COMP 422, Lecture 3: Physical Organization & Communication Costs in Parallel Machines

(Sections 2.4 & 2.5 of textbook)

Vivek Sarkar

Department of Computer Science
Rice University

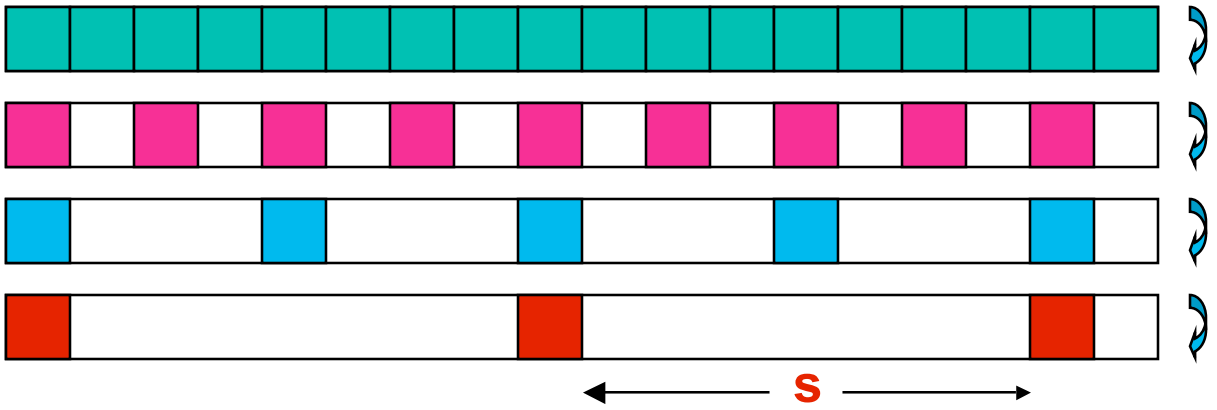
vsarkar@rice.edu

Recap of Lecture 2

- Taxonomy of Parallel Computing Platforms (Section 2.3)
  - —Shared memory vs. distributed memory
    - UMA vs. NUMA shared memory
  - —SIMD vs. MIMD
    - SIMD coprocessor vs. SIMD vector units
- Memory System Performance (Section 2.2)
  - —Latency
  - —Bandwidth
  - —Impact of caches
    - Membench microbenchmark
  - —Prefetching
  - —Multithreading

Experimental Study of Memory (Membench)

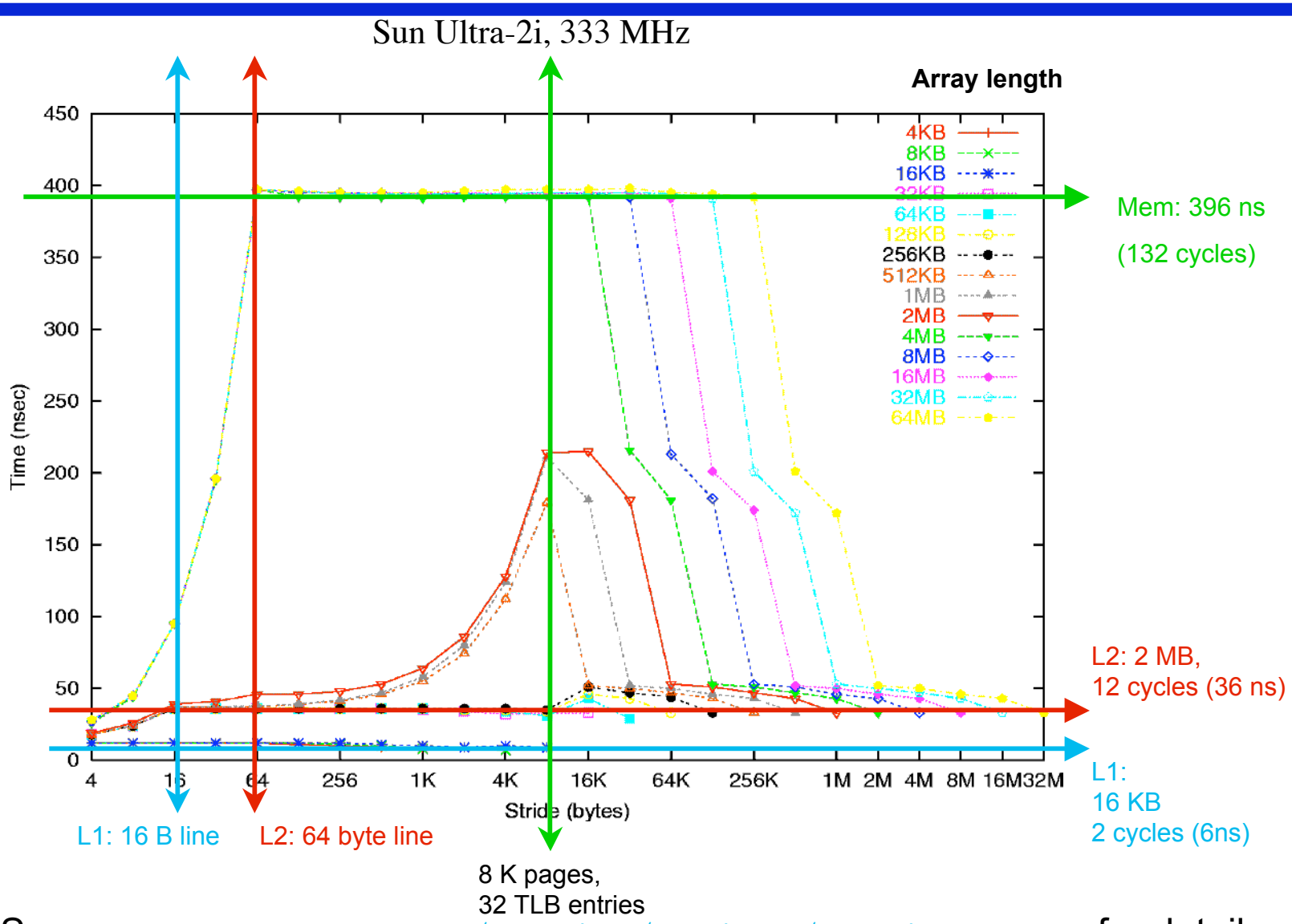
- Microbenchmark for memory system performance



  - for array A of length L from 4KB to 8MB by 2x
    for stride s from 4 Bytes (1 word) to L/2 by 2x
      time the following loop
      (repeat many times and average)
        for i from 0 to L by s
          load A[i] from memory (4 Bytes)

1 experiment

Memory Hierarchy on a Sun Ultra-2i



See www.cs.berkeley.edu/~yelick/arvindk/t3d-isca95.ps for details

Lecture 2 Review Question

- Consider two cases for a processor with a 1 GHz clock, 1ns access to cache, 100 ns access to DRAM, and a workload with 90% and 25% hit rates for cache sizes of 32KB and 1KB respectively:

  Case 1: 1 thread executes the workload with a 32KB cache

  Case 2: 32 threads execute the workload with 1KB cache/thread

— What memory bandwidth is required in the two cases?

