ΔΠΜΣ Προηγμένες Τεχνολογίες Πληροφορικής και Υπηρεσίες

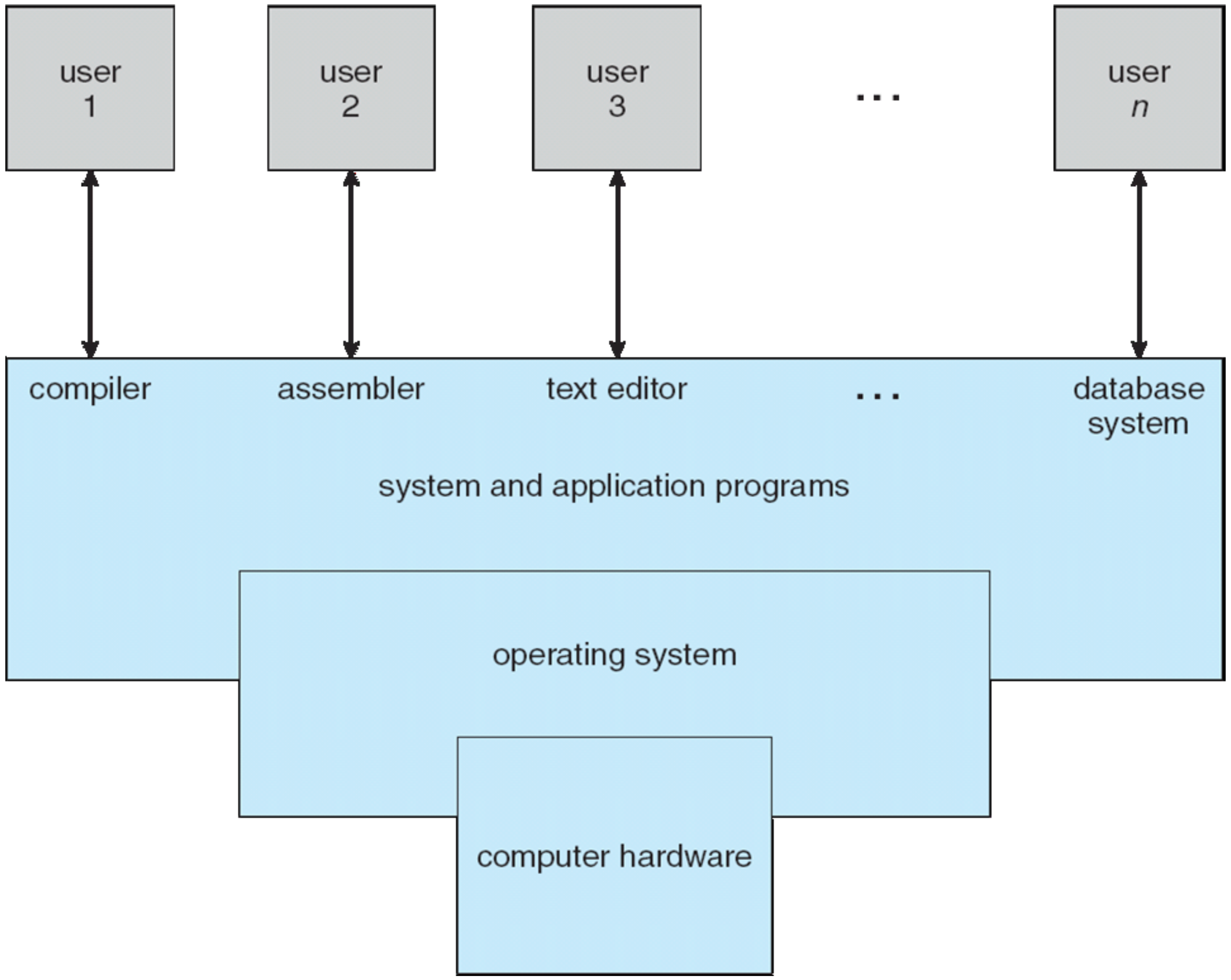
Προηγμένα Θέματα Αρχιτεκτονικής Υπολογιστών

