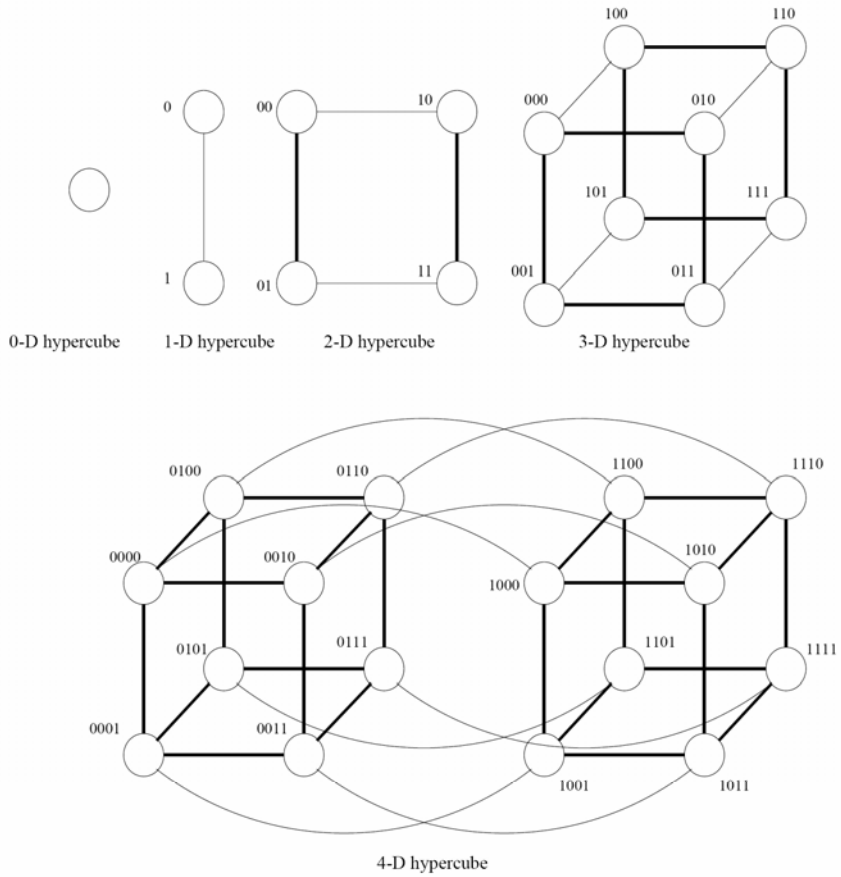
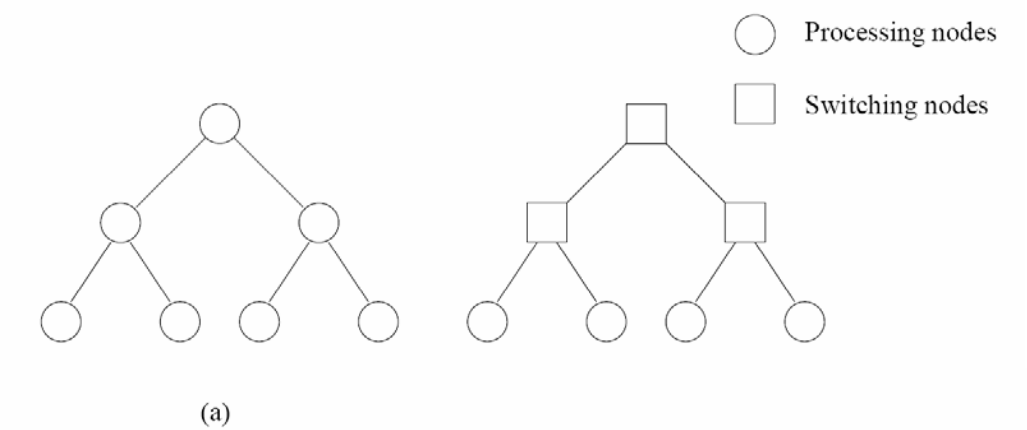