- Case 1: (1 - 0.9) words/cycle = 0.1*4B/1ns = 400MB/s
- Case 2: (1 - 0.25) words/cycle = 0.75*4B/1ns = 3GB/s

The bandwidth ratio for Cases 1 and 2 is directly proportional to the miss rate ratio, 0.75 / 0.1

See Example 2.9 on page 23 of textbook

Acknowledgments for today's lecture

- Guang Gao & Joseph Manzano (U. Delaware) --- eleg652 slides from Fall 2007
  - http://www.capsl.udel.edu/~jmanzano/eleg652-07/slides/
- Slides accompanying course textbook
  - http://www-users.cs.umn.edu/~karypis/parbook/

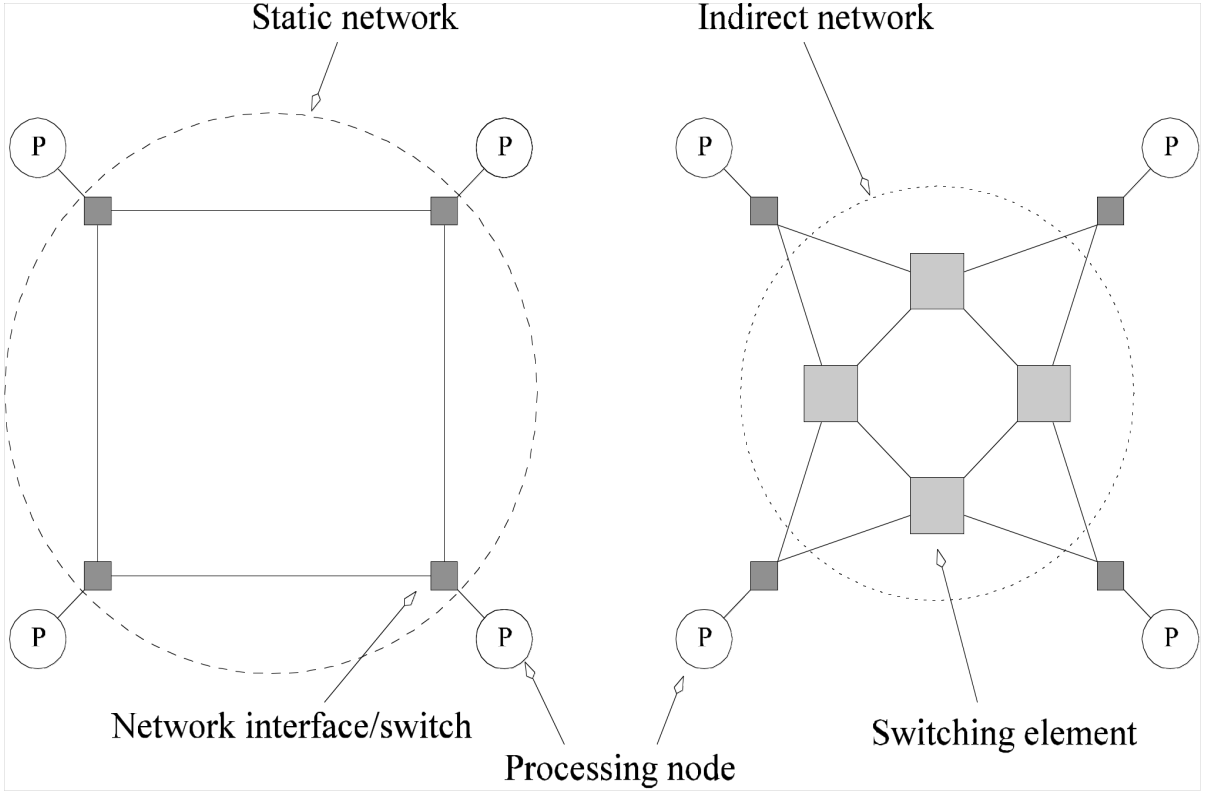
Outline of Today's Lecture

- **Interconnection Networks**
- **Cache Coherence**
- **Communication Costs**

Interconnection Networks: Terminology

- Interconnection networks carry data between processors and to memory.
- Topology: Connection Pattern between communication nodes (i.e. switches, NICs, routers, etc)
- Direct topology: static network with point-to-point communication links
  - —Analogy: point-to-point roads
- Indirect topology: dynamic networks built using *switches* and links
  - —Analogy: traffic signals and intersections
    - 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.
- Routing: The Discovery of a communication path between two computing resources
  - —Analogy: Car's Route
- Deadlock vs. Livelock
  - —Analogy: Gridlock vs. Getting stuck in a circle/rotary

Static and Dynamic Interconnection Networks



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

Metrics for Interconnection Networks

- ***Diameter:*** **The distance between the farthest two nodes in the network. Metric for worst-case latency.**
- ***Bisection Width:*** **The minimum number of wires you must cut to divide the network into two equal parts. Metric for worst-case bandwidth.**
- ***Arc Connectivity:*** **The minimum number of wires you must cut to partition the network into (not necessarily equal) parts. Metric for fault tolerance. Arc Connectivity is always <= Bisection Bandwidth.**
- ***Cost:*** **The number of links or switches (whichever is asymptotically higher) are important contributors to cost. However, a number of other factors, such as the ability to layout the network, the length of wires, etc., also factor in to the cost.**

Bisection Width: Example

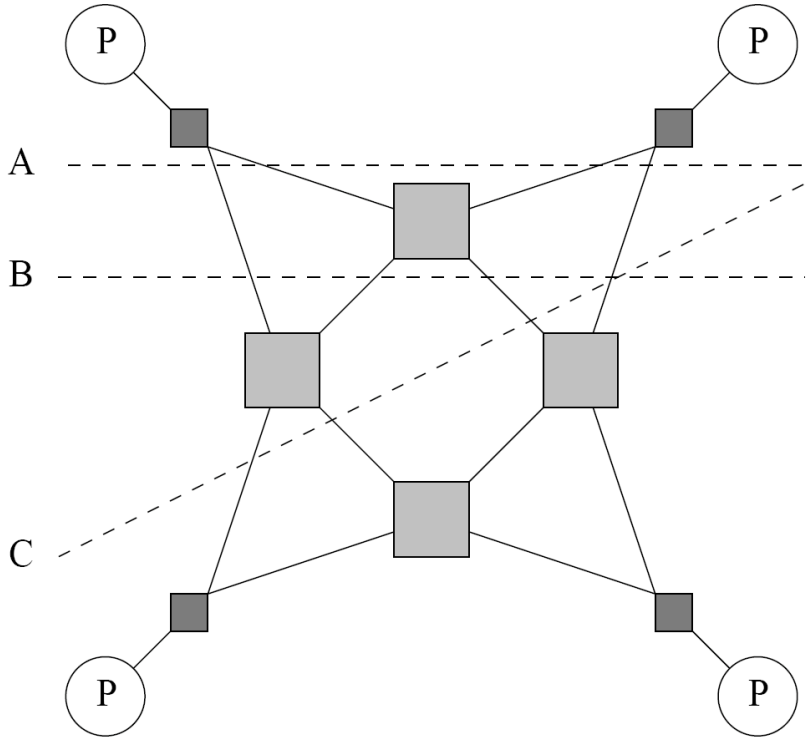


Figure 2.20 Bisection width of a dynamic network is computed by examining various equi-partitions of the processing nodes and selecting the minimum number of edges crossing the partition. In this case, each partition yields an edge cut of four. Therefore, the bisection width of this graph is four.

Evaluating Dynamic Interconnection Networks

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	$p - 1$

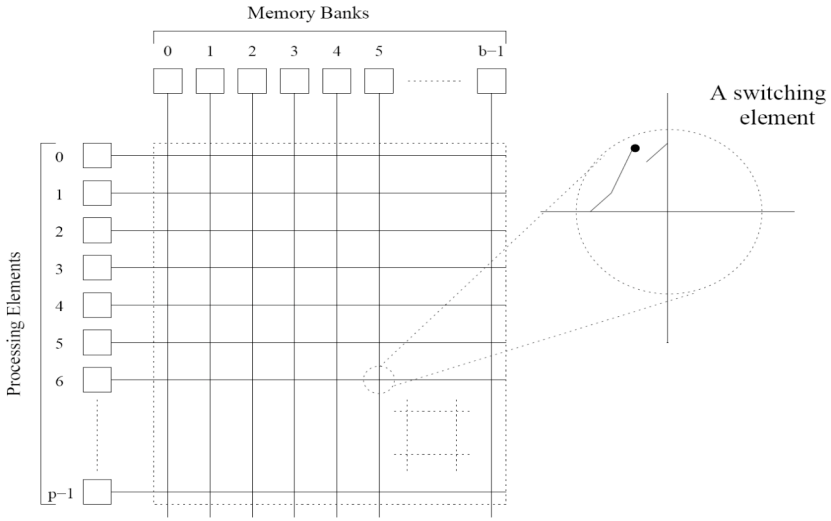
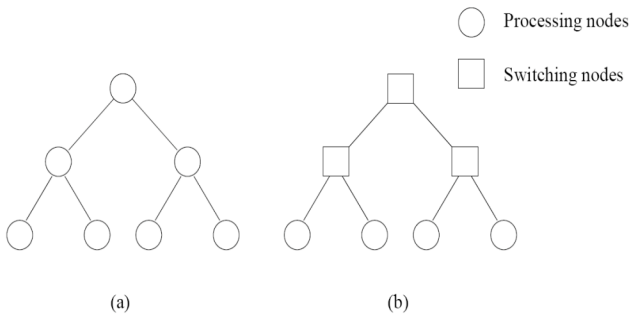


Figure 2.8 A completely non-blocking crossbar network connecting p processors to b memory banks.