Parallel Computing

Καθηγητής Αγγελίδης Παντελής

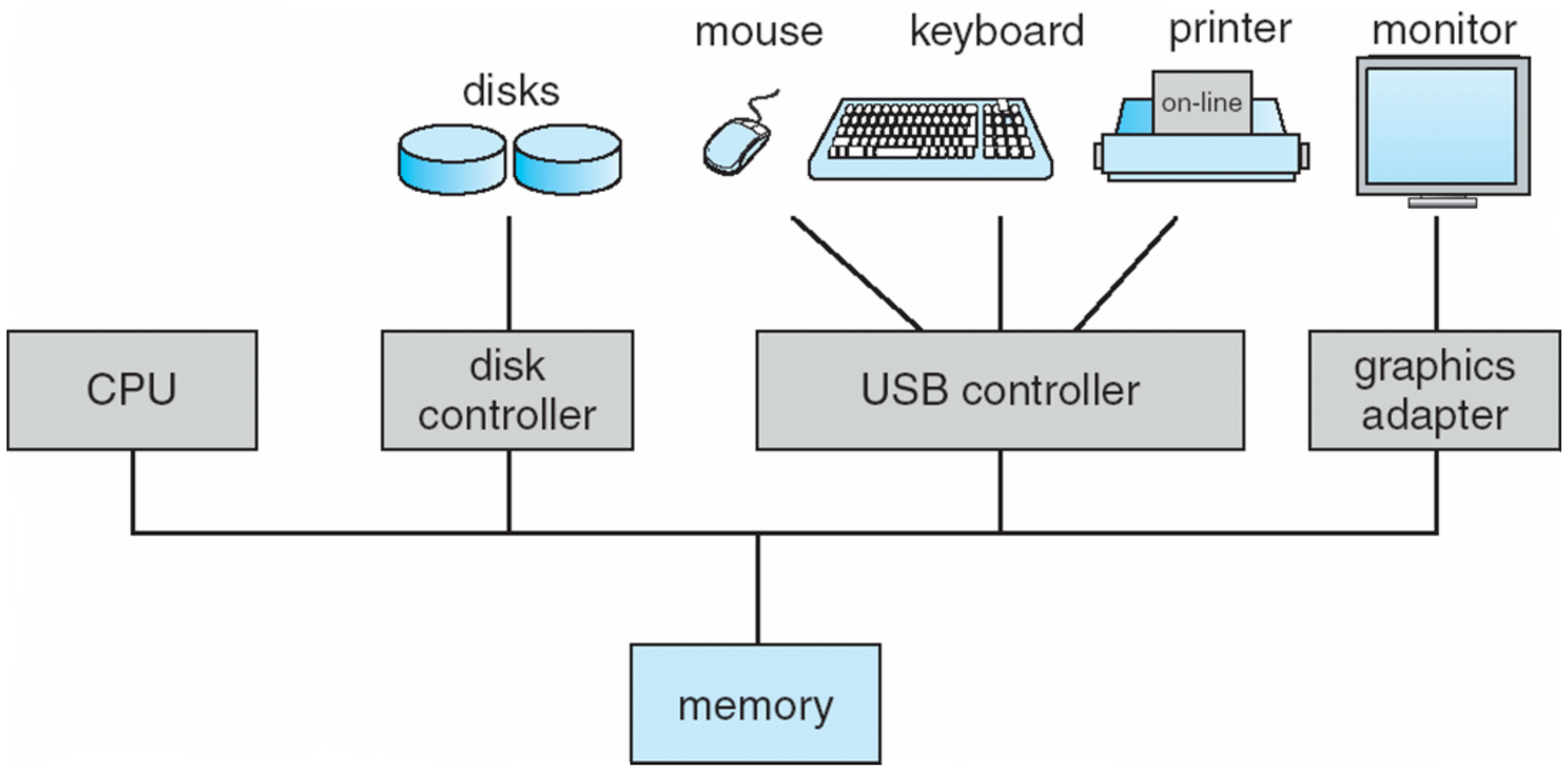
paggelidis@uowm.gr

Four Components of a Computer System

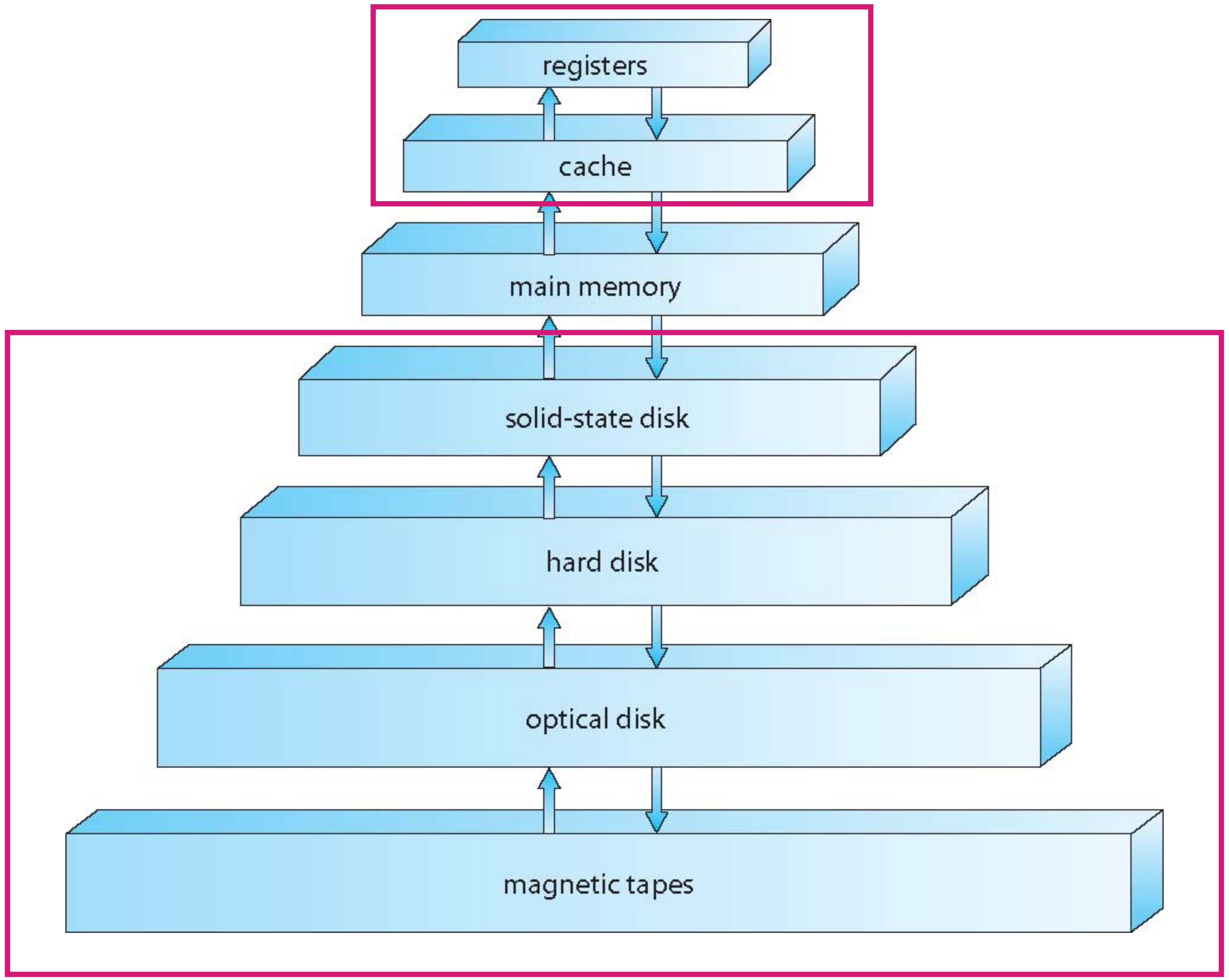


Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



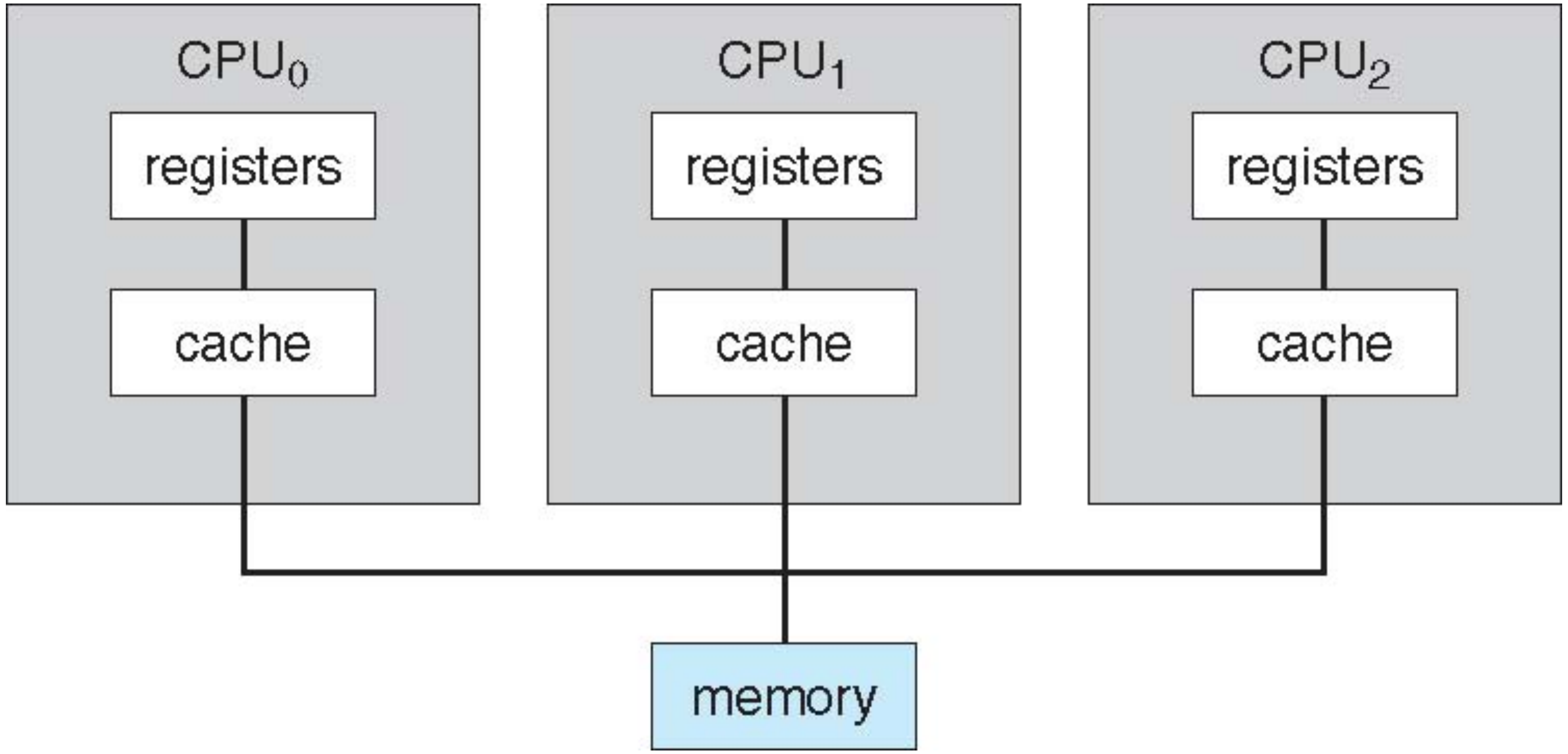
Storage-Device Hierarchy



Computer-System Architecture

- Most systems use a single general-purpose processor
  - Most systems have special-purpose processors as well
- **Multiprocessors** systems growing in use and importance
  - Also known as **parallel systems**, **tightly-coupled systems**