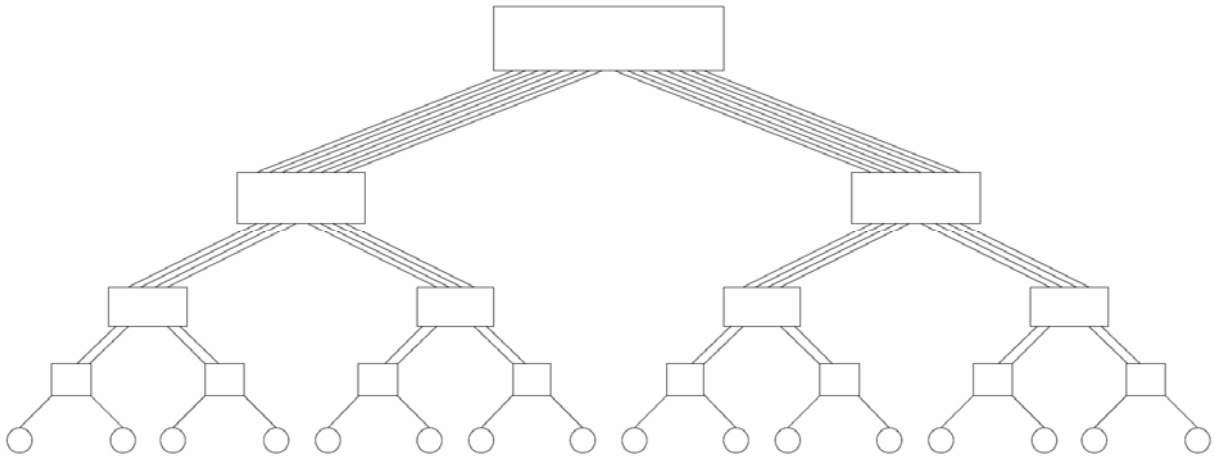