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

Multistage Networks

- Crossbars have excellent bisection width but high cost
- Trees and buses have excellent cost but poor bisection width
- Multistage interconnects strike a compromise between these extremes

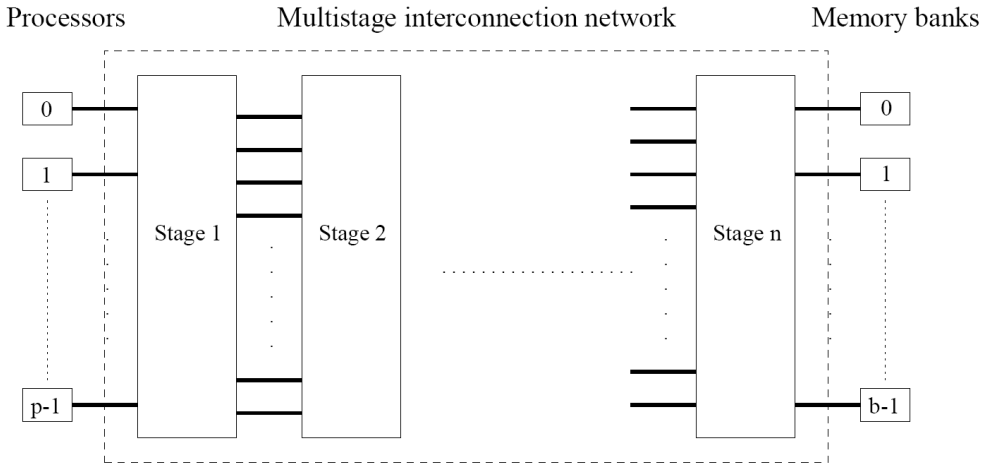
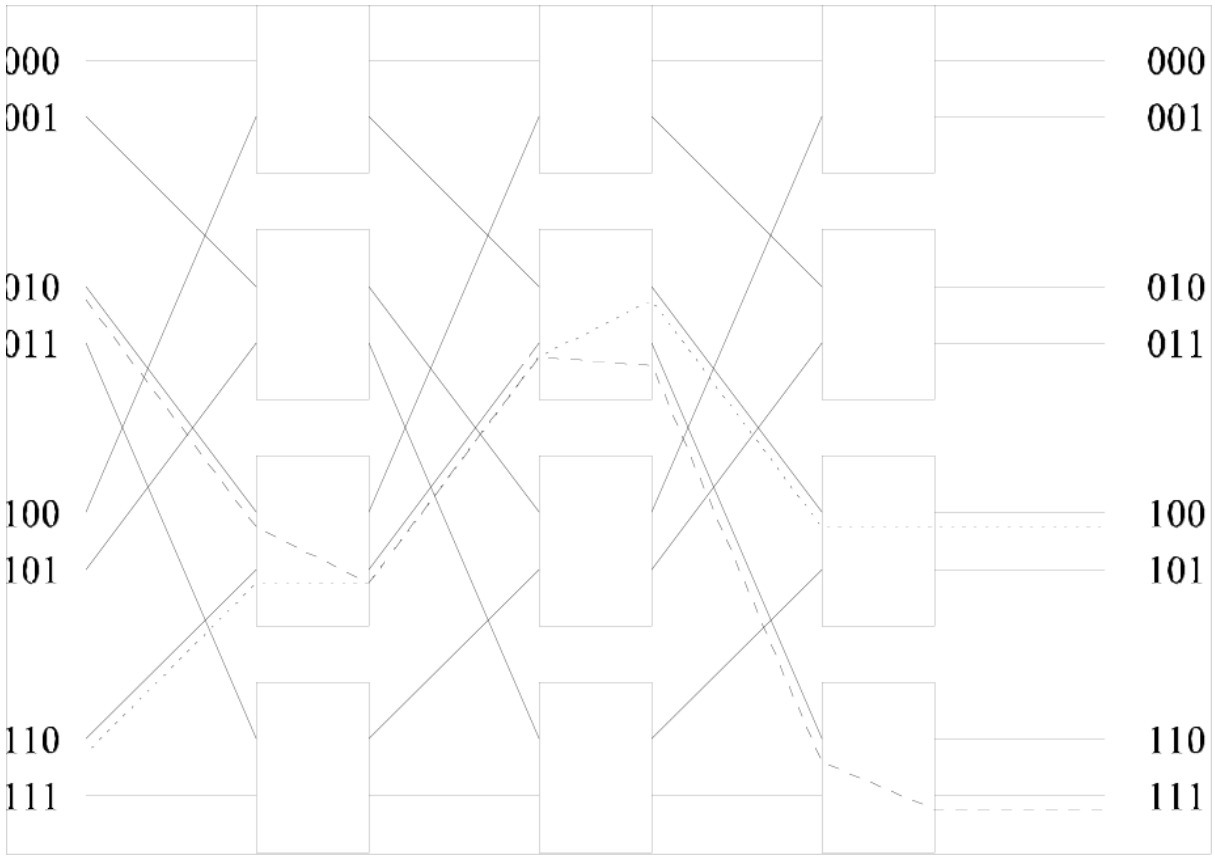


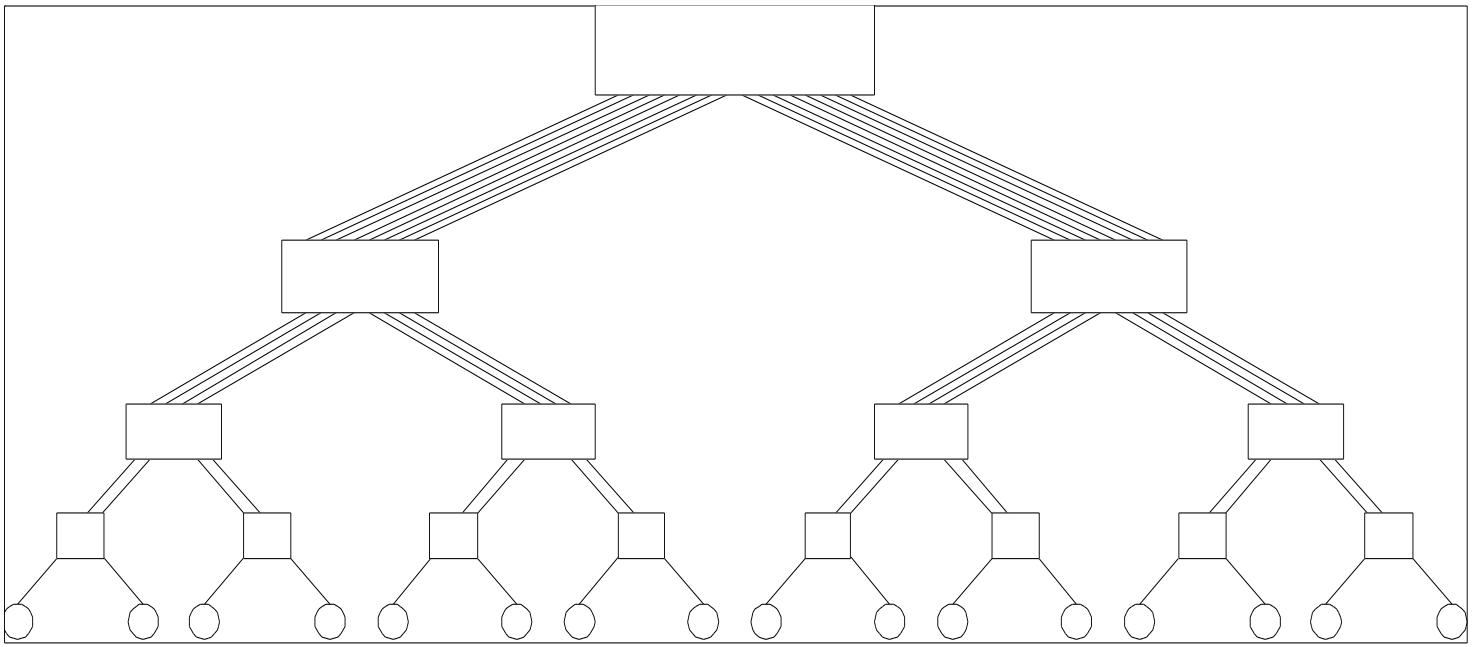
Figure 2.9 The schematic of a typical multistage interconnection network.

