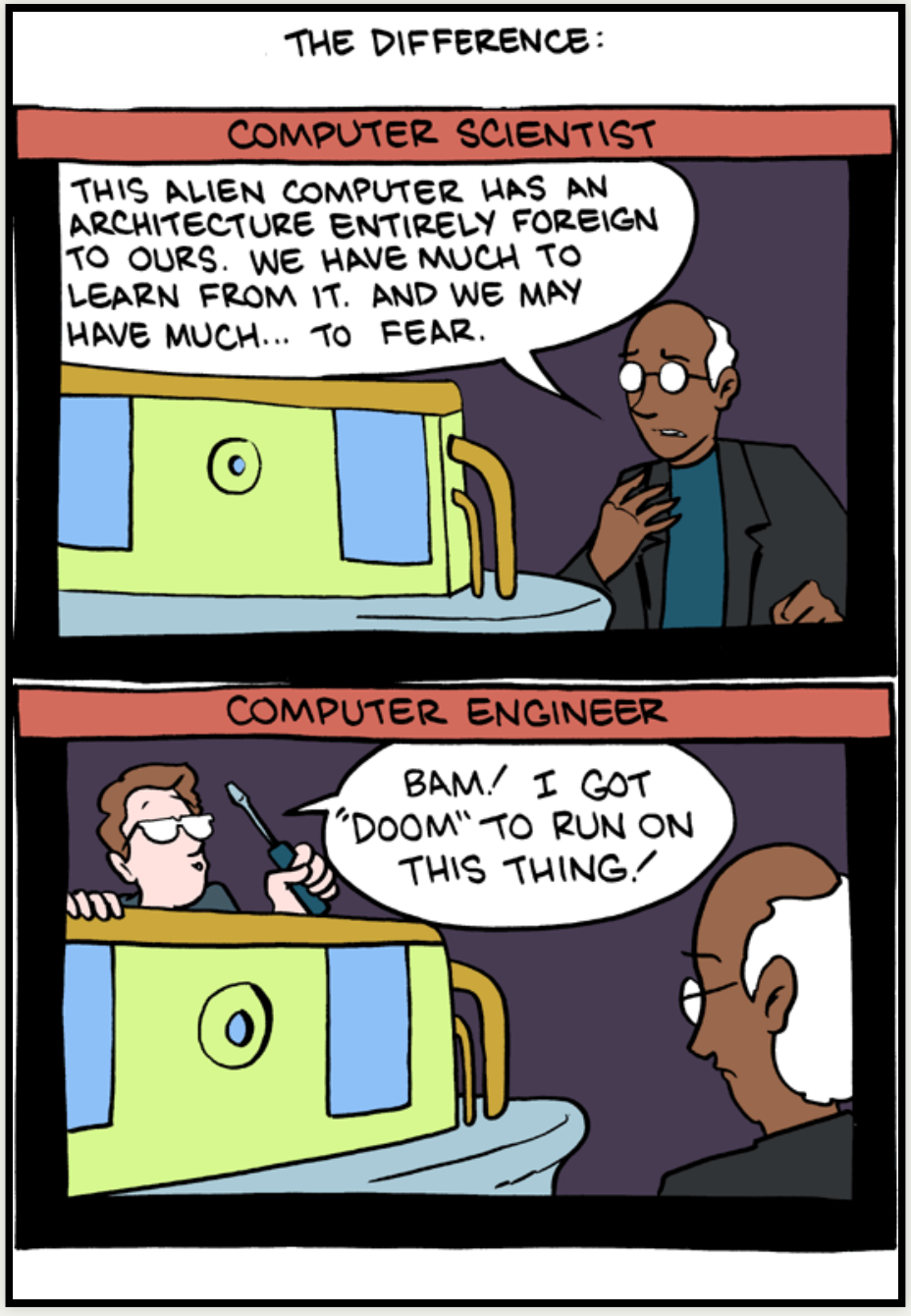
CS5005

Parallel Programming

Introduction

Department of Computer Science and Engineering, Indian Institute of Technology Palakkad


THE DIFFERENCE:
COMPUTER SCIENTIST
THIS ALIEN COMPUTER HAS AN ARCHITECTURE ENTIRELY FOREIGN TO OURS. WE HAVE MUCH TO LEARN FROM IT. AND WE MAY HAVE MUCH... TO FEAR.
COMPUTER ENGINEER
BAM! I GOT "DOOM" TO RUN ON THIS THING!

Books

- Ananth Grama, Georg Karypis, Vipin Kumar and Anshul Gupta: **Introduction to Parallel Computing**, Pearson Education India, 2 edition (2004), ISBN-13: 978-8131708071
- Peter S. Pacheco: **An Introduction to Parallel Programming**, Elsevier India, First edition (2011), ISBN-13: 978-9380931753
- Maurice Herlihy and Nir Shavit: **The Art of Multiprocessor Programming**, Elsevier India, First edition (2012), ISBN-13: 978-9382291510
- Michael Quinn, **Parallel Programming in C with Mpi and Openmp**, McGraw Hill Education, 1 edition (1 July 2017), ISBN-13: 978-0070582019

Grading scheme

- 15% Quiz, 40% End-sem, 45% Continuous evaluation (assignments/projects/homeworks/class-tests)

OR

- 10% Quiz-1, 10% Quiz-2, 35% End-sem, 45% Continuous evaluation

Elements of a Parallel Computer (Hardware)

- Multiple Processors (multiple cores)
- Multiple Memories
- Interconnect Network

Elements of a Parallel Computer (Software)

- Operating System that supports parallelism
- Programming constructs, and its associated tool-chain for implementation
- Parallel Algorithms

Example

Serial program to compute sum of 'n' numbers

```
sum = 0;
for (i = 0; i < n; i++) {
        x = compute_next_value(...);
        sum += x;
}
```

Sample Parallelization

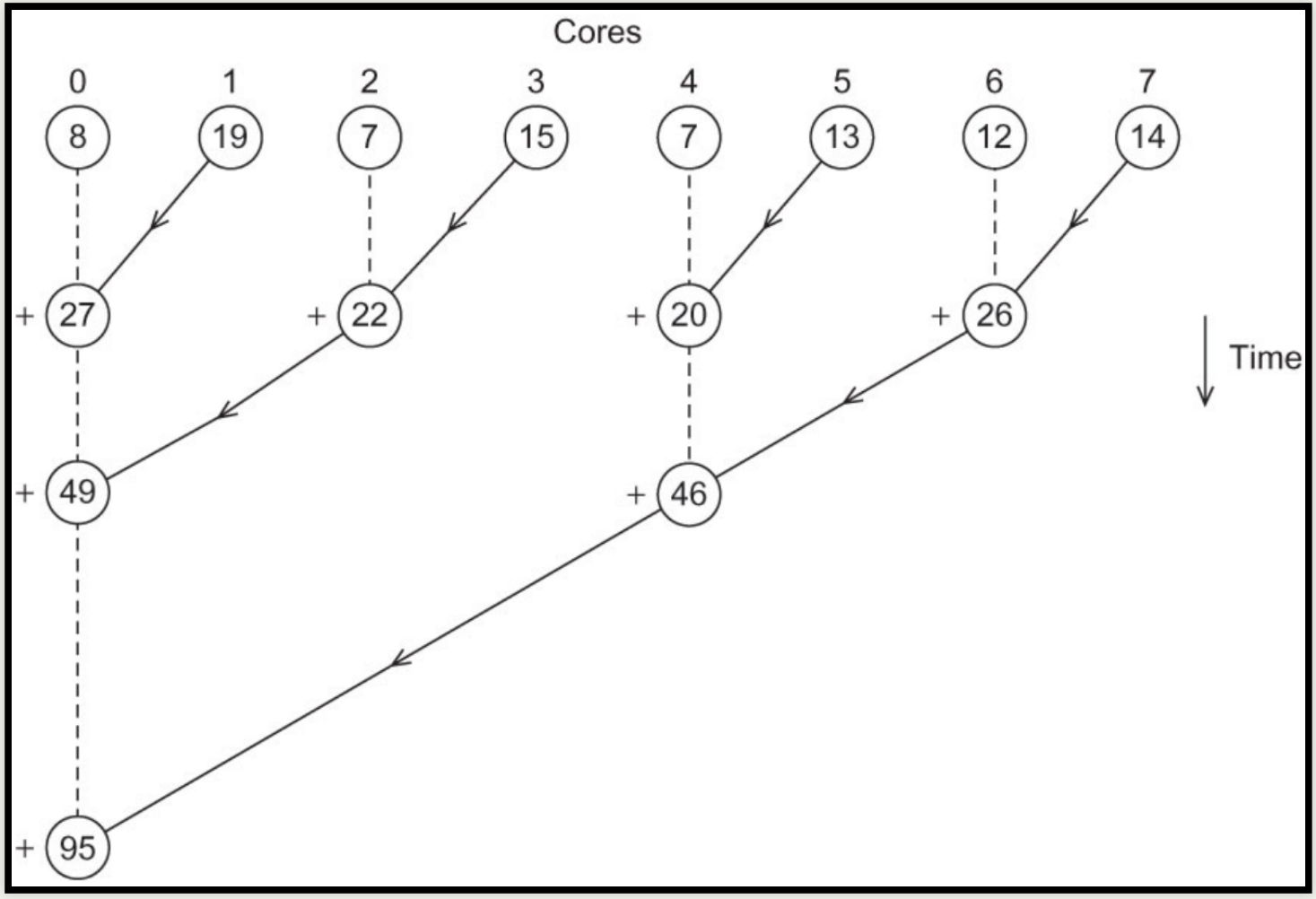
```
/* 'p' cores, p << n
 Each core computes partial sum of n/p values
 */
my_sum = 0;
my_first_i = ...;
my_last_i = ...;
for (my_i = my_first_i; my_i < my_last_i; my_i++) {
        my_x = compute_next_value(...);
        my_sum += my_x;
}
```

Sample Parallelization (contd.)

```
if (I'm the master core) {
        sum = my_x;
        for (each core other than myself) {
                value = received from the core;
                sum += value;
        }
}
else {
        send my_x to the master;
}
```

Can we do better?