  - Two types:
    1. **Asymmetric Multiprocessing** – each processor is assigned a specific task.
    2. **Symmetric Multiprocessing** – each processor performs all tasks

Symmetric Multiprocessing Architecture



Operating System Structure

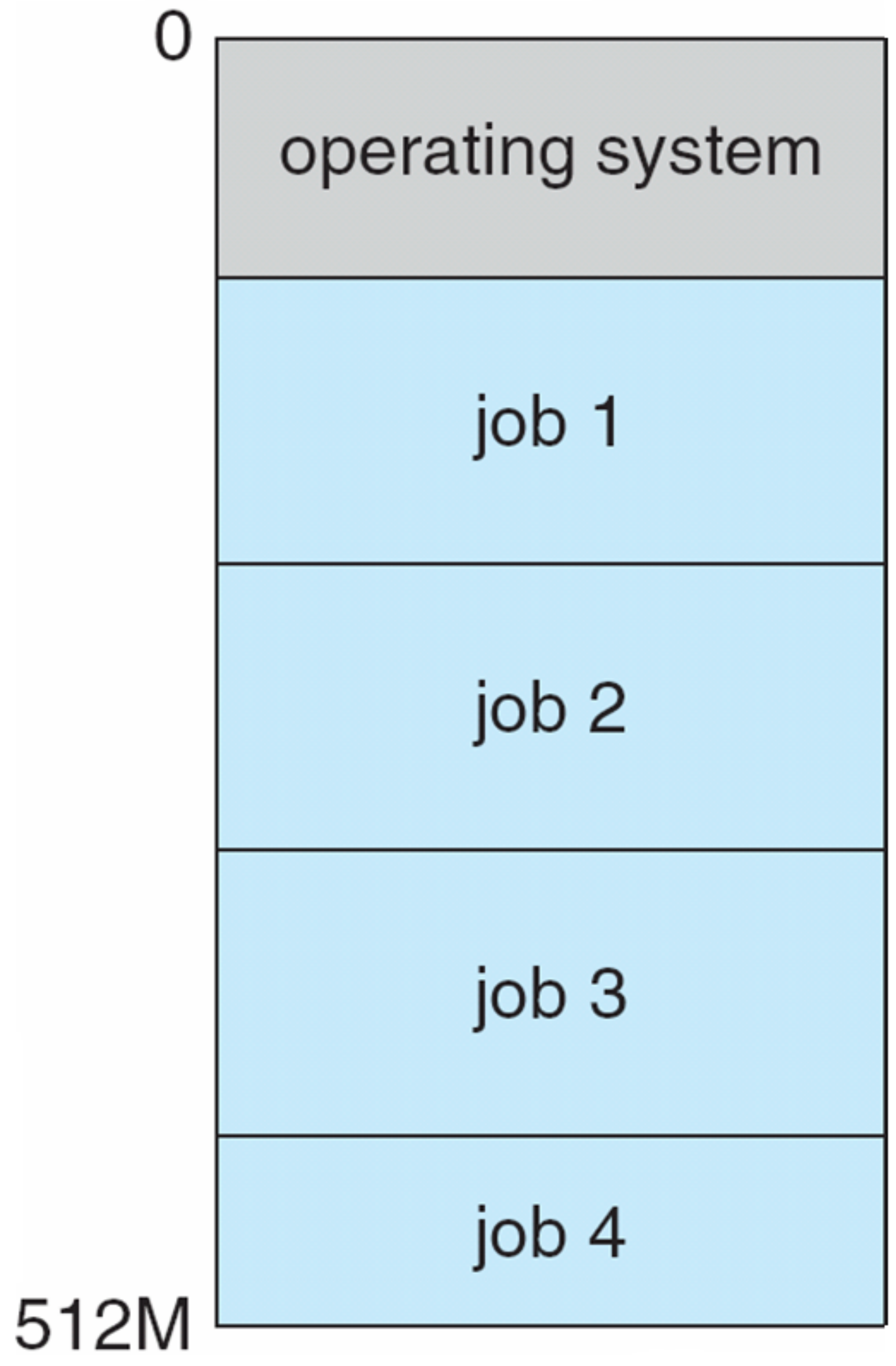
- **Multiprogramming** (**Batch system**) needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job

- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**

Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

Memory Layout for Multiprogrammed System



Storage Management

- OS provides uniform, logical view of information storage
  - logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)

- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Backup files onto stable (non-volatile) storage media

Distributed systems

- Virtually all large computer-based systems are now distributed systems.
- Information processing is distributed over several computers rather than confined to a single machine.
- Distributed software engineering is therefore very important for enterprise computing systems.