Multistage Omega Network – Routing



An example of routing in the omega network for source = 010 and dest = 111

Network Topologies: Fat Trees



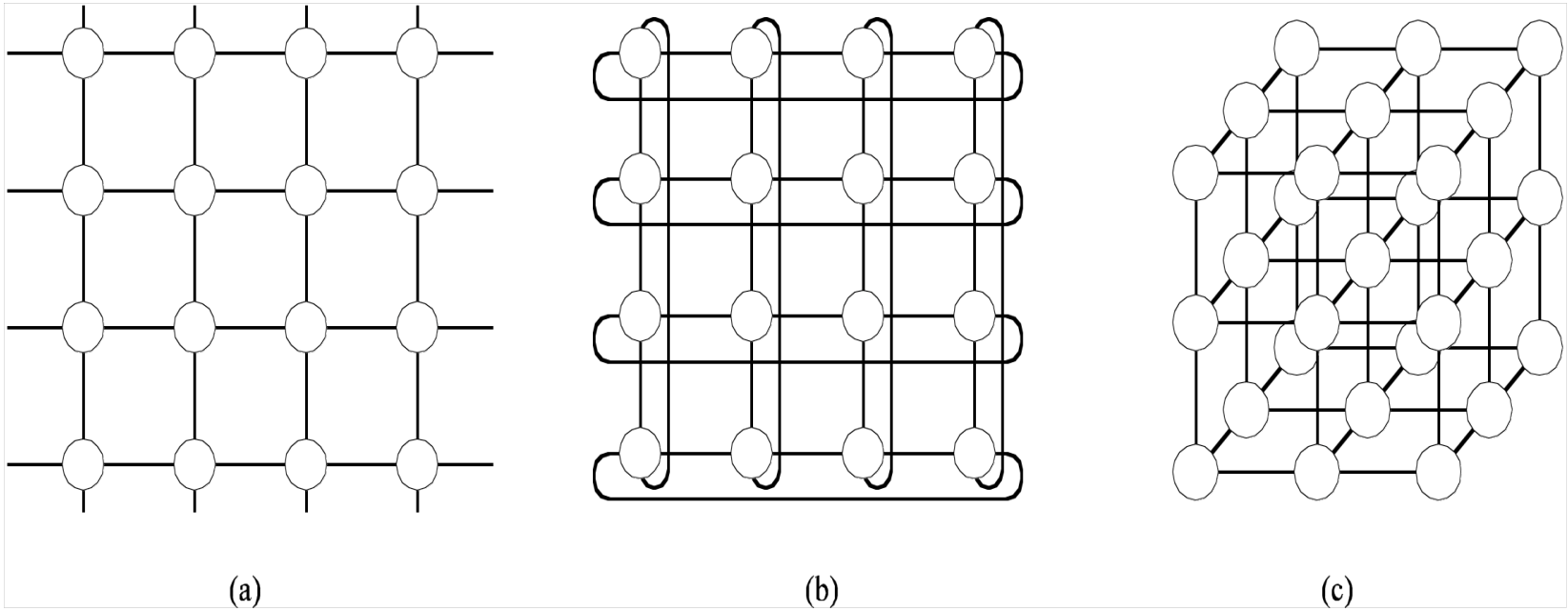
- Links higher up the tree potentially carry more traffic than those at the lower levels.
- For this reason, a variant called a fat-tree, fattens the links as we go up the tree.
- Trees can be laid out in 2D with no wire crossings. This is an attractive property of trees.

Evaluating Static Interconnection Networks

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

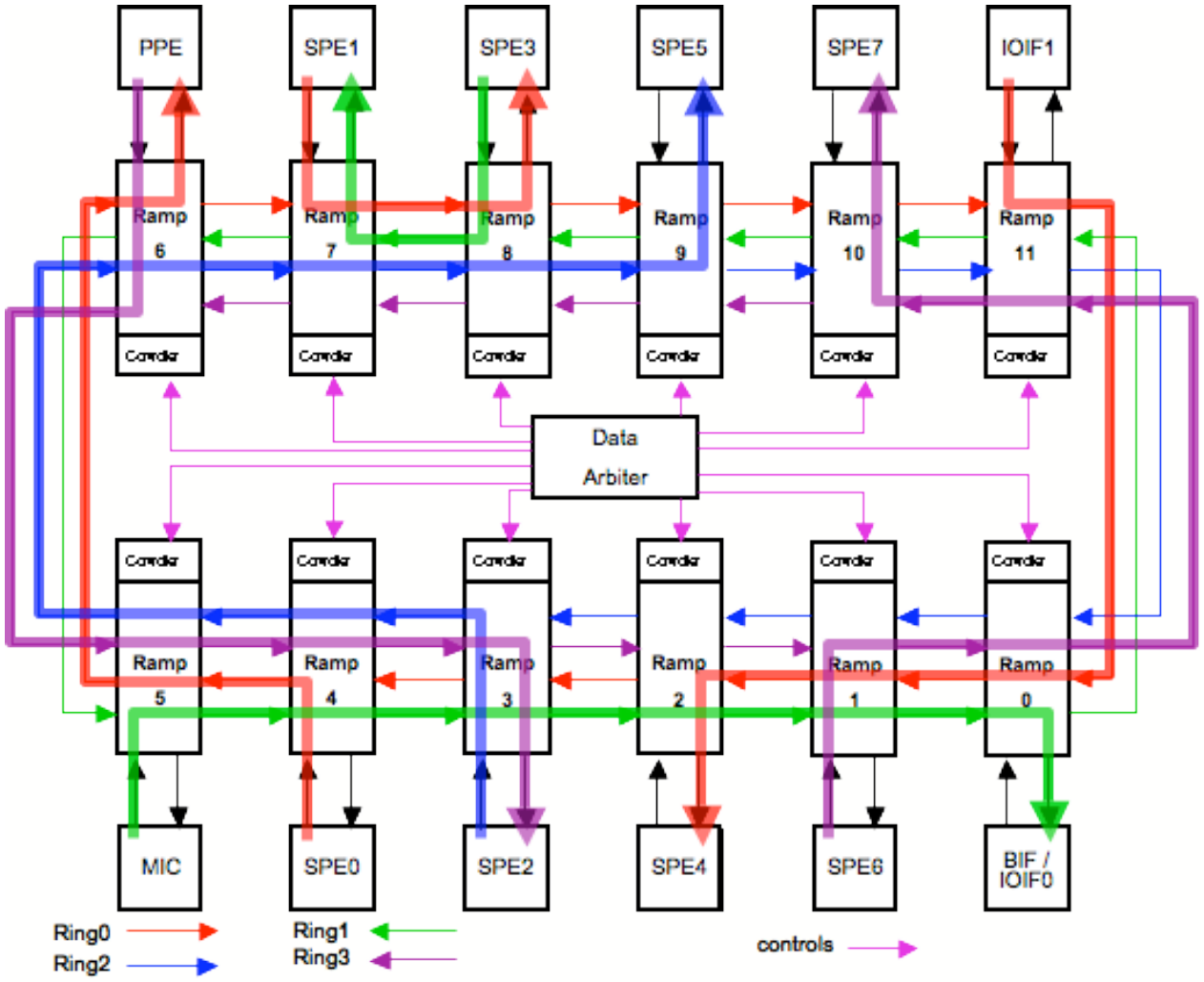
Network Topologies: Two- and Three Dimensional Meshes