Multiple cores forming global sum



Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Task Co-ordination

- Communication
- Load Balancing
- Synchronization

Tip: These determine the performance of your program!

How to write Parallel Programs?

A class of 100 students is handled by 1 Professor, and 2 TAs. After an exam, how can they evaluate all answer scripts?

Task Parallel: Each one corrects a subset of questions in all answer scripts

Data Parallel: Each one corrects all questions for a subset of answer scripts

Task Parallelism

Partition various steps involved in solving a problem (or tasks) among the cores

Data Parallelism

- Partition data (input/intermediate) across multiple cores
- Each core carries out similar operations/tasks

Task Parallel or Data Parallel?

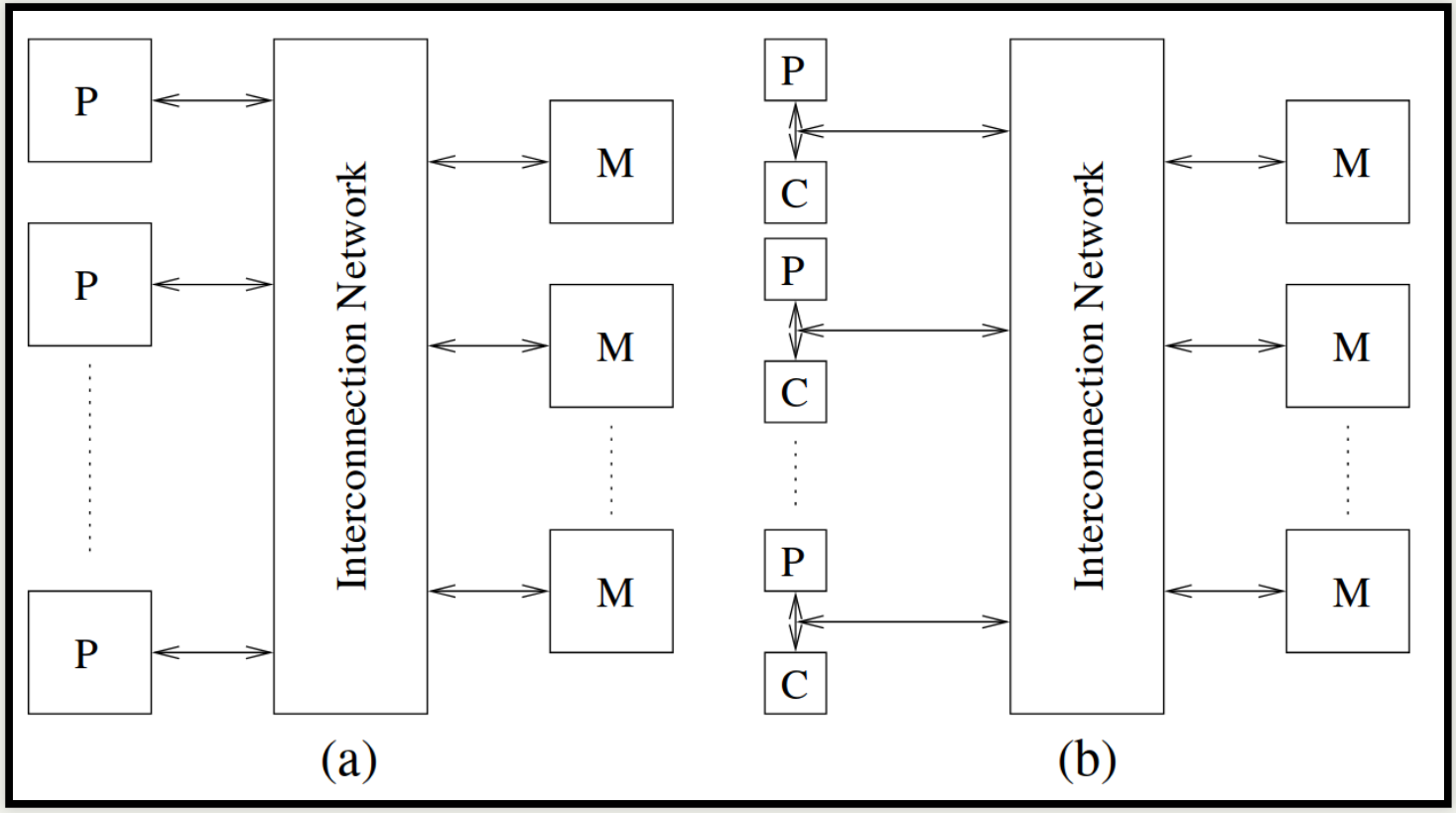
```
for (i = 0; i < 1024; i++)
        c[i] = a[i] + b[i];
```

Parallelism from single instruction on multiple processors

SIMD (Single Instruction and multiple data streams): Data parallelism and (a limiting case of) Task parallelism

MIMD? SPMD (Single Program, multiple data)?

Shared Memory System



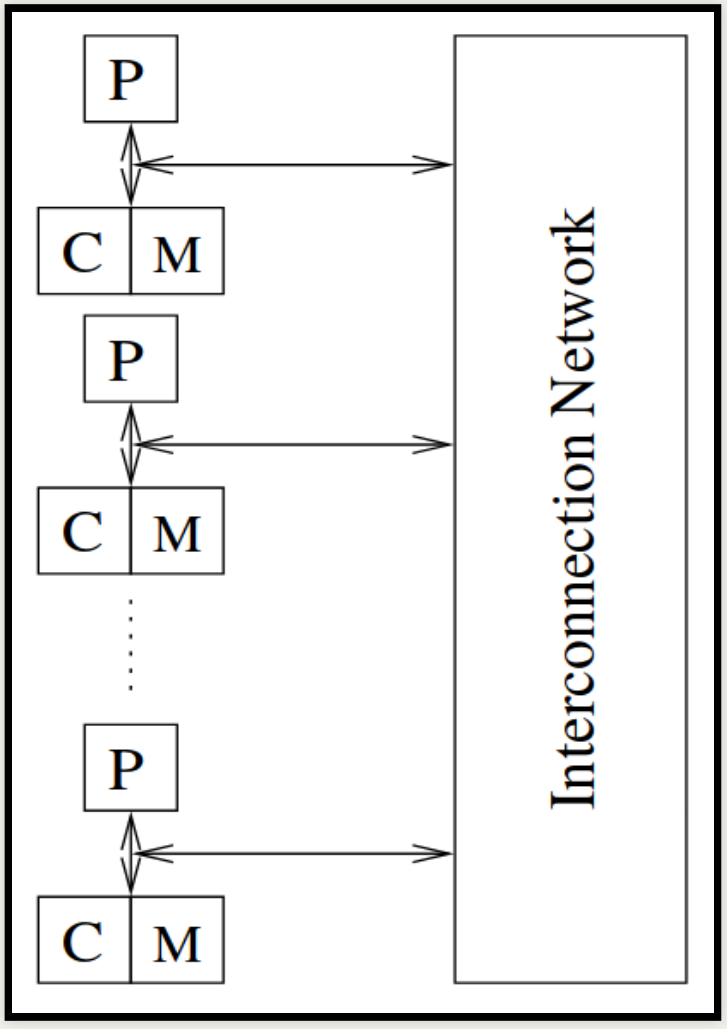
(a) & (b) are examples of simple shared-memory machines

Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Shared Memory System

- Cores can share access to the computer's memory
- Cores can coordinate by reading and writing to shared memory locations

Is this a Shared Memory System?