Figure 2.17 Construction of hypercubes from hypercubes of lower dimension.

Network Topologies

- Trees





Summary of Performance Metrics

Table 2.1 A summary of the characteristics of various static network topologies connecting p nodes.

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

Table 2.2 A summary of the characteristics of various dynamic network topologies connecting p processing nodes.

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	$p-1$

Physical Organization

- Cache Coherence in Shared Memory Systems
  - A certain level of consistency must be maintained for multiple copies of the same data
  - Required to ensure proper semantics and correct program execution
    - serializability
  - Two general protocols for dealing with it
    - invalidate & update

Invalidate/Update Protocols

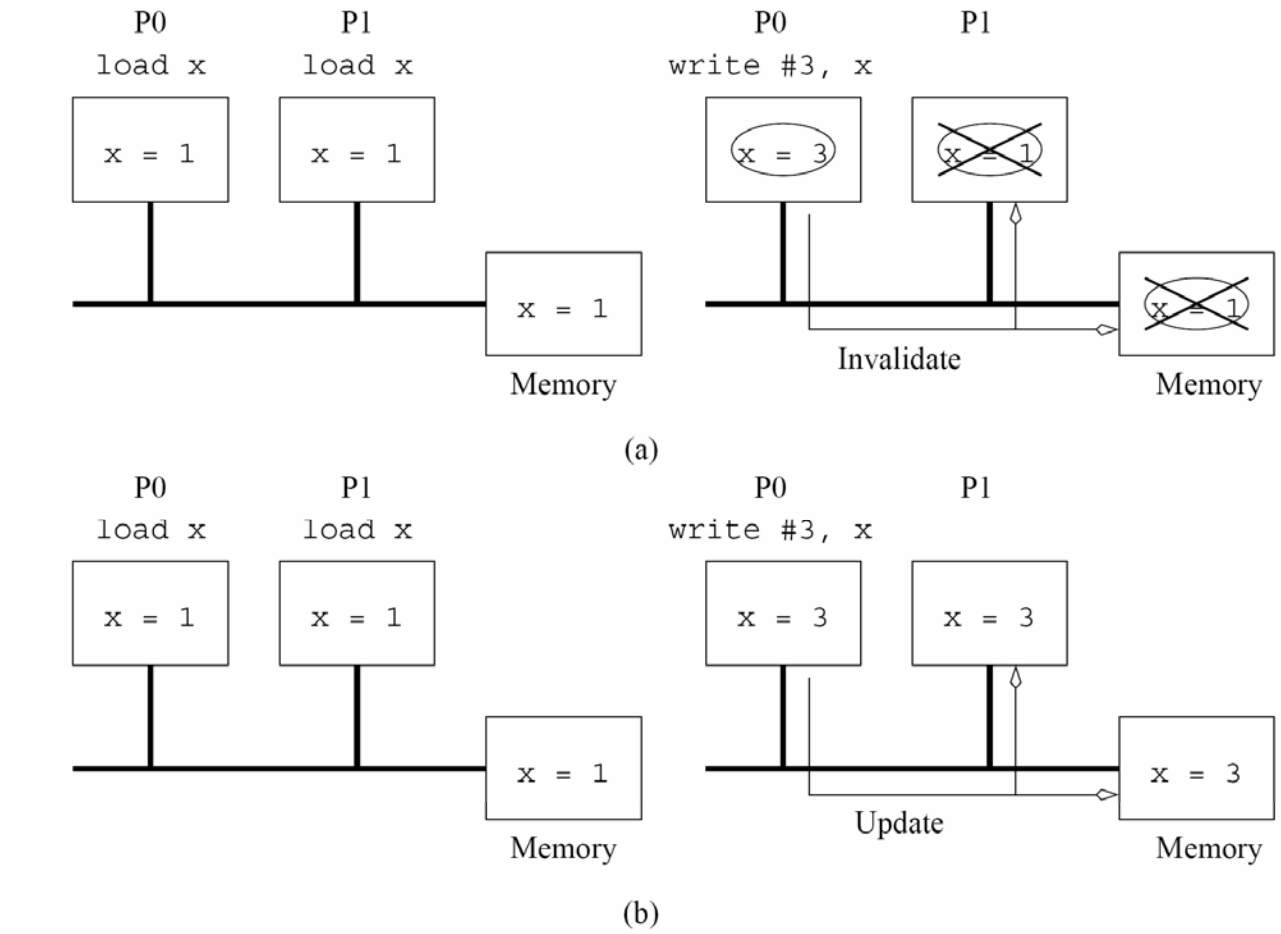


Figure 2.21 Cache coherence in multiprocessor systems: (a) Invalidate protocol; (b) Update protocol for shared variables.

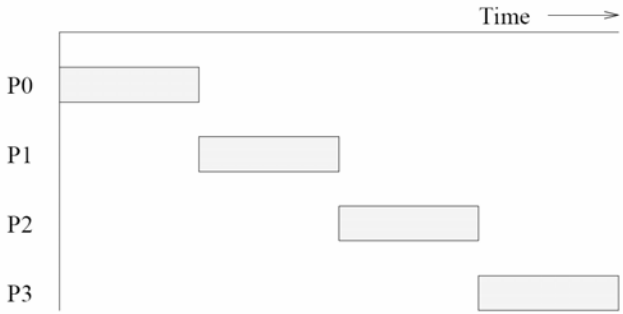
Invalidate/Update Protocols

- The preferred scheme depends on the characteristics of the underlying application
  - frequency of reads/writes to shared variables
- Classical trade-off between communication overhead (updates) and idling (stalling in invalidates)
- Additional problems with *false sharing*
- Existing schemes are based on the invalidate protocol
  - A number of approaches have been developed for maintaining the state/ownership of the shared data