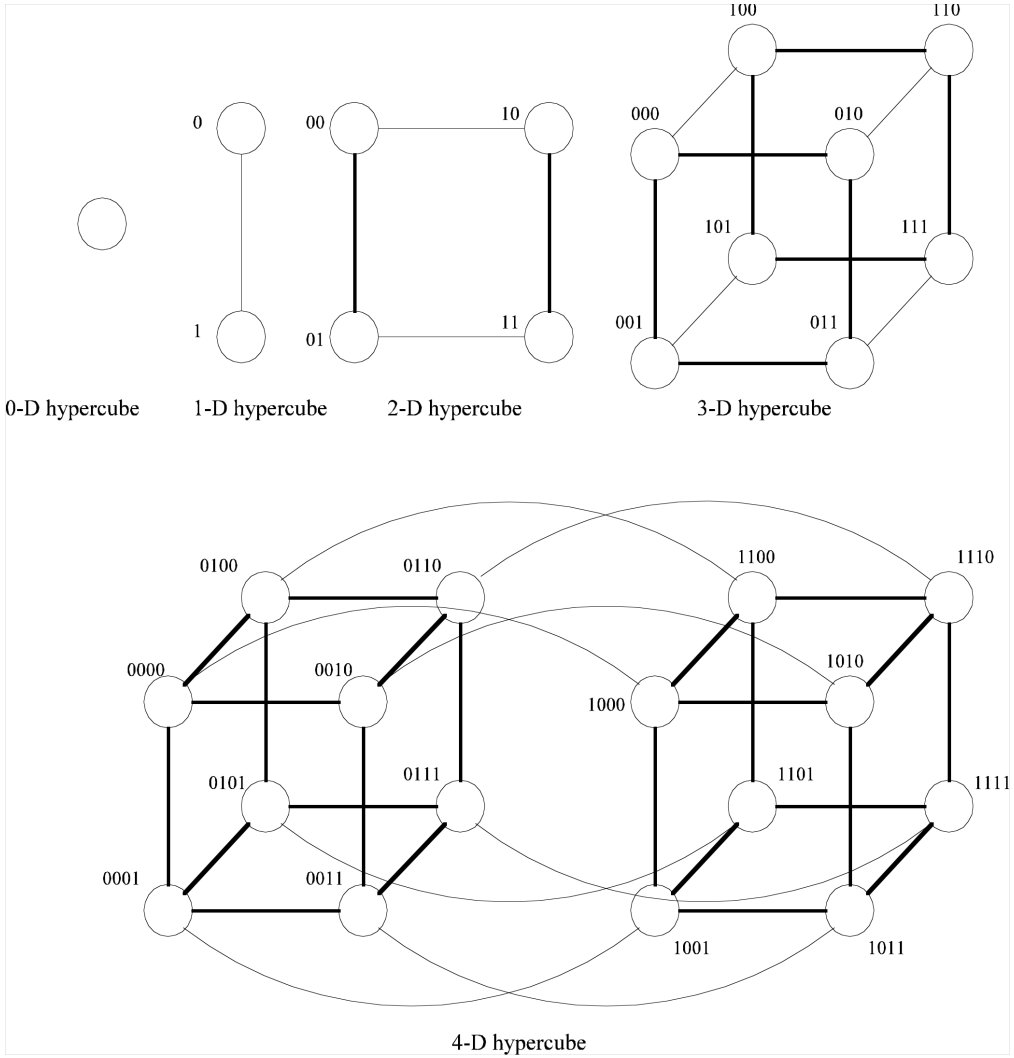
Two and three dimensional meshes:
(a) 2-D mesh with no wraparound;
(b) 2-D mesh with wraparound link (2-D torus);
(c) a 3-D mesh with no wraparound.

On-chip Interconnect Networks
e.g., Cell Element Interconnect Bus

- EIB data ring for internal communication
- Four unidirectional 16 byte data rings, supporting multiple transfers
  - —2 clockwise, 2 anti-clockwise; worst-case latency is half ring length
- 96B/cycle peak bandwidth
- Over 100 outstanding requests

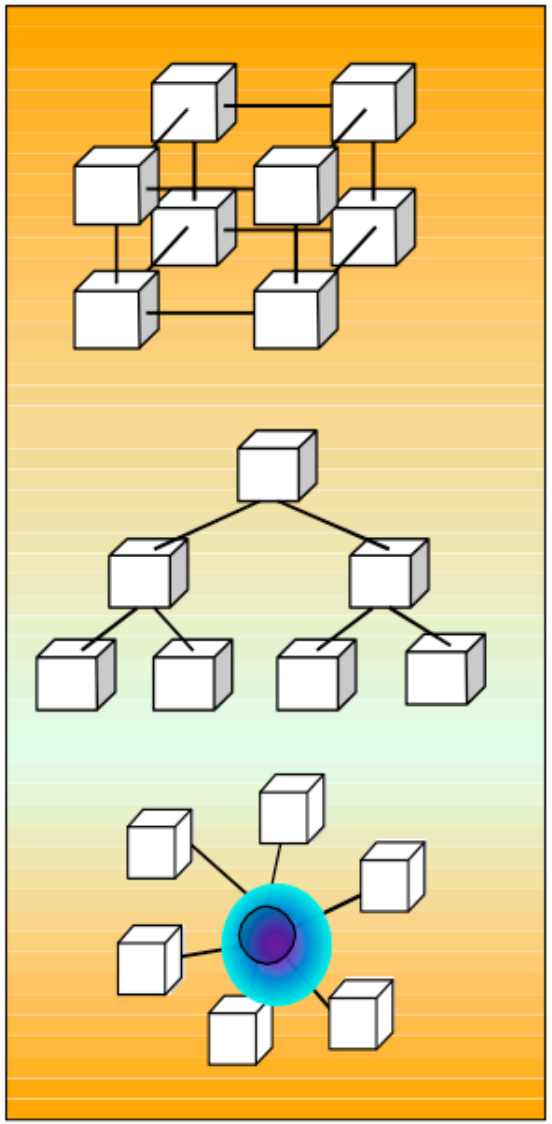


Hypercubes and their Construction



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

BlueGene/P Interconnection Networks



3 Dimensional Torus

- Interconnects all compute nodes
  - Communications backbone for computations
- Adaptive cut-through hardware routing
- 3.4 Gb/s on all 12 node links (5.1 GB/s per node)
- 0.5 μs latency between nearest neighbors, 5 μs to the farthest
  - MPI: 3 μs latency for one hop, 10 μs to the farthest
- 1.7/2.6 TB/s bisection bandwidth, 188TB/s total bandwidth (72k machine)

Collective Network

- Interconnects all compute and I/O nodes (1152)
- One-to-all broadcast functionality
- Reduction operations functionality
- 6.8 Gb/s of bandwidth per link
- Latency of one way tree traversal 2 μs, MPI 5 μs
- ~62TB/s total binary tree bandwidth (72k machine)

Low Latency Global Barrier and Interrupt

- Latency of one way to reach all 72K nodes 0.65 μs, MPI 1.6 μs

Other networks

- 10Gb Functional Ethernet
  - I/O nodes only
- 1Gb Private Control Ethernet
  - Provides JTAG access to hardware. Accessible only from Service Node system