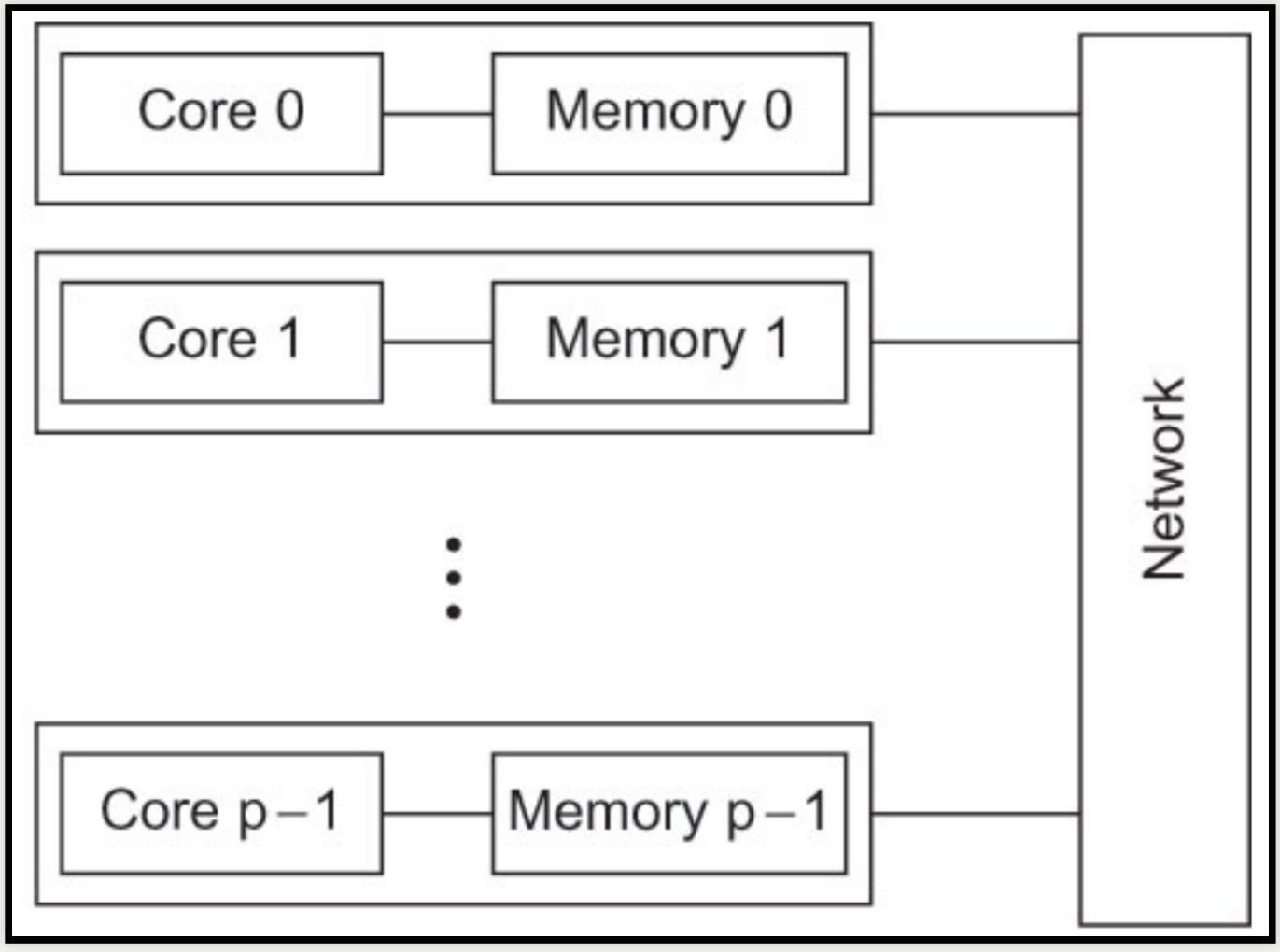


Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Is this a Shared Memory System?

Shared memory is a logical organization of the system (created by OS, assisted by hardware). The physical organization shown can be shared, but access times to memory location may not be uniform.

Distributed Memory System



Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Distributed Memory System

- Each core has its private memory
- Cores can co-ordinate by explicitly sending messages.

Ideal Parallel Computer

PRAM: Parallel Random Access Machine

All 'p' processors can uniformly access global memory of unbounded size

Subclasses of PRAM

- Exclusive-Read, Exclusive-Write (EREW)
- Concurrent-Read, Exclusive-Write (CREW)
- Exclusive-Read, Concurrent-Write (ERCW)
- Concurrent-Read, Concurrent-Write (CRCW)

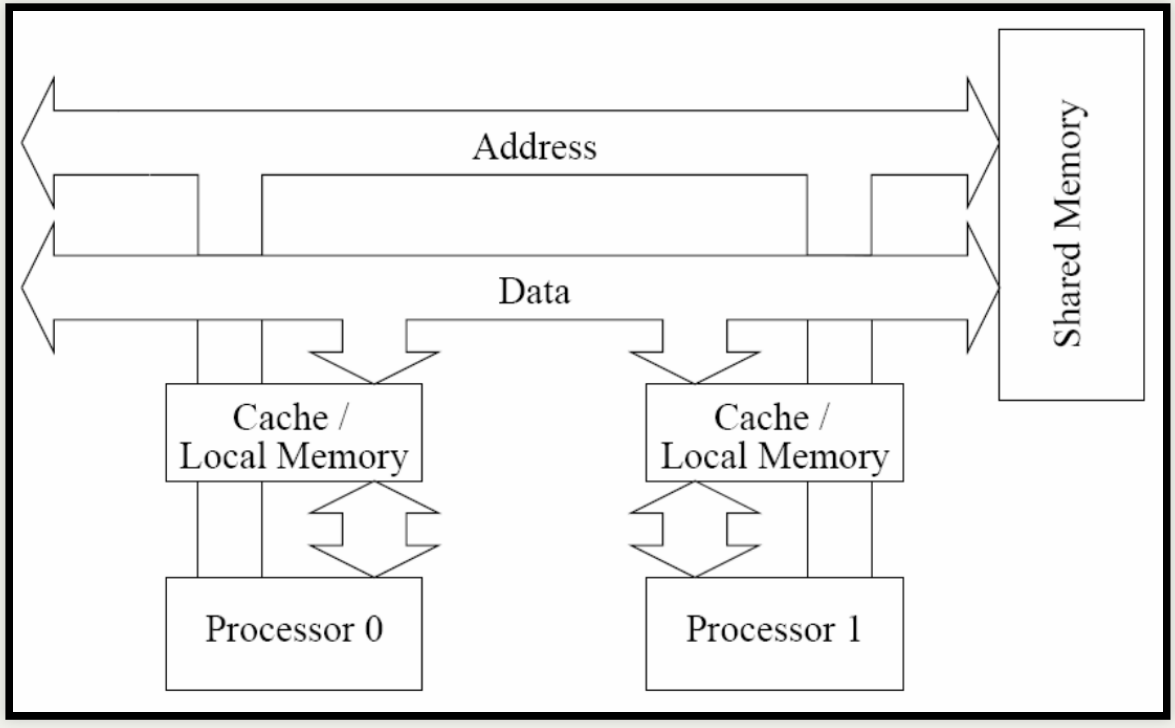
Concurrent Write Arbitration

- Common
- Arbitrary
- Priority
- Sum

Interconnect Networks

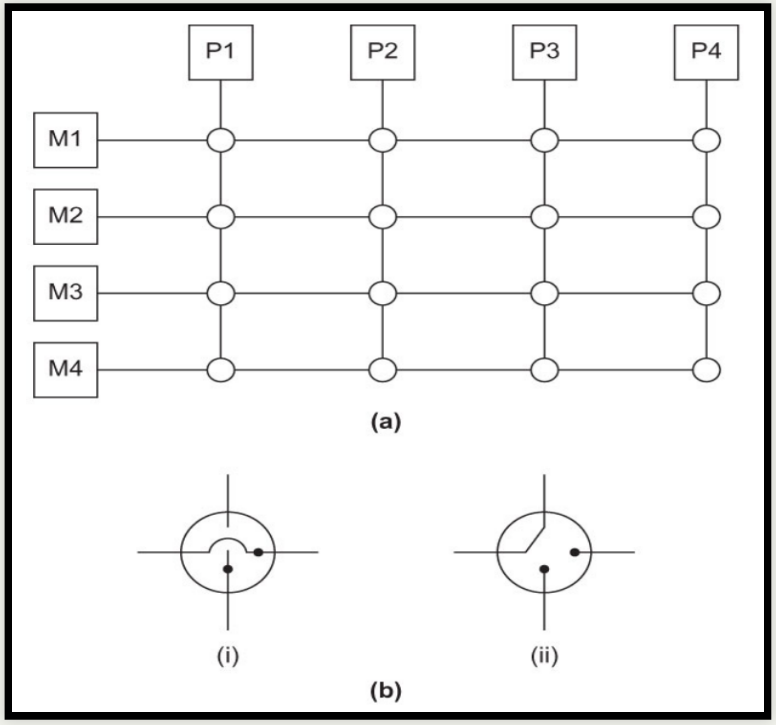
- Platform based classification
  - Shared Memory Interconnects
  - Distributed Memory Interconnects
- Architecture based classification
  - Static
  - Dynamic

Shared Memory Interconnects - Bus (Direct)



Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

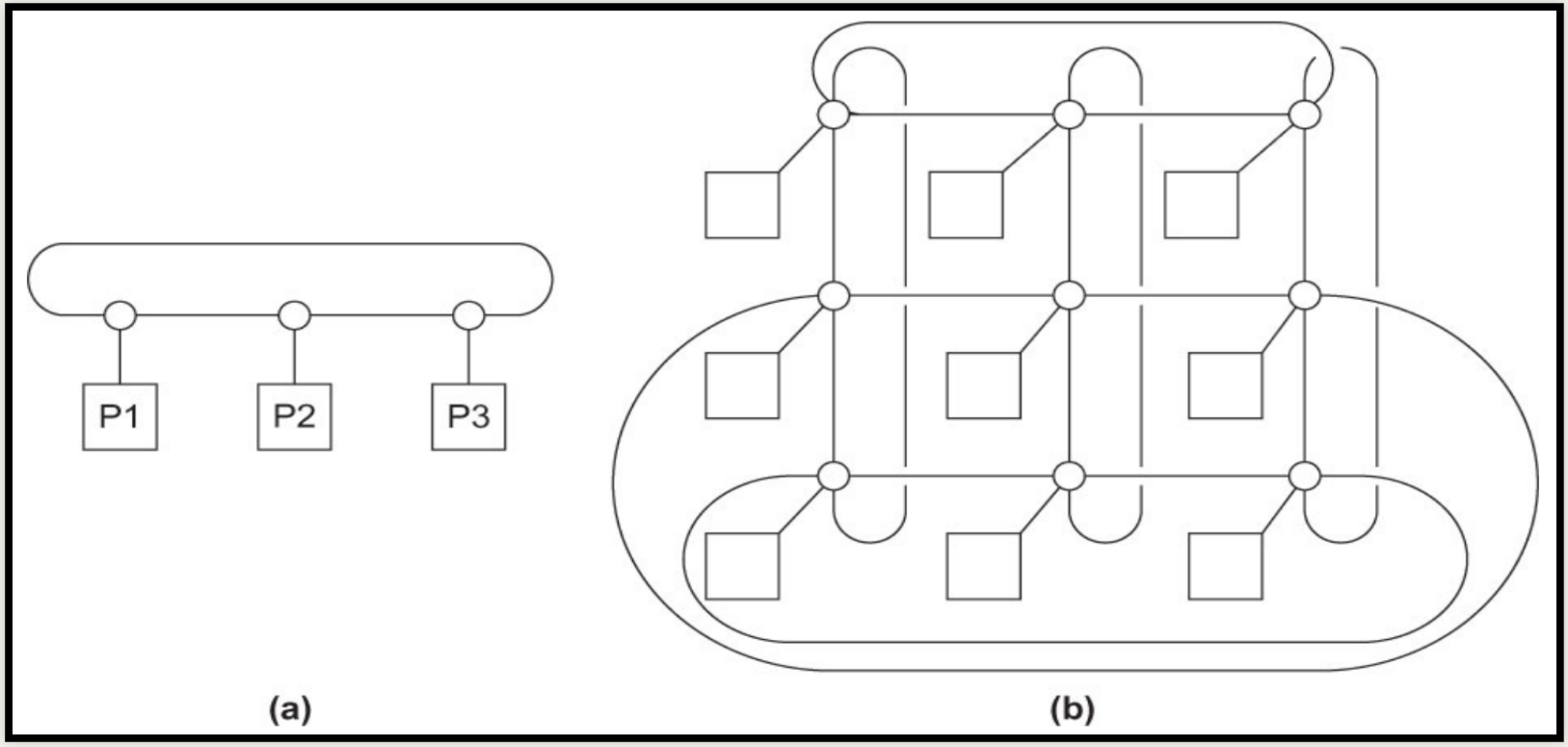
Shared Memory Interconnects - Crossbar (Indirect)



(a) 4x4 crossbar switch, (b) configuration of internal switches

Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

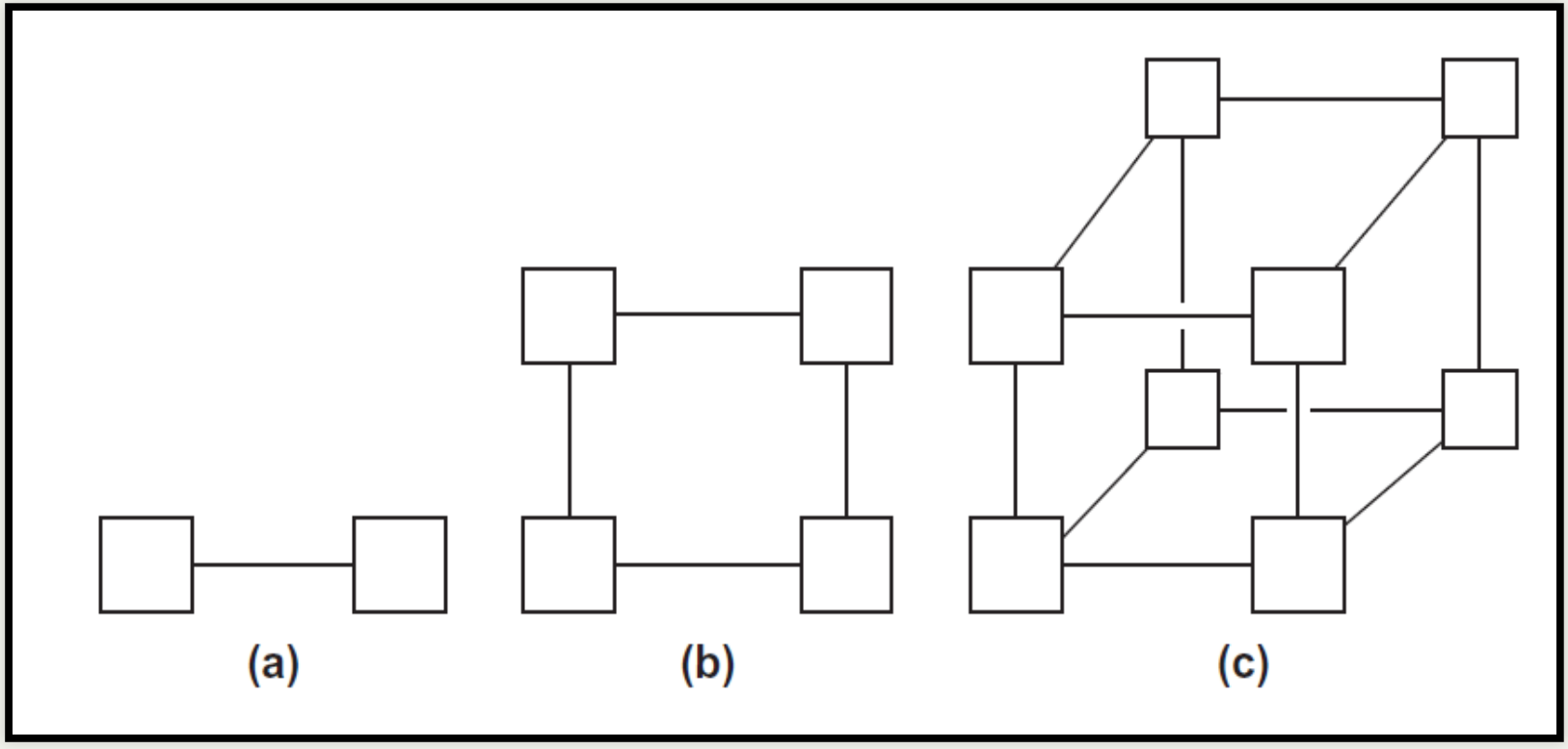
Distributed Memory Interconnects - Direct



(a) ring, (b) torroidal

Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

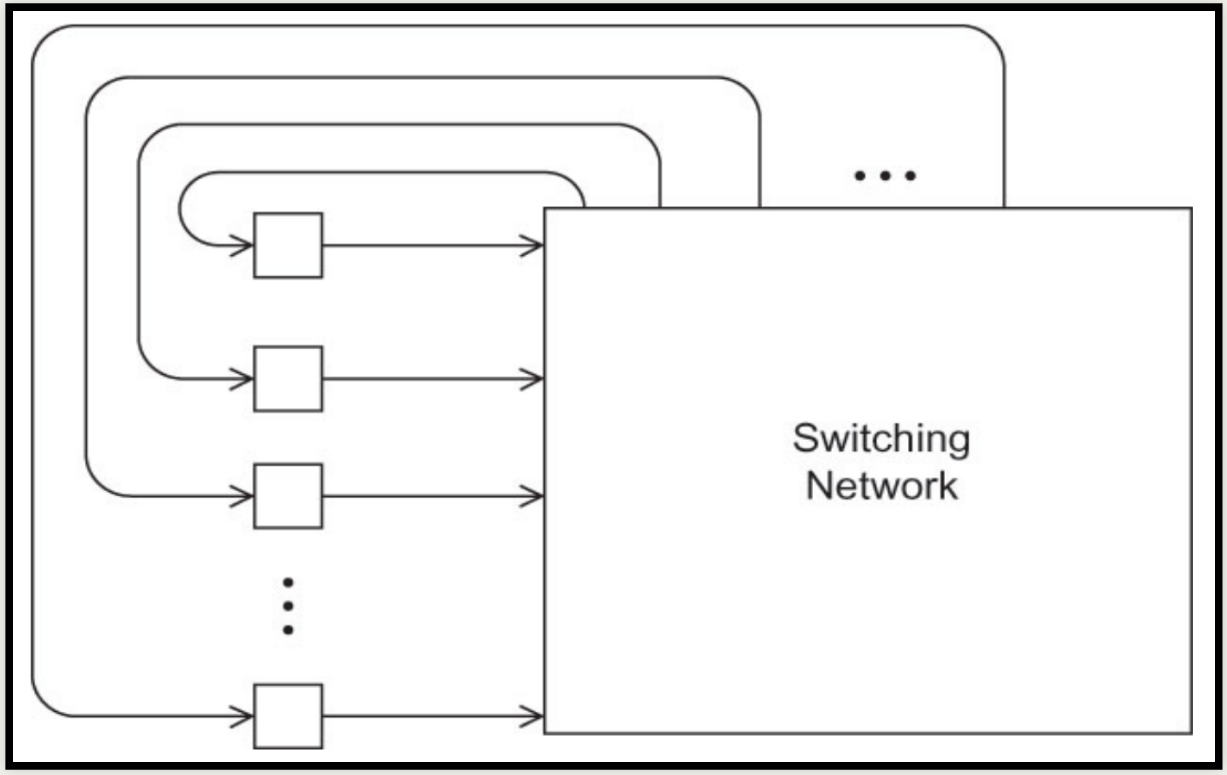
Distributed Memory Interconnects - Direct