Communication Costs in Parallel Systems

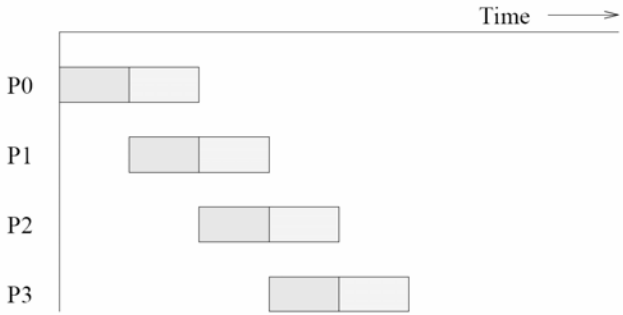
- Message-Passing Systems
  - The communication cost of a data-transfer operation depends on:
    - start-up time: t_s
      - add headers/trailer, error-correction, execute the routing algorithm, establish the connection between source & destination
    - per-hop time: t_h
      - time to travel between two directly connected nodes.
        - *node latency*
    - per-word transfer time: t_w
      - 1/channel-width

Store-and-Forward & Cut-Through Routing



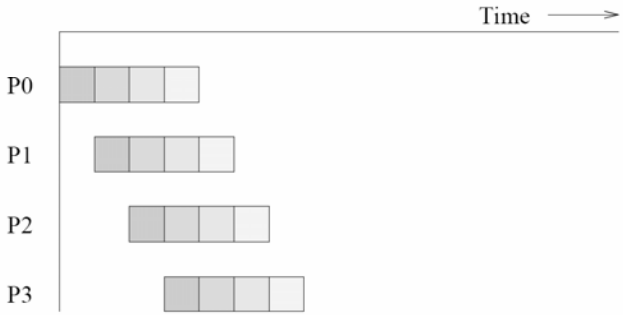
(a) A single message sent over a store-and-forward network

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



(b) The same message broken into two parts and sent over the network.