Source: "Blue Gene: A Next Generation Supercomputer (BlueGene/P)", Alan Gara,

http://www.cisl.ucar.edu/dir/CAS2K7/Presentations/ThuAM/gara.pdf

BG/L vs. BG/P comparison

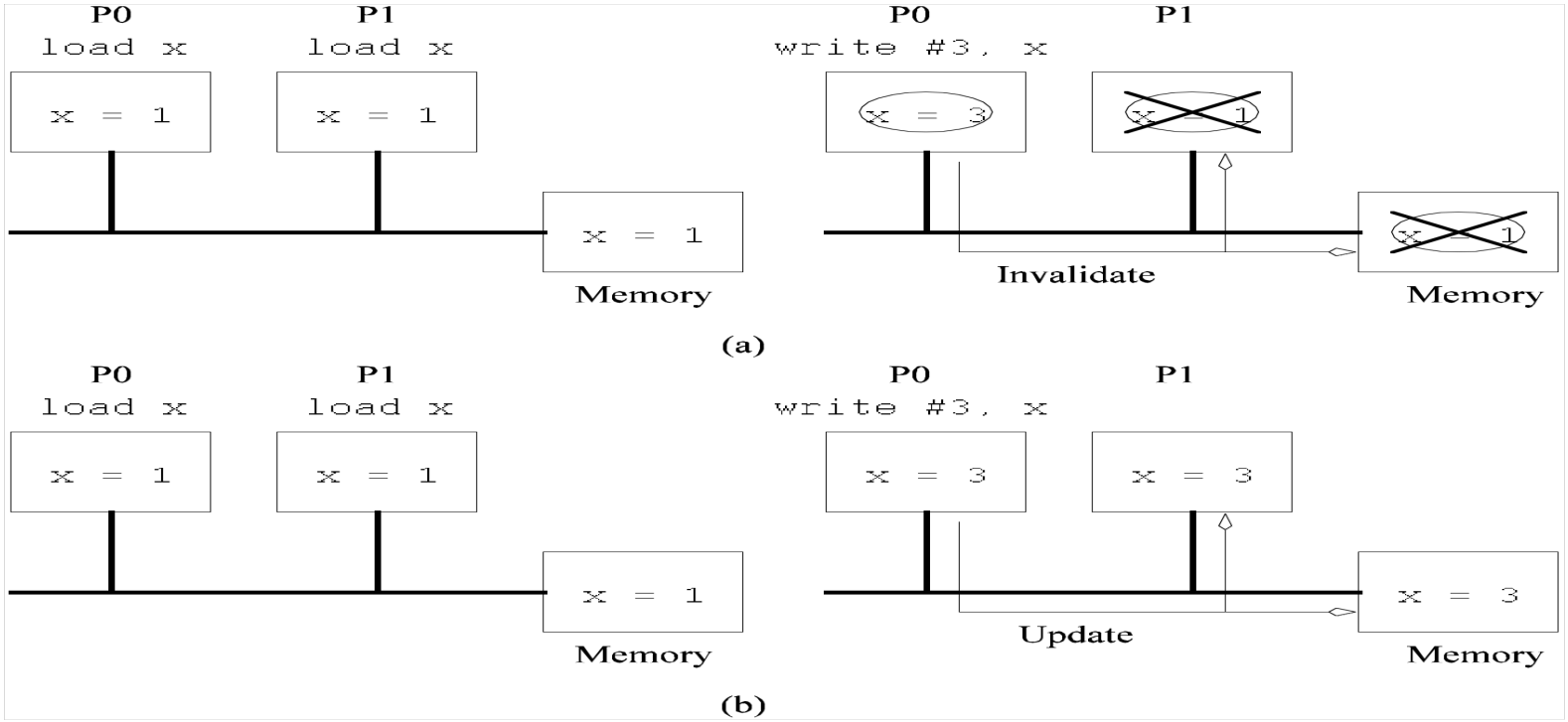
Property		BG/L	BG/P
Node Properties	Node Processors	2* 440 PowerPC	4* 450 PowerPC
	Processor Frequency	0.7GHz	0.85GHz (target)
	Coherency	Software managed	SMP
	L1 Cache (private)	32KB/processor	32KB/processor
	L2 Cache (private)	14 stream prefetching	14 stream prefetching
	L3 Cache size (shared)	4MB	8MB
	Main Store/node	512MB/1GB	2GB
	Main Store Bandwidth	5.6GB/s (16B wide)	13.6 GB/s (2*16B wide)
	Peak Performance	5.6GF/node	13.6 GF/node
Torus Network	Bandwidth	6*2*175MB/s=2.1GB/s	6*2*425MB/s=5.1GB/s
	Hardware Latency (Nearest Neighbor)	200ns (32B packet) 1.6us(256B packet)	160ns (32B packet) 500ns(256B packet)
	Hardware Latency (Worst Case)	6.4us (64 hops)	5us(64 hops)
Collective Network	Bandwidth	2*350MB/s=700MB/s	2*0.85GB/s=1.7GB/s
	Hardware Latency (round trip worst case)	5.0us	4us
System Properties	Peak Performance (72k nodes)	410TF	1PF
	Total Power	1.7MW	2.7 MW

Outline of Today's Lecture

- Interconnection Networks
- Cache Coherence
- Communication Costs

Cache Coherence in Multiprocessor Systems

When the value of a cache line changes, all its copies must either be invalidated or updated.



Cache coherence in multiprocessor systems:
(a) Invalidate protocol; (b) Update protocol

Cache Coherence: Update and Invalidate Protocols

- If two processors make interleaved test and updates to a shared variable, then an *update* protocol is better
- If one processor performs multiple writes on a shared variable before it is accessed by another processor, then an *invalidate* protocol is better
- Most current machines use invalidate protocols.
- Both protocols suffer from *false sharing* overheads (two words that are not actually shared, but happen to lie on the same cache line).

Maintaining Coherence Using an Invalidate Protocol

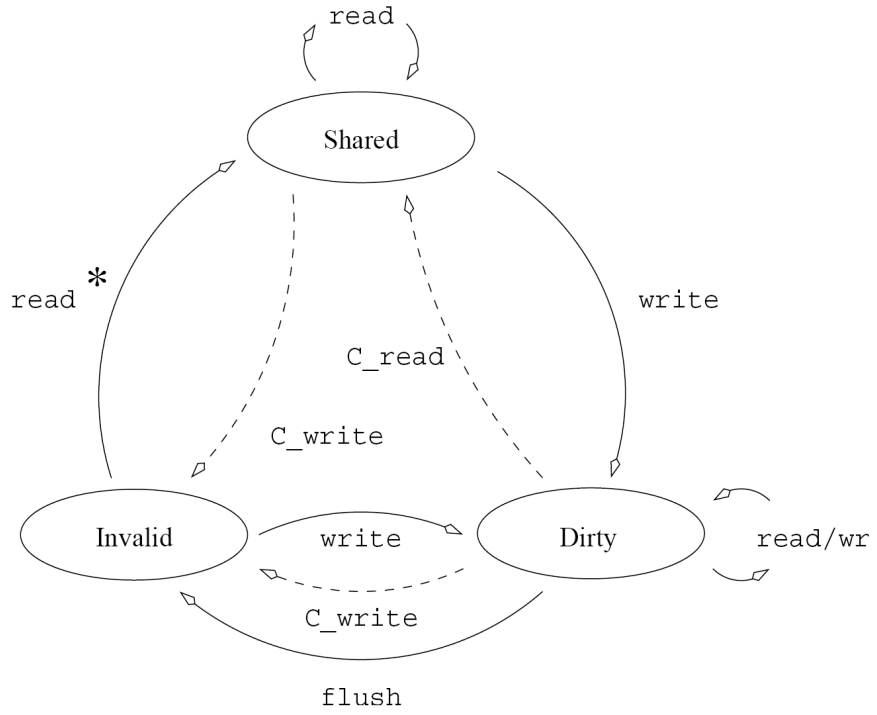


Figure 2.22 State diagram of a simple three-state coherence protocol.