Hypercube of (a) one, (b) two, and (c) three dimensions

Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

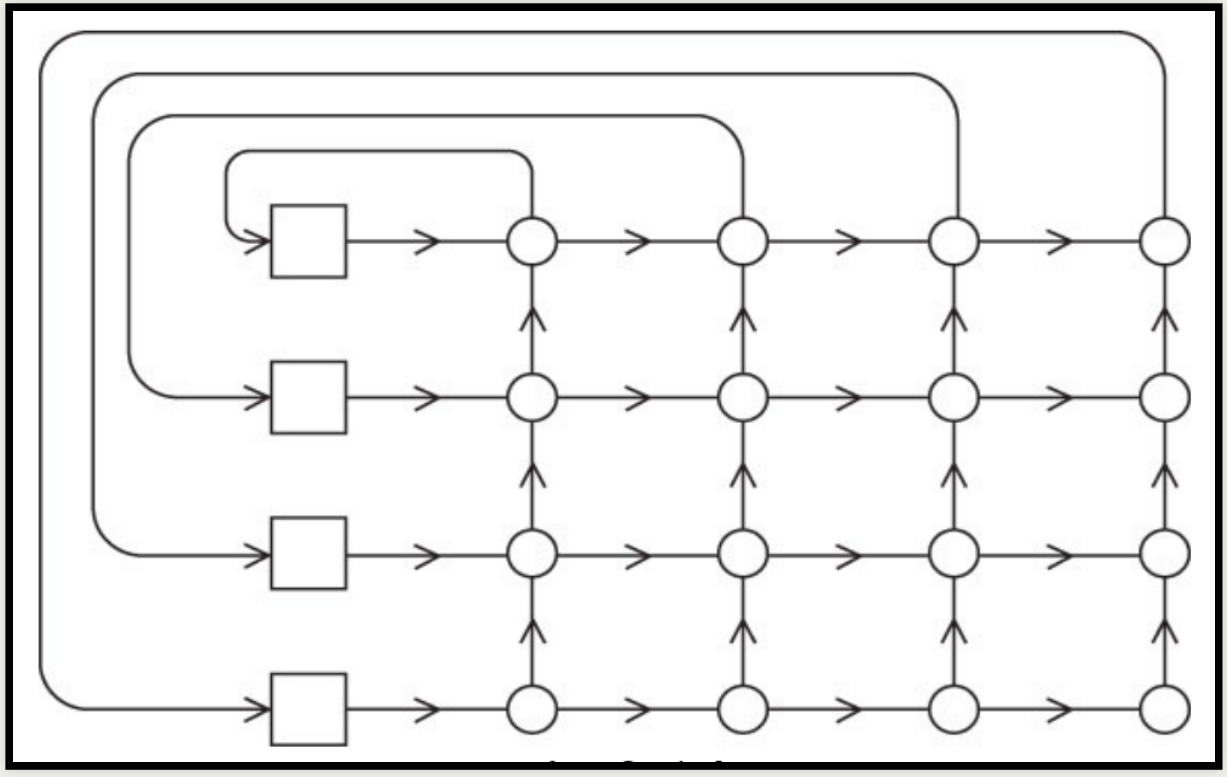
Distributed Memory Interconnects - Indirect



A generic indirect switch

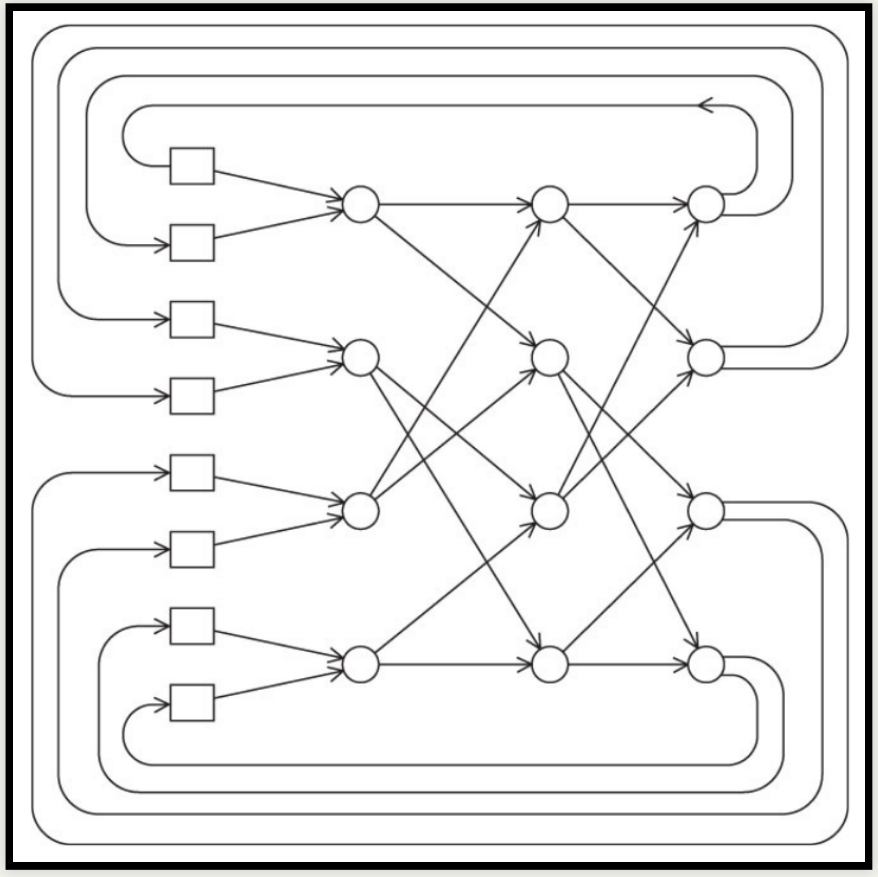
Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Distributed Memory Interconnects - Crossbar (Indirect)



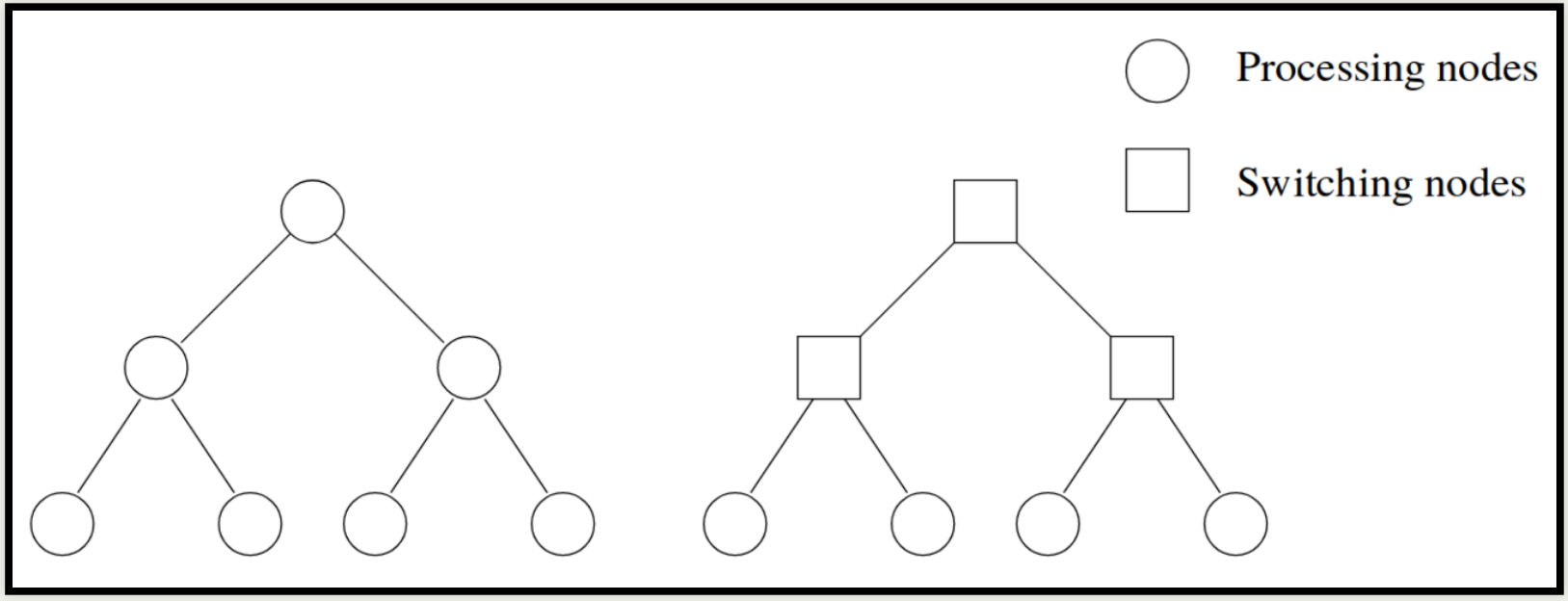
Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Distributed Memory Interconnects - Omega N/W (Indirect)