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



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

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

Cut-through Routing Deadlocks

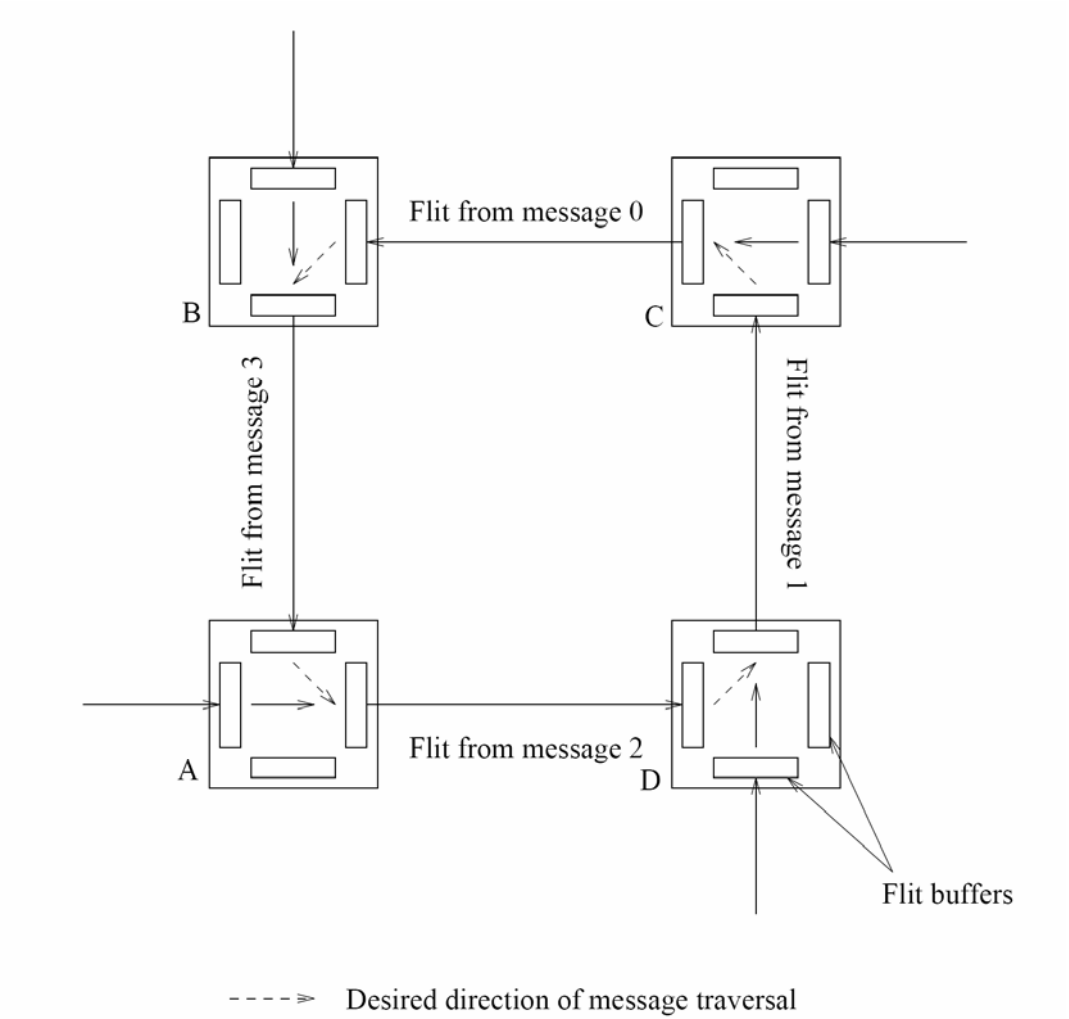


Figure 2.27 An example of deadlock in a cut-through routing network.

Messages 0, 1, 2, and 3 need to go to nodes A, B, C, and D, respectively

Communication Model Used for this Class

- We will assume that the cost of sending a message of size m is:

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

- In general true because t_s is much larger than t_h and for most of the algorithms that we will study mt_w is much larger than lt_h

Routing Mechanisms

- Routing:
  - The algorithm used to determine the path that a message will take to go from the source to destination
- Can be classified along different dimensions
  - minimal *vs* non-minimal
  - deterministic *vs* adaptive

Dimension Ordered Routing

- There is a predefined ordering of the dimensions
- Messages are routed along the dimensions in that order until they cannot move any further
  - X-Y routing for meshes
  - E-cube routine for hypercubes