System types

- Personal systems that are not distributed and that are designed to run on a personal computer or workstation.
- Embedded systems that run on a single processor or on an integrated group of processors.
- Distributed systems where the system software runs on a loosely integrated group of cooperating processors linked by a network.

Distributed system characteristics

- Resource sharing
  - Sharing of hardware and software resources.
- Openness
  - Use of equipment and software from different vendors.
- Concurrency
  - Concurrent processing to enhance performance.
- Scalability
  - Increased throughput by adding new resources.
- Fault tolerance
  - The ability to continue in operation after a fault has occurred.

Distributed system disadvantages

- Complexity
  - Typically, distributed systems are more complex than centralised systems.
- Security
  - More susceptible to external attack.
- Manageability
  - More effort required for system management.
- Unpredictability
  - Unpredictable responses depending on the system organisation and network load.

Distributed systems architectures

- Client-server architectures
  - Distributed services which are called on by clients. Servers that provide services are treated differently from clients that use services.
- Distributed object architectures
  - No distinction between clients and servers. Any object on the system may provide and use services from other objects.

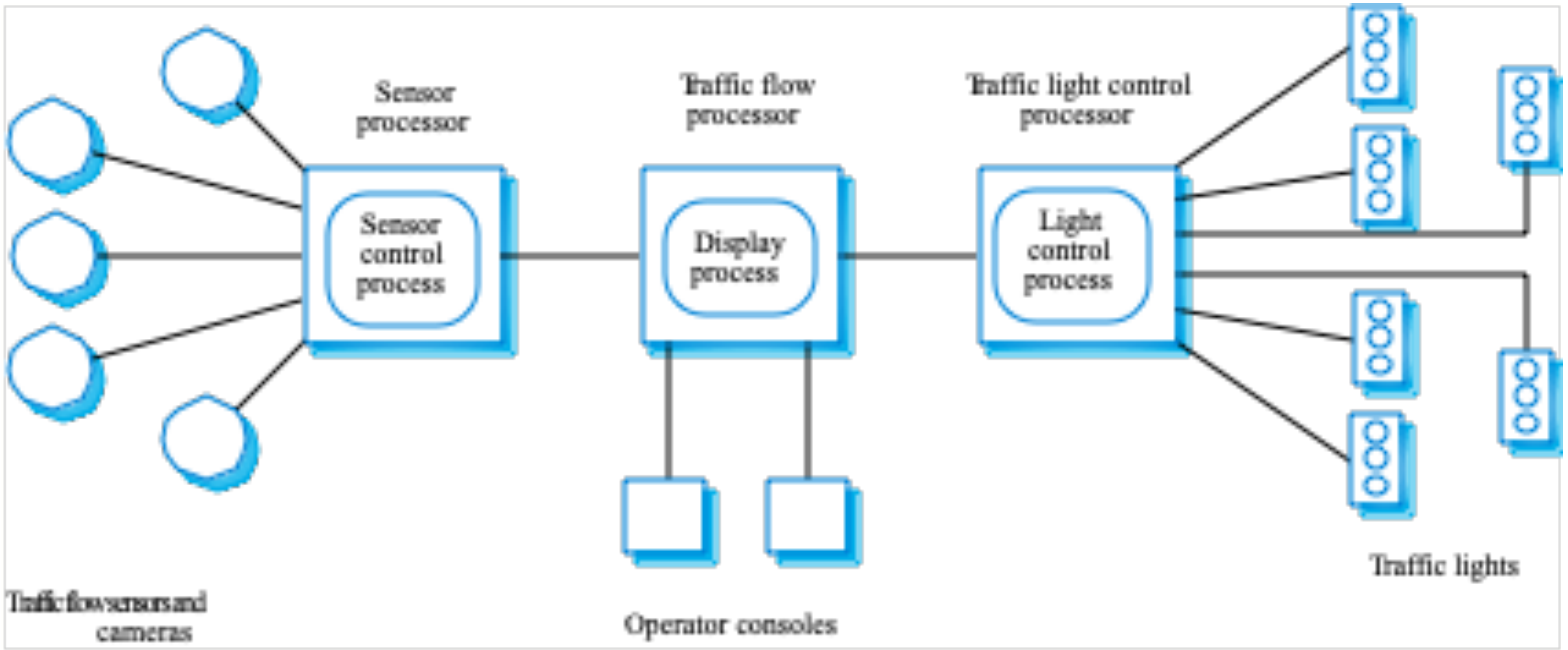
Middleware

- Software that manages and supports the different components of a distributed system. In essence, it sits in the *middle* of the system.
- Middleware is usually off-the-shelf rather than specially written software.
- Examples
  - Transaction processing monitors;
  - Data converters;
  - Communication controllers.