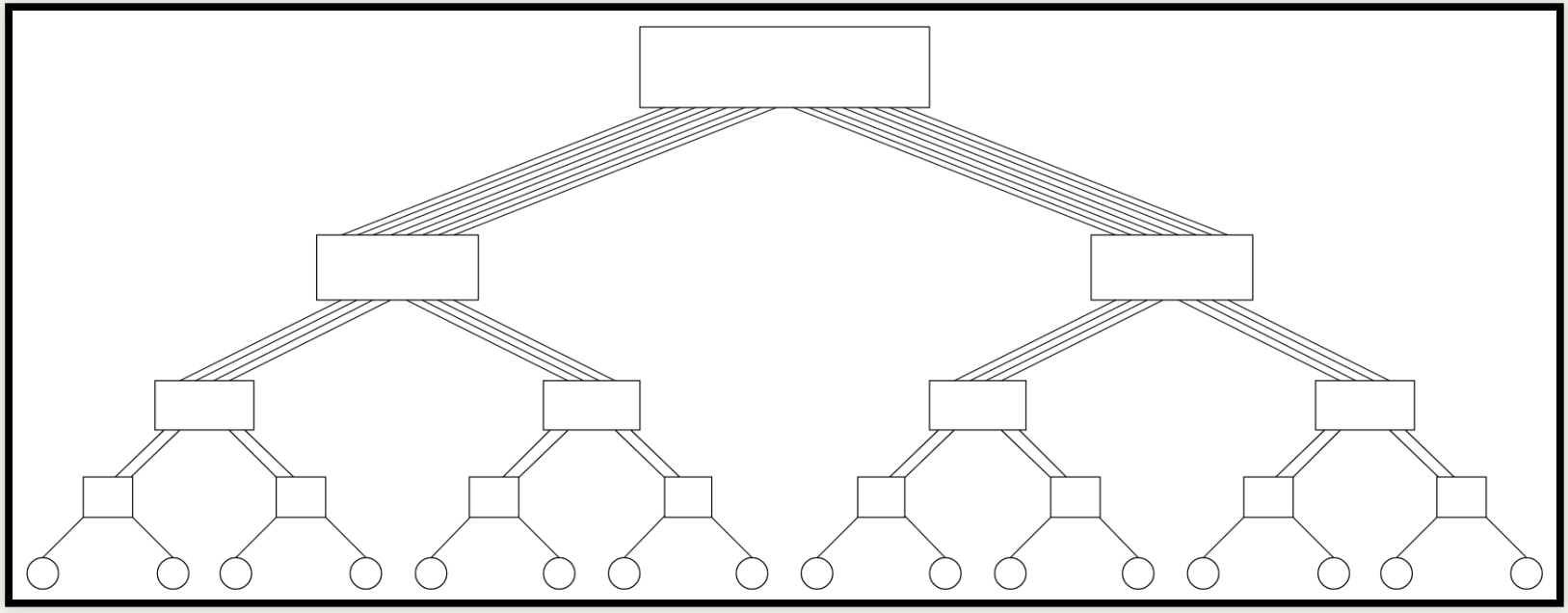
Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Tree networks



Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Tree networks



Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Static Interconnect Networks

Consists of a number of point-to-point links (Direct Network)

Dynamic Interconnect Networks

Consists of switching elements that the various processors attach to (Indirect Network)

Latency and Bandwidth of ICN

Latency - The time elapsed between the source beginning to transmit data and the destination starting to recieve the first byte.

Bandwidth - The rate at which the destination receives data after it has started to receive the first byte.

Evaluation Metrics for ICNs

Diameter - The maximum distance between any two nodes

Connectivity - The minimum number of arcs that must be removed to break it into two disconnected networks

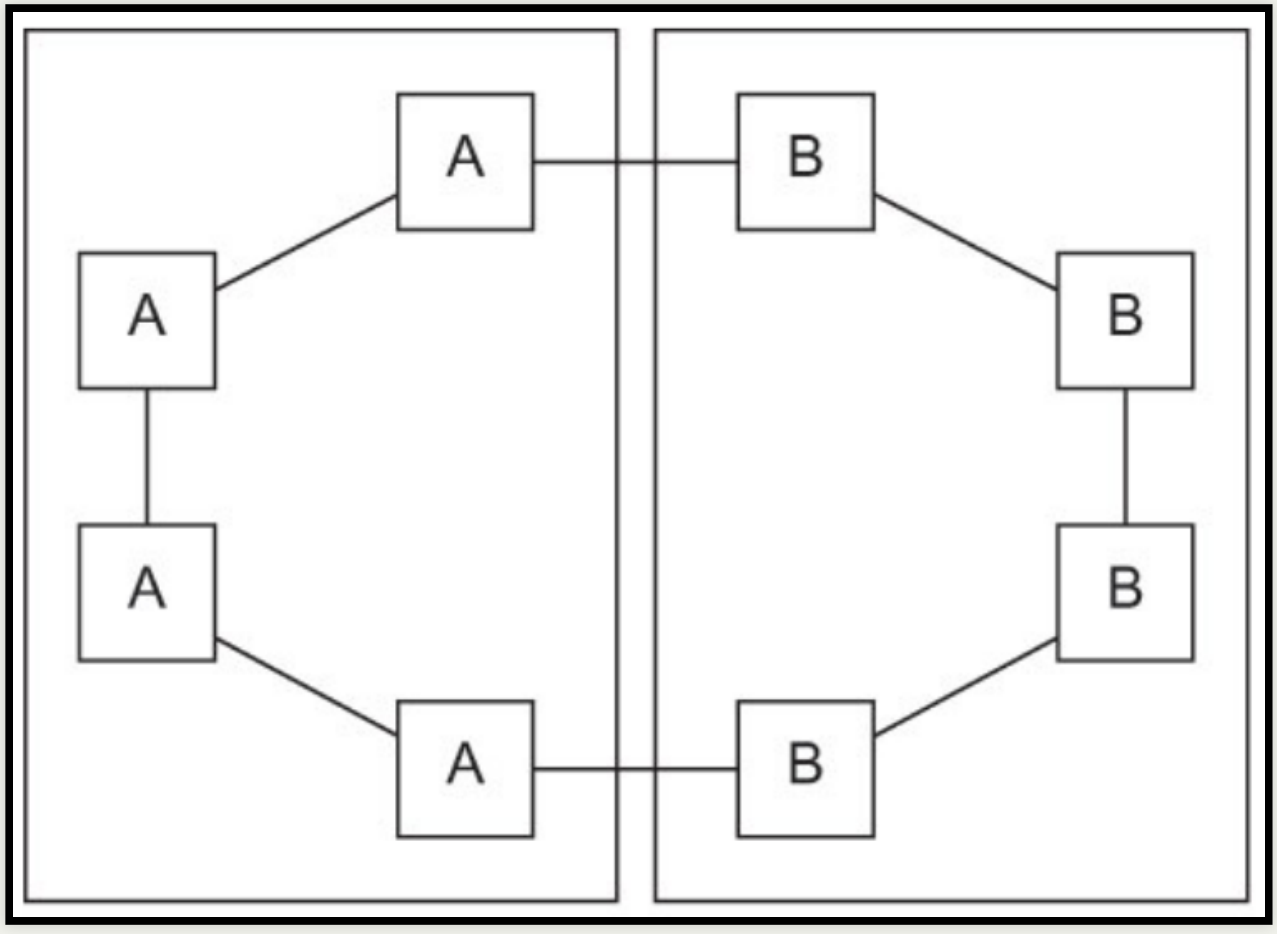
Cost - Number of links in the network

Evaluation Metrics for ICNs

Bi-section width - The minimum number of arcs that must be removed to partition the network into two equal halves