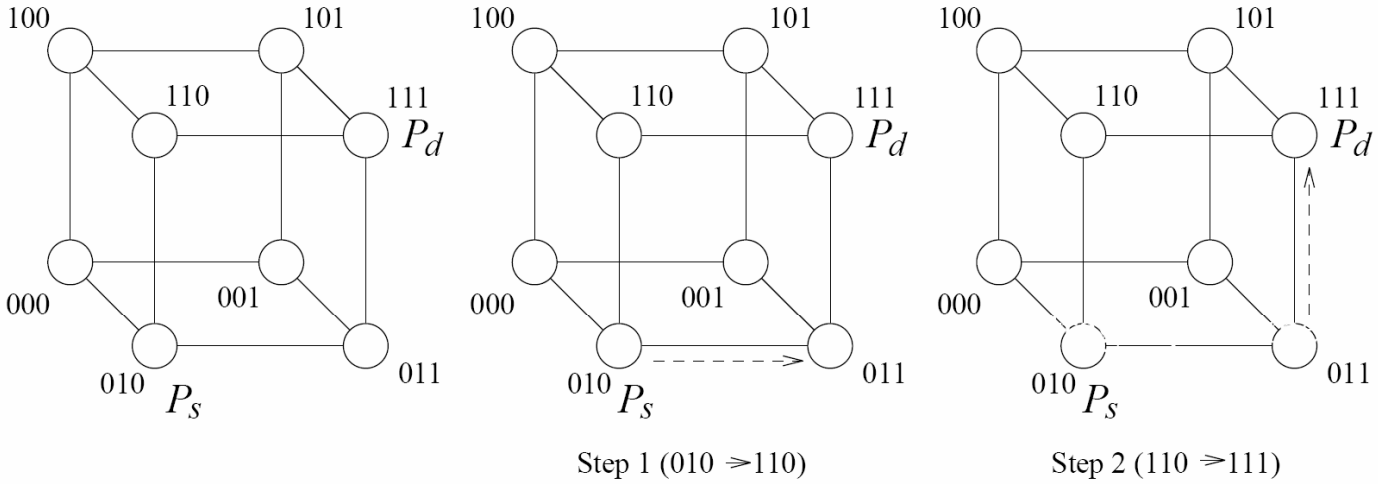


Figure 2.28 Routing a message from node P_s (010) to node P_d (111) in a three-dimensional hypercube using E-cube routing.

Topology Embeddings

- Mapping between networks
  - Useful in the early days of parallel computing when topology specific algorithms were being developed.
- Embedding quality metrics
  - dilation
    - maximum number of lines an edge is mapped to
  - congestion
    - maximum number of edges mapped on a single link

Mapping a Cartesian Topology onto a Hypercube

Cool things ☺

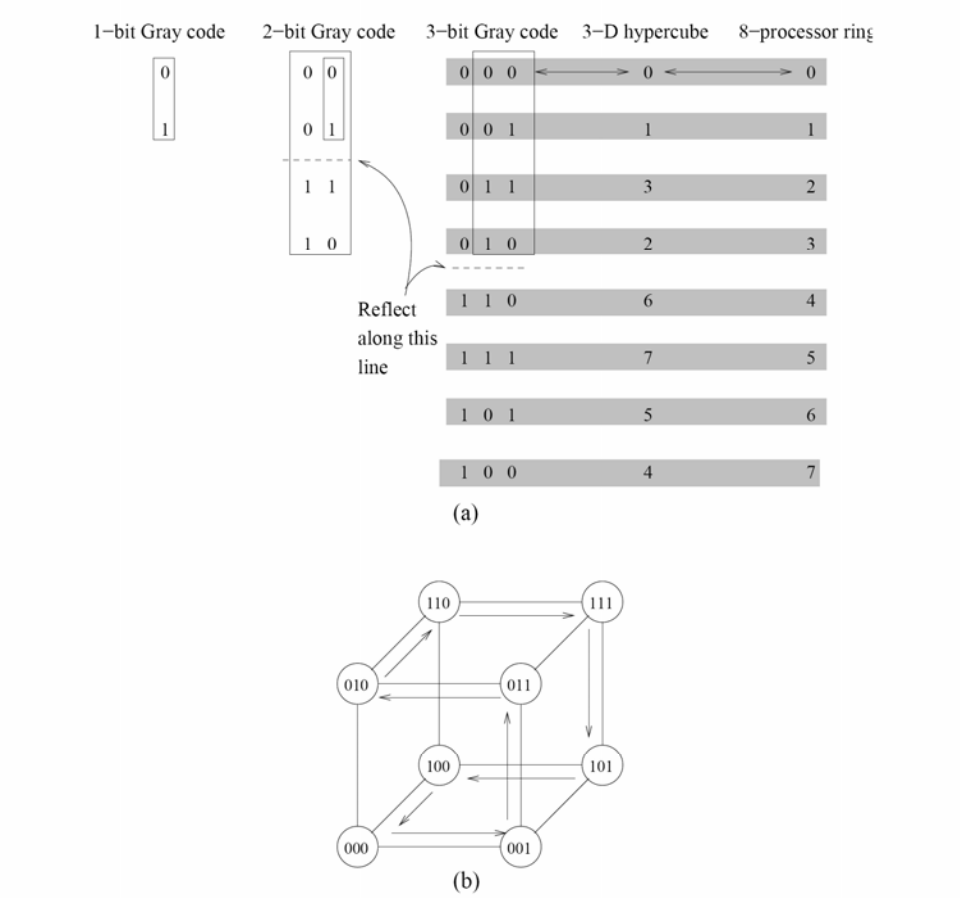


Figure 2.30 (a) A three-bit reflected Gray code ring; and (b) its embedding into a three-dimensional hypercube.