Multiprocessor architectures

- Simplest distributed system model.
- System composed of multiple processes which may (but need not) execute on different processors.
- Architectural model of many large real-time systems.
- Distribution of process to processor may be pre-ordered or may be under the control of a dispatcher.

A multiprocessor traffic control system



https://www.youtube.com/watch?v=IZfWjg3U3mA

Multiprocessors

ELEC 6200 Computer Architecture and Design
Instructor: Dr. Agrawal

Yu-Chun Chen

Why Choose a Multiprocessor?

- A single CPU can only go so fast, use more than one CPU to improve performance
- Multiple users
- Multiple applications
- Multi-tasking within an application
- Responsiveness and/or throughput
- Share hardware between CPUs

Multiprocessor Symmetry

- In a multiprocessing system, all CPUs may be equal, or some may be reserved for special purposes.
- A combination of hardware and operating-system software design considerations determine the symmetry.

- Systems that treat all CPUs equally are called symmetric multiprocessing (SMP) systems.
- If all CPUs are not equal, system resources may be divided in a number of ways, including asymmetric multiprocessing (ASMP), non-uniform memory access (NUMA) multiprocessing, and clustered multiprocessing.

Instruction and Data Streams
Mike Flynn’s Taxonomy (1966)

Multiprocessors can be used in different ways:

- Uniprossesors (single-instruction, single-data or SISD)
- Within a single system to execute multiple, independent sequences of instructions in multiple contexts (multiple-instruction, multiple-data or MIMD);
- A single sequence of instructions in multiple contexts (single-instruction, multiple-data or SIMD, often used in vector processing);
- Multiple sequences of instructions in a single context (multiple-instruction, single-data or MISD, used for redundancy in fail-safe systems and sometimes applied to describe pipelined processors or hyper threading).

Instruction and Data Streams
Mike Flynn’s Taxonomy (1966)

Processor Coupling

Tightly-coupled multiprocessor systems:

- Contain multiple CPUs that are connected at the bus level.
- These CPUs may have access to a central shared memory (Symmetric Multiprocessing, or SMP), or may participate in a memory hierarchy with both local and shared memory (Non-Uniform Memory Access, or NUMA).
- Example: IBM p690 Regatta, Chip multiprocessors, also known as multi-core computing.

Loosely-coupled multiprocessor systems:

- Often referred as clusters
- Based on multiple standalone single or dual processor commodity computers interconnected via a high speed communication system, such as Gigabit ethernet.
- Example: Linux Beowulf cluster