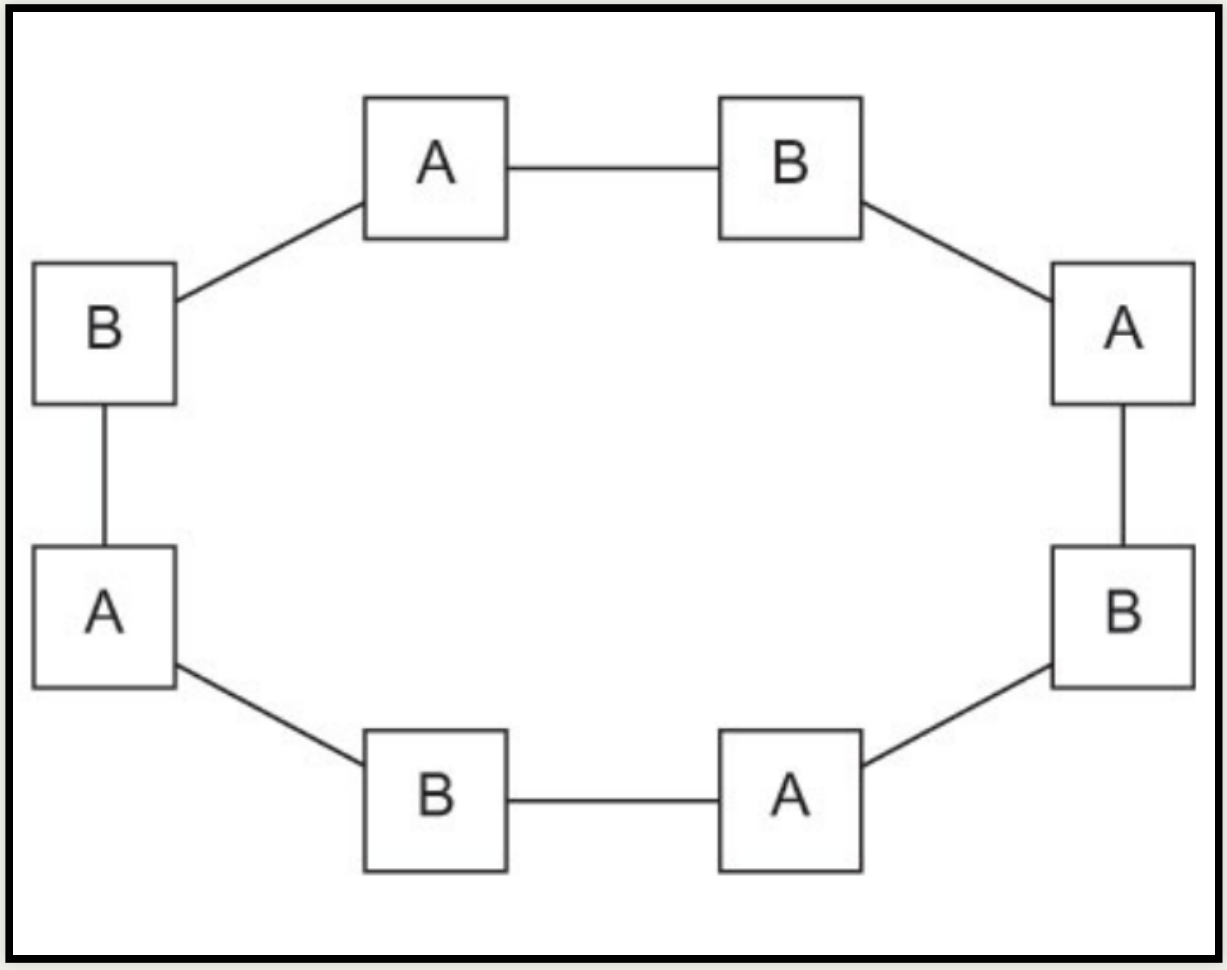
Bi-section Bandwidth - Minimum volume of communication allowed between any two halves of a network

Bi-section width



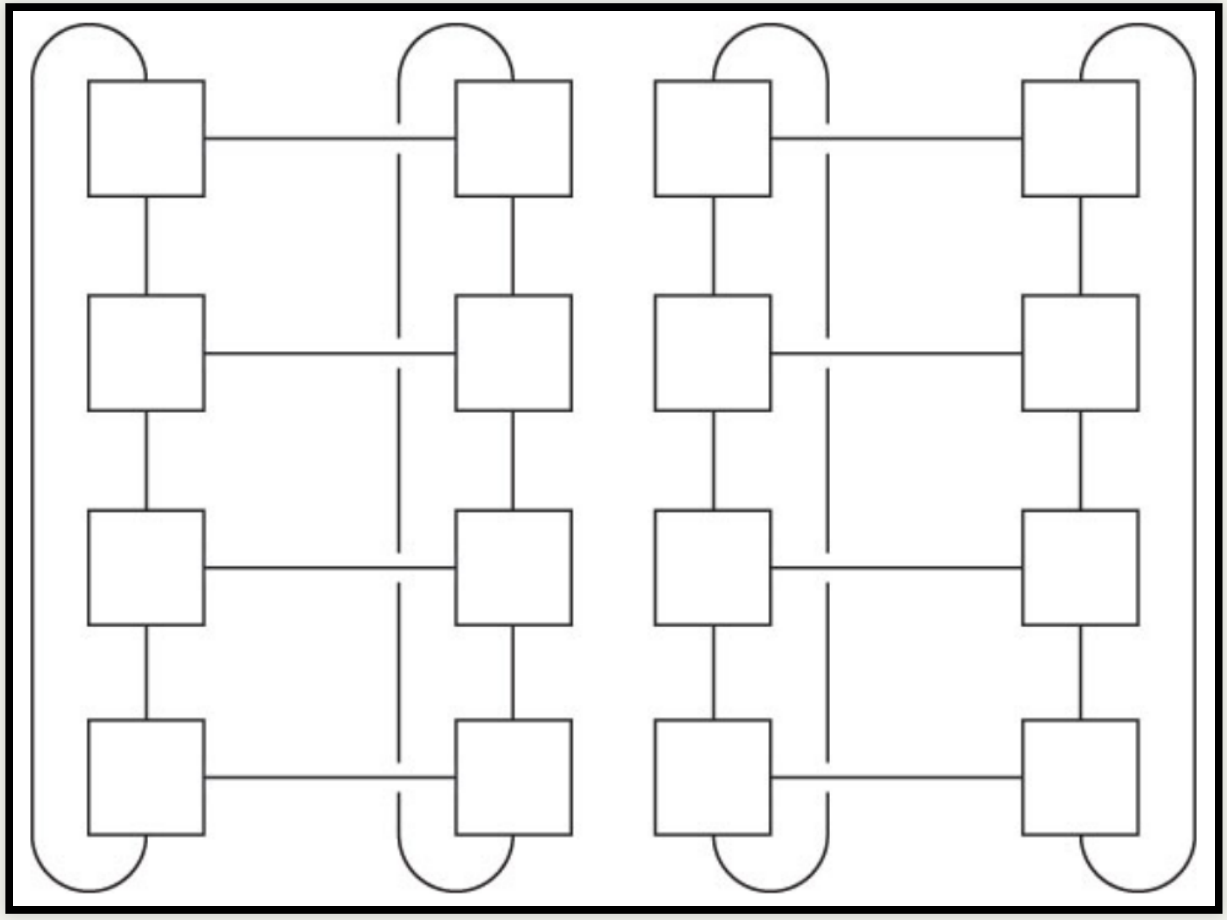
Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Bi-section width



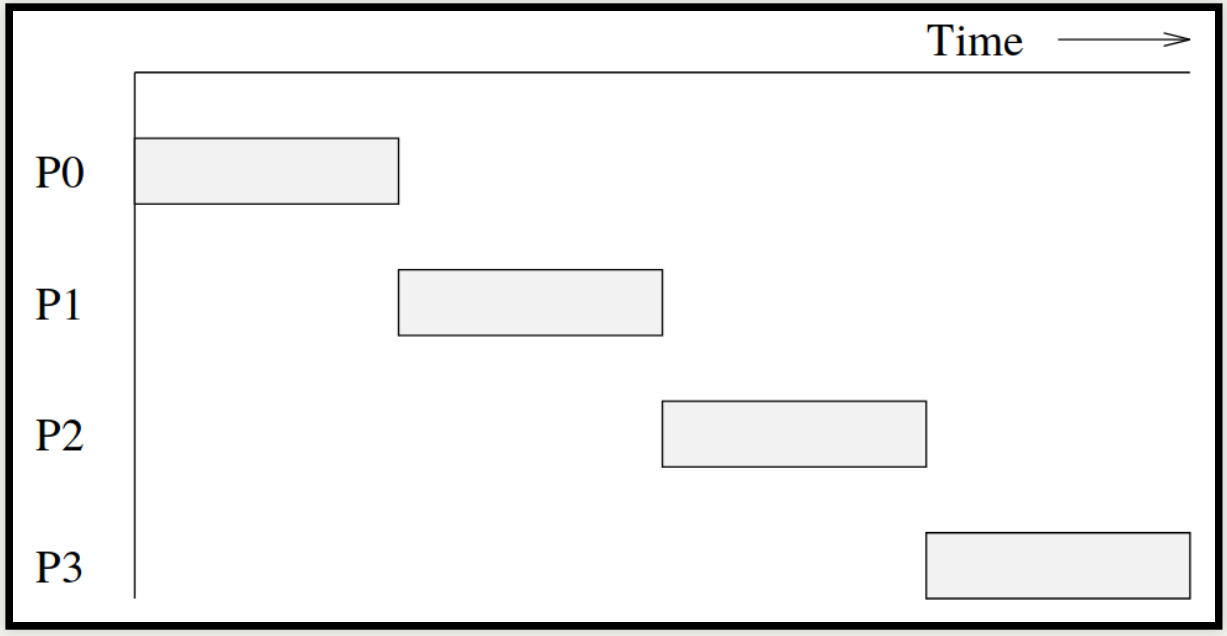
Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Bi-section width