Time

Instruction at Processor 0	Instruction at Processor 1	Variables and their states at Processor 0	Variables and their states at Processor 1	Variables and their states in Global mem.
				x = 5, D y = 12, D
read x		x = 5, S		x = 5, S y = 12, S
	read y		y = 12, S	
x = x + 1		x = 6, D		x = 5, I y = 12, I
	y = y + 1		y = 13, D	
read y		y = 13, S	y = 13, S	y = 13, S
	read x	x = 6, S	x = 6, S	x = 6, S
x = x + y		x = 19, D	x = 6, I	x = 6, I
	y = x + y	y = 13, I	y = 19, D	y = 13, I
x = x + 1		x = 20, D		x = 6, I
	y = y + 1		y = 20, D	y = 13, I

Figure 2.23 Example of parallel program execution with the simple three-state coherence prot discussed in Section 2.4.6.

* If another processor has a dirty copy, it will service the read instead of the memory

Snoopy Cache Systems

How are invalidates sent to the right processors?

In snoopy caches, there is a broadcast media (a bus) that listens to all invalidates and read requests and performs appropriate coherence operations locally.

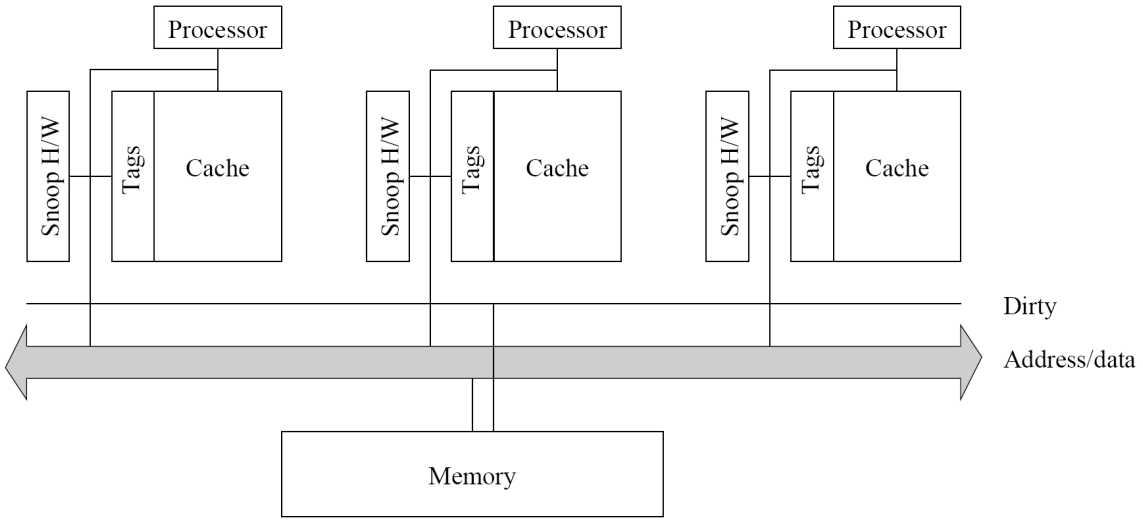
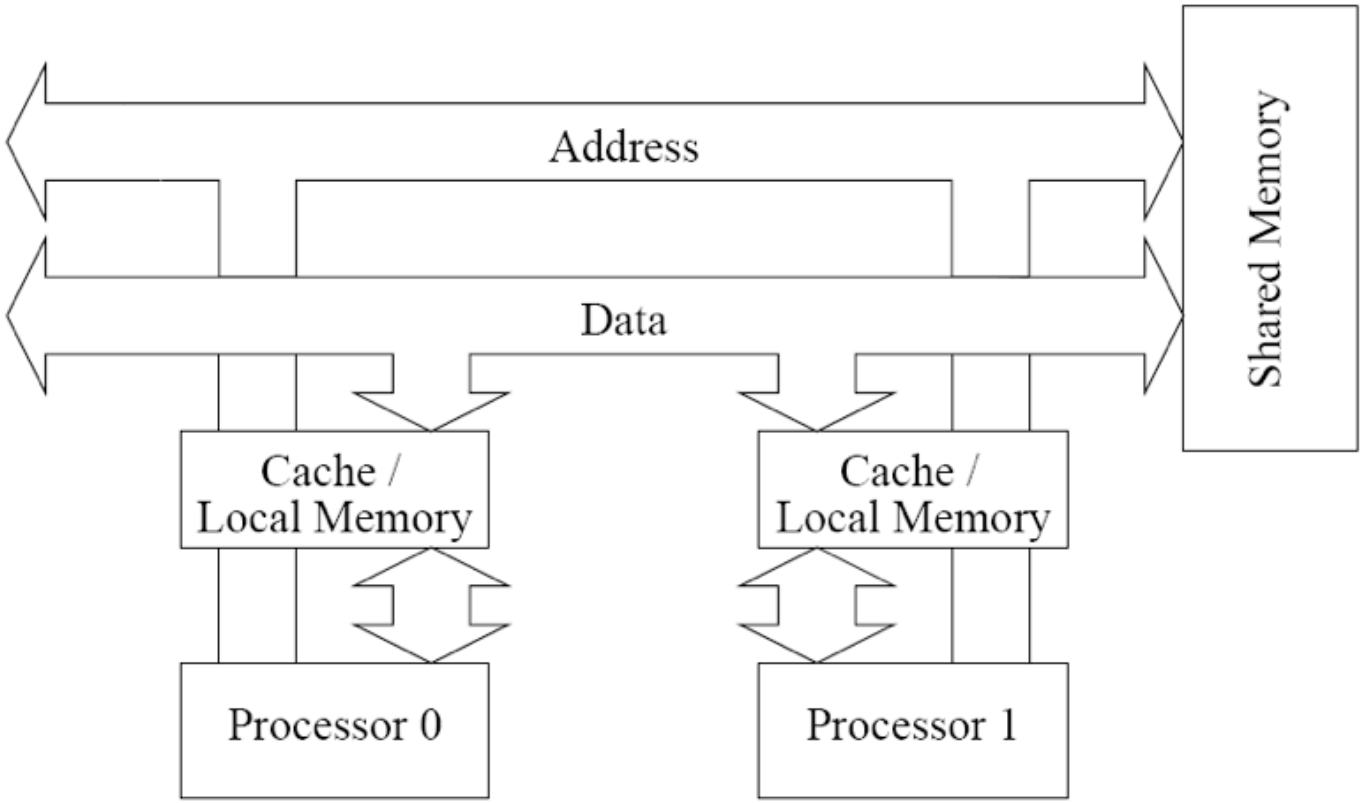


Figure 2.24 A simple snoopy bus based cache coherence system.

Network Topologies: Buses



- Bus-based interconnects 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: Buses

- Some of the simplest and earliest parallel machines used buses.
  - Multicore processors are following suit!
- 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.
- Also, in snoopy caches, each coherence operation is sent to all processors.
  - Why not send coherence requests to only those processors that need to be notified?