Mapping a Cartesian Topology onto a Hypercube

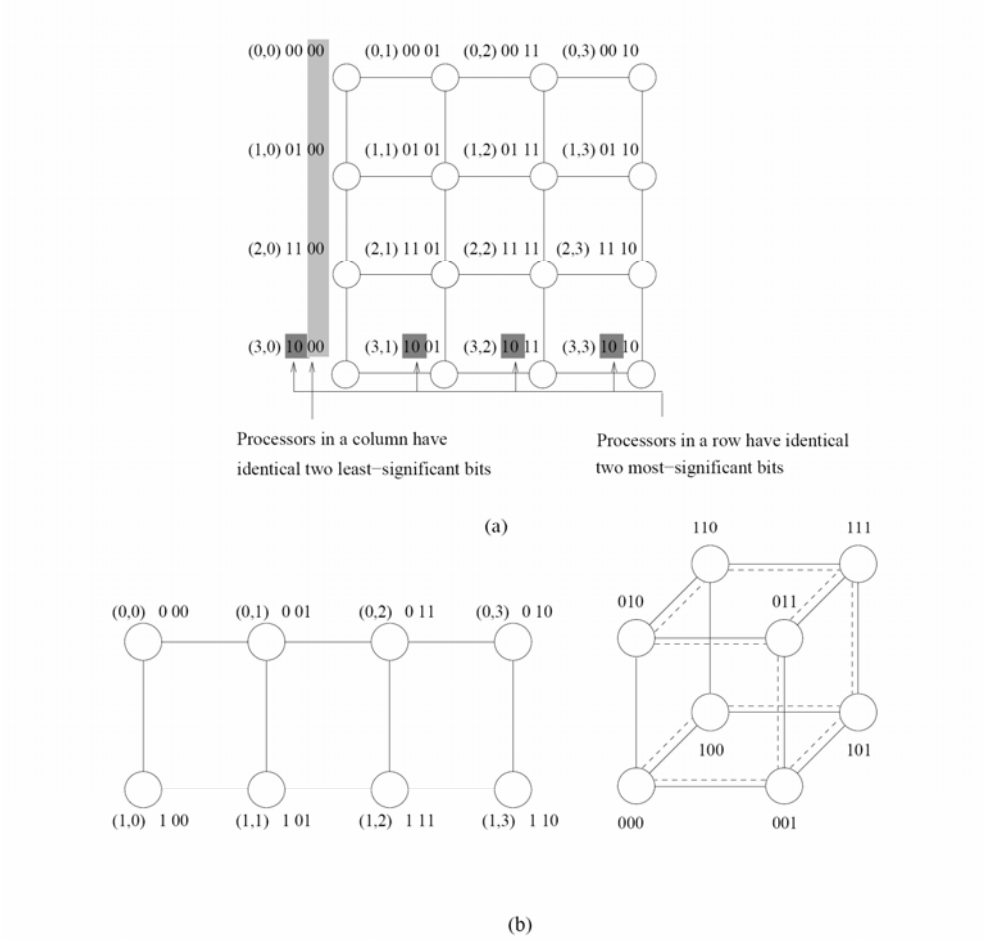


Figure 2.31 (a) A 4 × 4 mesh illustrating the mapping of mesh nodes to the nodes in a four-dimensional hypercube; and (b) a 2 × 4 mesh embedded into a three-dimensional hypercube.