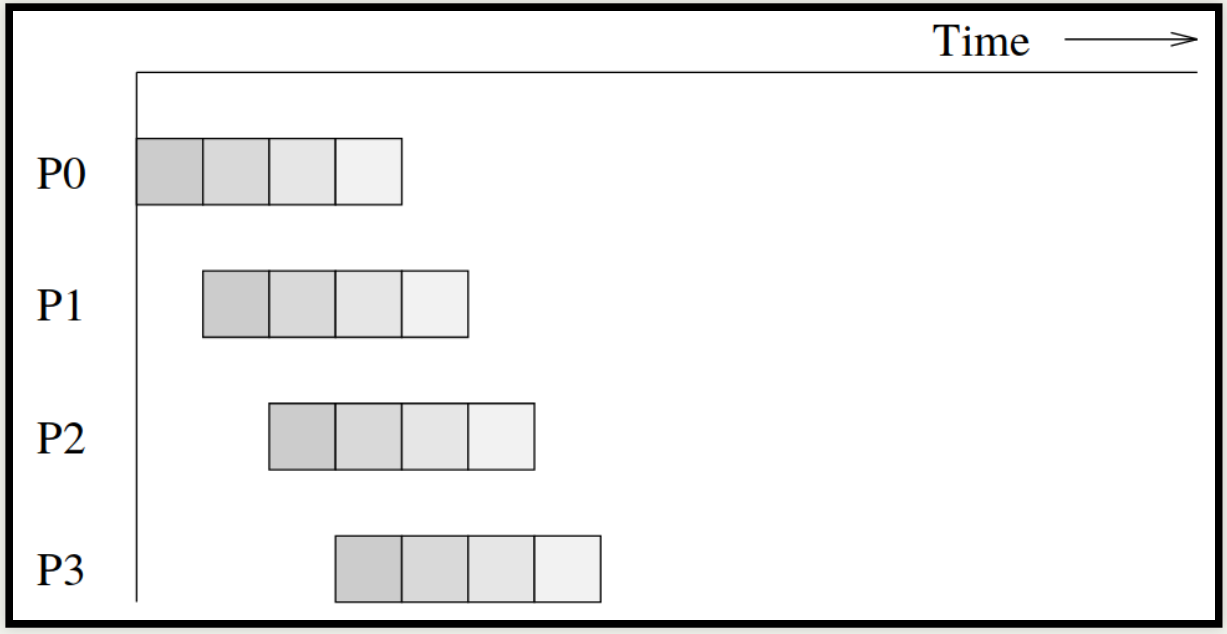
Source: Peter S. Pacheco, "An Introduction to Parallel Programming, ISBN:978-93-80931-75-3"

Store-and-forward routing



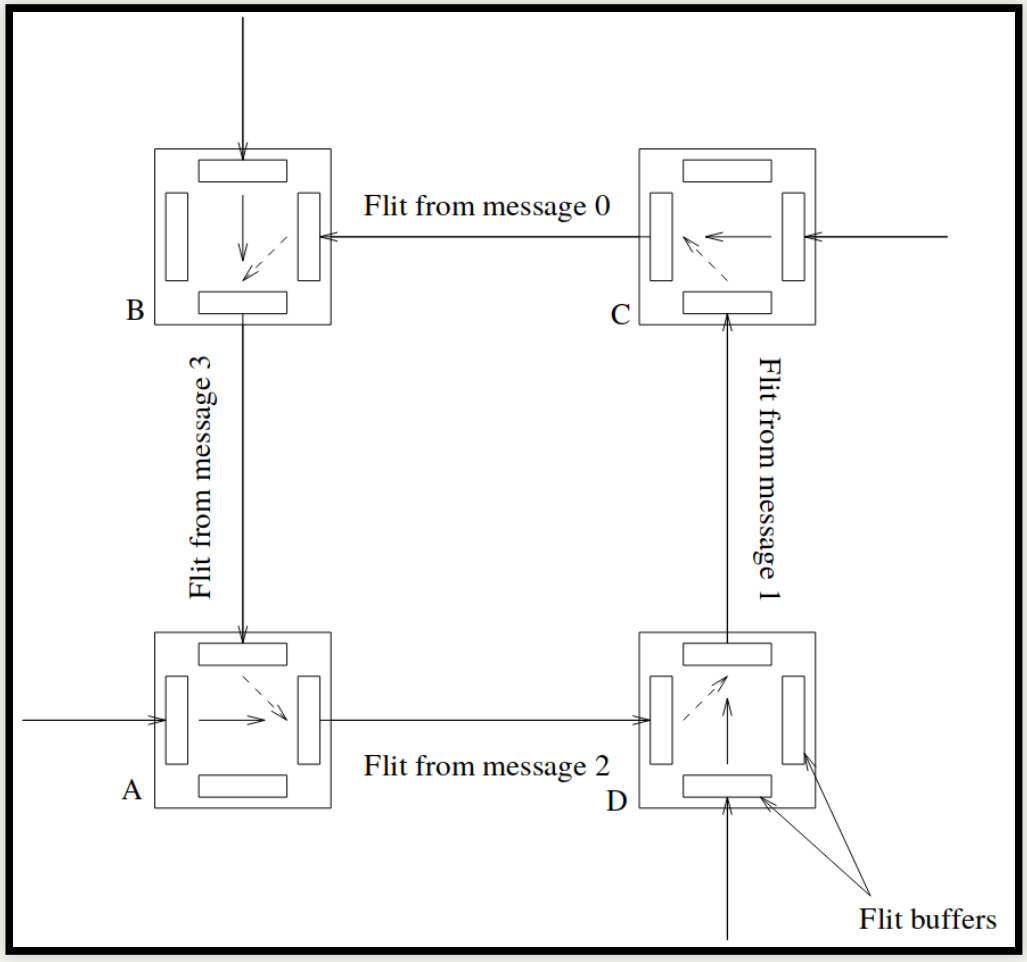
Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Cut-through routing



Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Deadlocks in Cut-through routing



Source: Ananth Grama et. al., "An Introduction to Parallel Computing, ISBN:978-81-317-0807-1"

Communication cost

- Startup time (t_s)
- Per-hop time (or node-latency) (t_h)
- Per-word transfer time ($t_w = \frac{1}{r}$)

For store-and-forward routing: $t_{comm} = t_s + l(t_h + mt_w)$

For cut-through routing: $t_{comm} = t_s + lt_h + mt_w$

Communication cost

Implications of cut-through routing costs ($t_{comm} = t_s + lt_h + mt_w$)

- Communicate in bulk
- Minimize volume of data transfer
- Minimize distance of data transfer

Realistic communication cost

$$t_{comm} = t_s + mt_w$$

Holds true because $t_s >> t_h$, and $mt_w >> lt_h$

Communication cost

(shared-memory programs)

- Memory layout
- Thrashing in the caches
- Overheads to maintain coherence
- False sharing
- Contention in shared accesses

Parallel Software

- Shared-memory programs:
  - Start a ***single process***
  - This process then creates/forks threads
  - ***Threads*** do all the work/task
- Distributed-memory systems:
  - Start ***multiple processes***
  - ***Processes*** do all the work/task

Single Program Multiple Data (SPMD)

```
if (I am thread/process i) {
        // do this
}
else {
        // do something else
}
```

Writing Parallel Programs (Recap)

- Divide work among threads (or processes) such that:
  - each thread (or processes) gets roughly same amount of work
  - communication is minimized
- Arrange for threads (or processes) to synchronize
- Arrange for communication among threads (or processes)

Non-determinism - Shared memory

- Arises when processors execute asynchronously
  - Race condition
  - Critical section
  - Mutual-exclusion lock

Input-Output

- one process/thread can access `stdin`
- all processes/threads can access `stdout` and `stderr`
- To avoid non-determinism, except for debugging, only one process accesses `stdout`

Speedup

- Serial run-time = T_{serial}
- Parallel run-time = $T_{parallel}$
- Number of cores = p

$$\text{linear speedup: } T_{parallel} = \frac{T_{serial}}{p}$$

$$\text{Effect of overheads: } T_{parallel} = \frac{T_{serial}}{p} + T_{overhead}$$

Speedup and Efficiency

$$\text{Speedup: } S = \frac{T_{serial}}{T_{parallel}}$$

$$\text{Efficiency: } E = \frac{S}{p}$$

$$E = \frac{T_{serial}}{p \cdot T_{parallel}}$$

Amdahl's Law

Speedup: $S = \frac{1}{(1-f)+f/p}$

As $p \to \infty$, $S \to \frac{1}{(1-f)}$

Task: Find algorithms with larger f

Foster's methodology

- Partitioning: Divide computation and data into small tasks
- Communication: Determine communication between tasks
- Aggregation: Combine tasks and communication into larger tasks
- Mapping: Assign composite tasks to processes/threads

Exercise: Find histogram of `N` floats (in parallel)