Amdahl's law

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

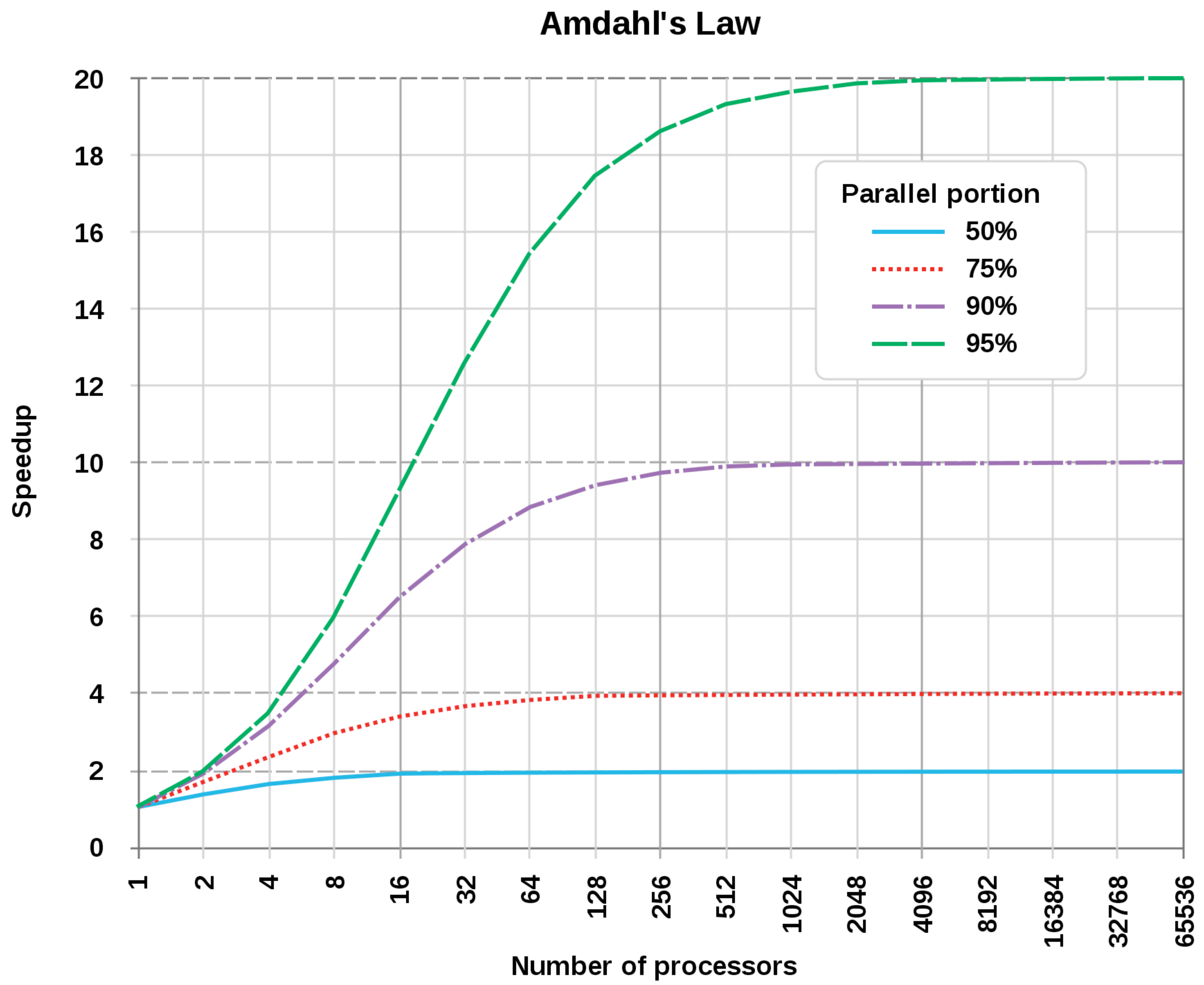
where

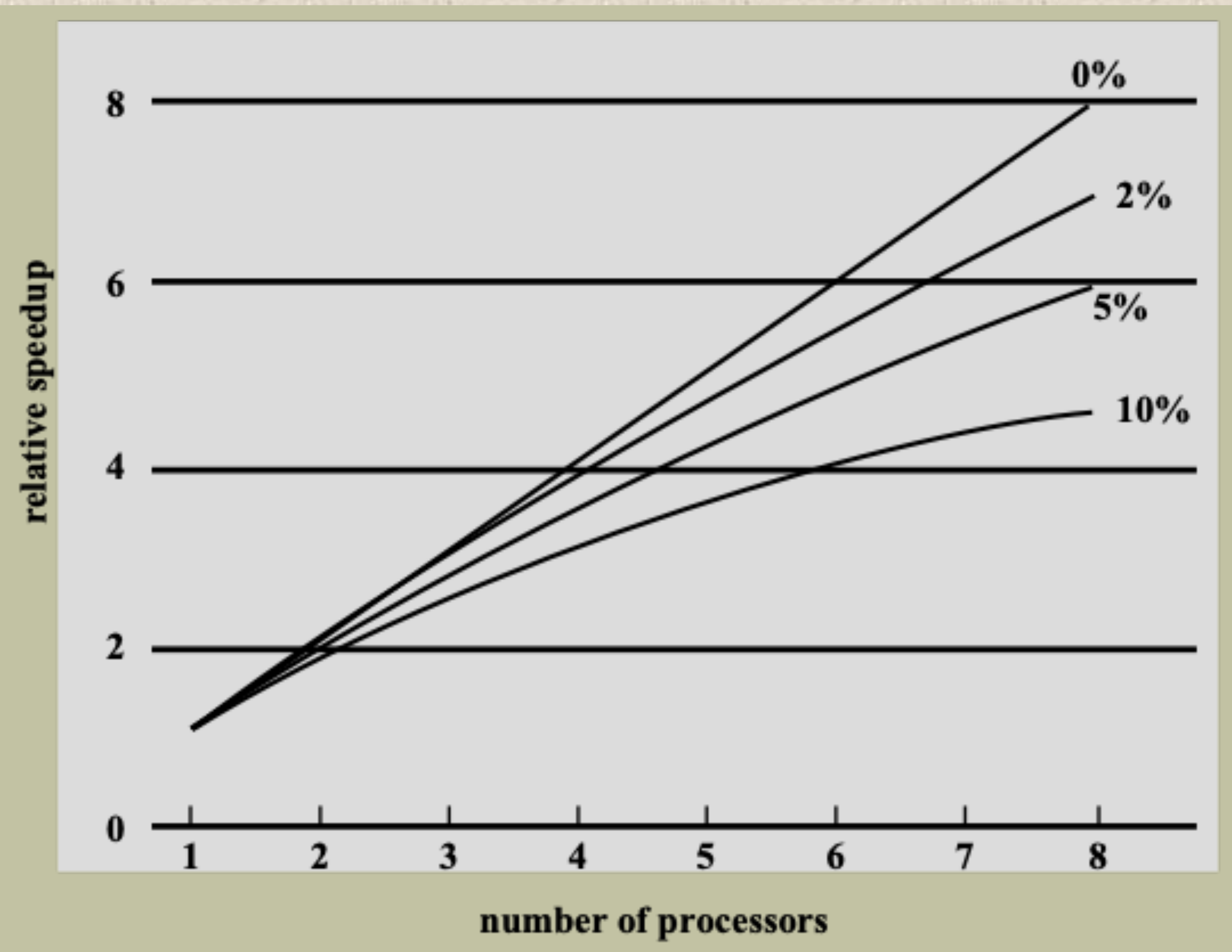
- S_{latency} is the theoretical speedup of the execution of the whole task;
- s is the speedup of the part of the task that benefits from improved system resources;
- p is the proportion of execution time that the part benefiting from improved resources originally occupied.