Directory Based Systems

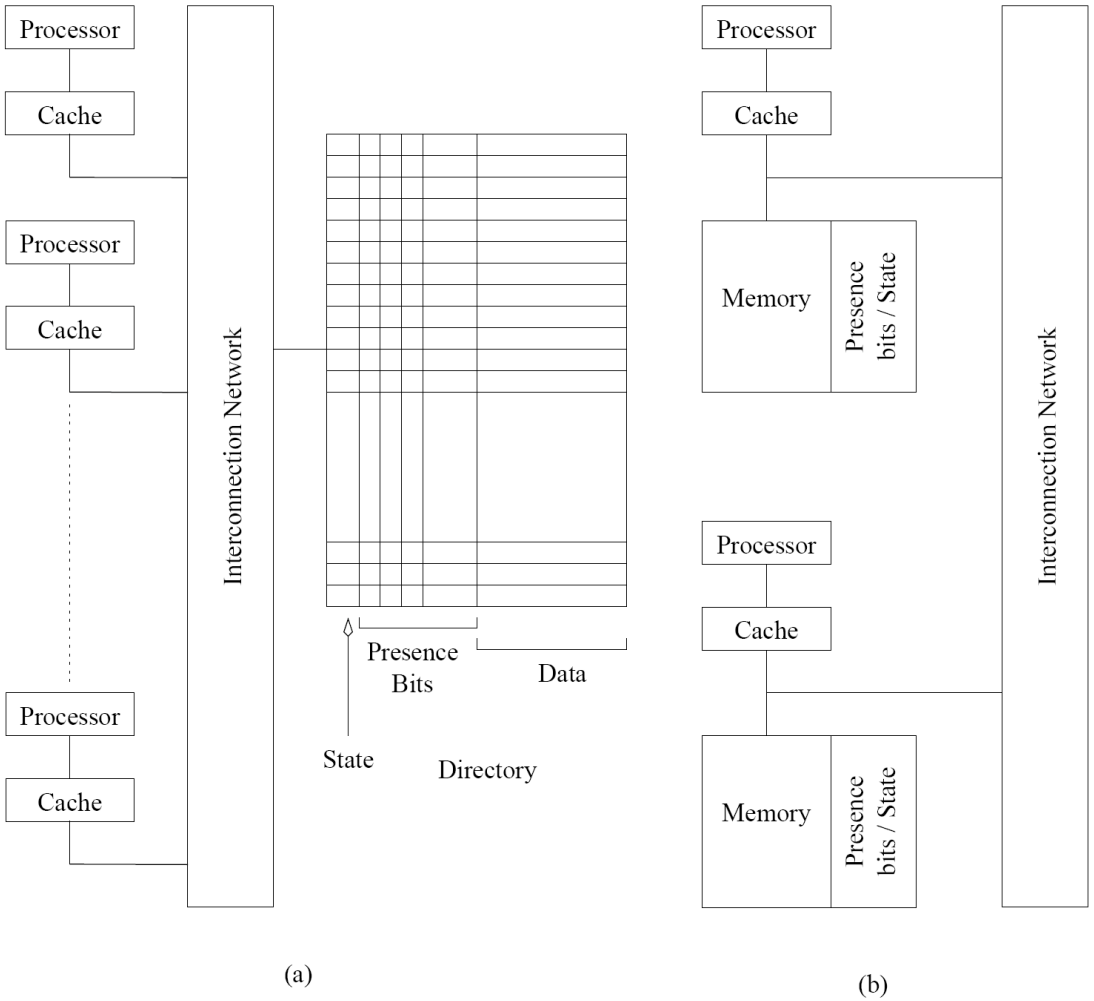


Figure 2.25 Architecture of typical directory based systems: (a) a centralized directory; and (b) a distributed directory.

Outline of Today's Lecture

- **Interconnection Networks**
- **Cache Coherence**
- **Communication Costs**

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

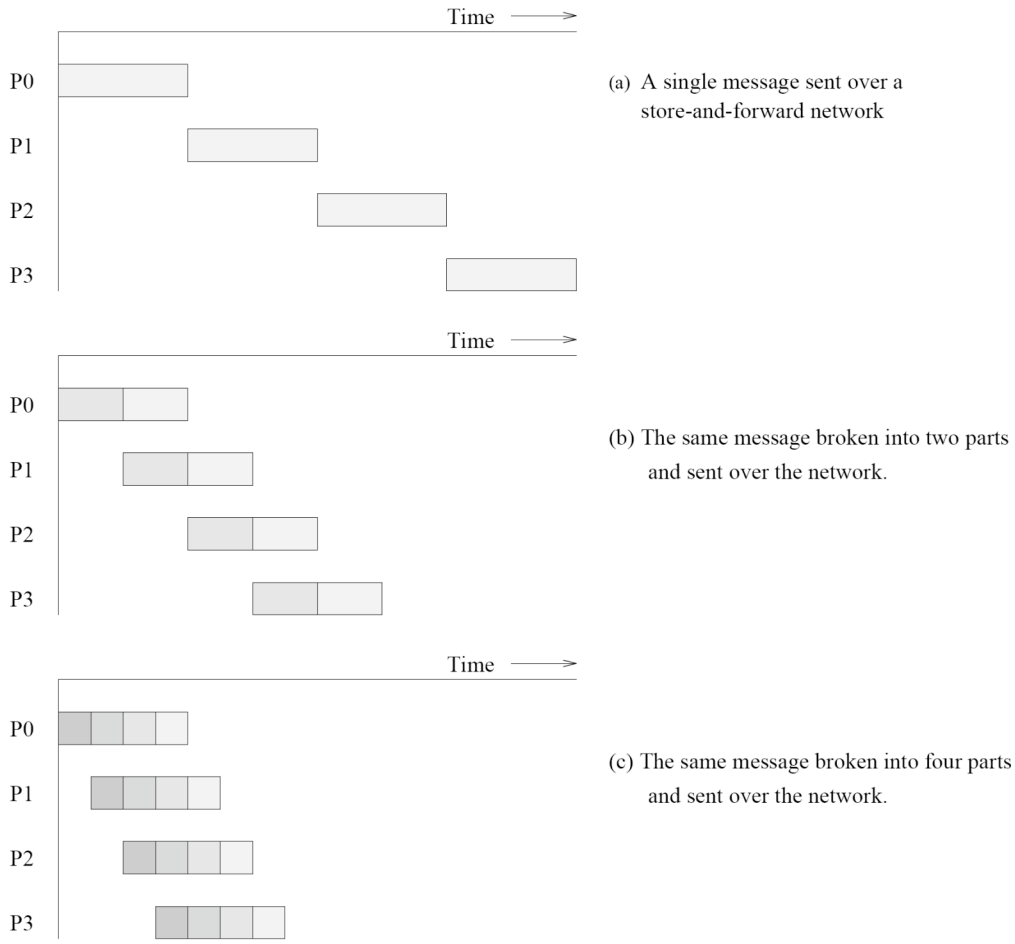


Figure 2.26 Passing a message from node P_0 to P_3 (a) through a store-and-forward communication network; (b) and (c) extending the concept to cut-through routing. The shaded regions represent the time that the message is in transit. The startup time associated with this message transfer is assumed to be zero.

Simplified Communication Cost Model

- 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 in the RHS 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

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

Ordering of Course Topics

- **Introduction (Chapter 1)**
- **Parallel Programming Platforms (Chapter 2)**
  - ***New material: homogeneous & heterogeneous multicore platforms***
- **Principles of Parallel Algorithm Design (Chapter 3)**
- **Programming Shared Address Space Platforms (Chapter 7)**
  - ***New material: new programming models (beyond threads and OpenMP) --- Java Concurrency Utilities, Intel Thread Building Blocks, .Net Parallel Extensions (Task Parallel Library & PLINQ), Cilk, X10***
- **Analytical Modeling of Parallel Programs (Chapter 5)**
  - ***New material: theoretical foundations of task scheduling***
- **Dense Matrix Operations (Chapter 8)**
- **Graph Algorithms (Chapter 10)**
- **Programming Using the Message-Passing Paradigm (Chapter 6)**
  - ***New material: Partitioned Global Address Space (PGAS) languages --- Unified Parallel C (UPC), Co-array Fortran (CAF)***
- ***New material: Programming Heterogeneous Processors and Accelerators***
- ***New material: Problem Solving on Large Scale Clusters using MapReduce***

Topic for next lecture

Topic to be started next week

Summary of Today's Lecture

- Interconnection Networks
- Cache Coherence
- Communication Costs

Reading List for Next Lecture (Jan 17th)

- Sections 3.1, 3.2

We will move on to Chapter 7 (Shared Memory Parallel Programming) next week

- REMINDER: send email to TA w/ Ada account info (HW 1)