Furthermore,

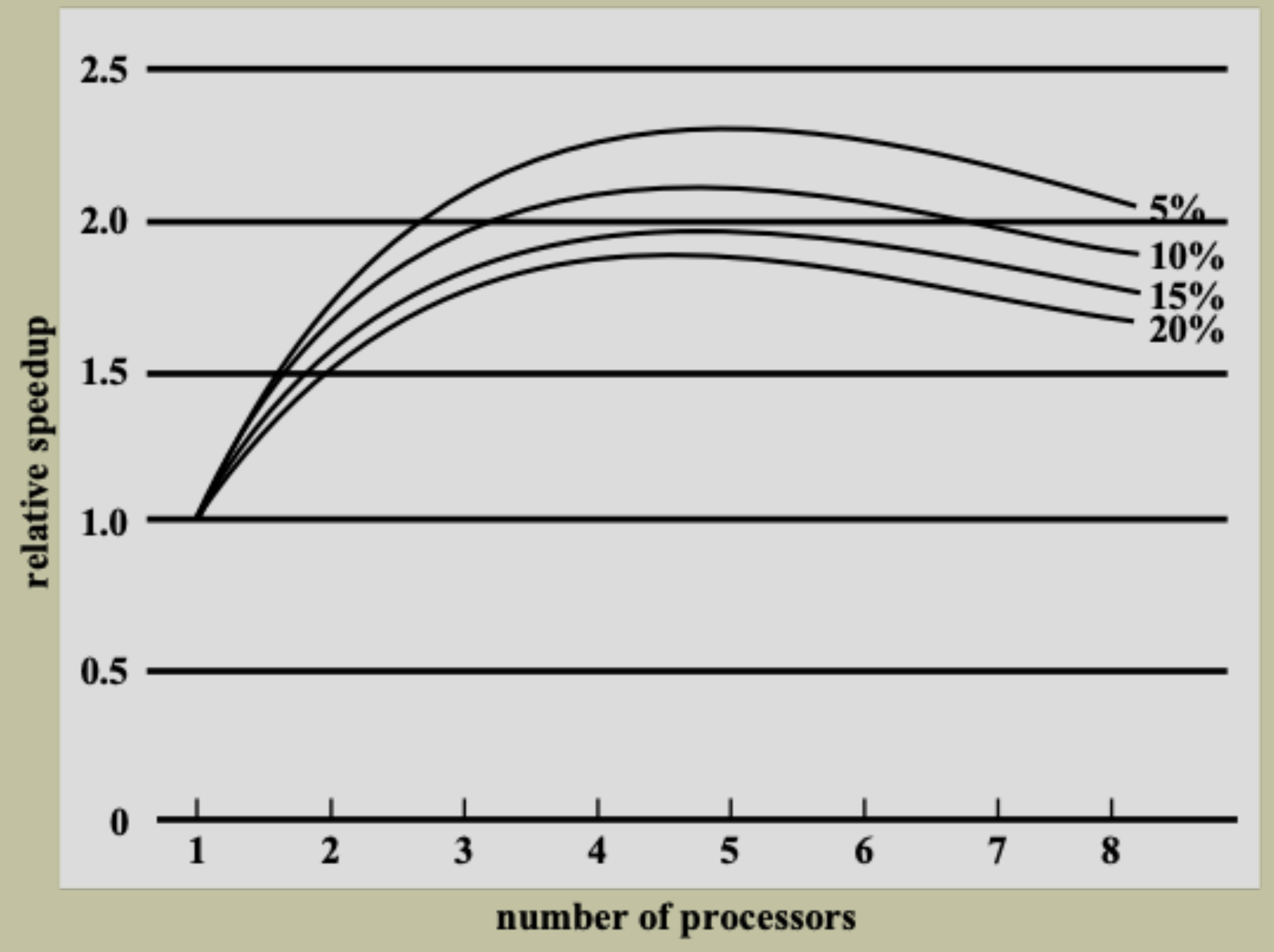
$$\begin{cases} S_{\text{latency}}(s) \leq \dfrac{1}{1-p} \\ \lim\limits_{s\to\infty} S_{\text{latency}}(s) = \dfrac{1}{1-p}. \end{cases}$$

shows that the theoretical speedup of the execution of the whole task increases with the improvement of the resources of the system and that regardless of the magnitude of the improvement, the theoretical speedup is always limited by the part of the task that cannot benefit from the improvement.


Amdahl's Law
Speedup
Number of processors
Parallel portion
50%
75%
90%
95%
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536



(a) Speedup with 0%, 2%, 5%, and 10% sequential portions



(b) Speedup with overheads

Figure 18.3 Performance Effect of Multiple Cores

Multiprocessor Communication Architectures

Message Passing

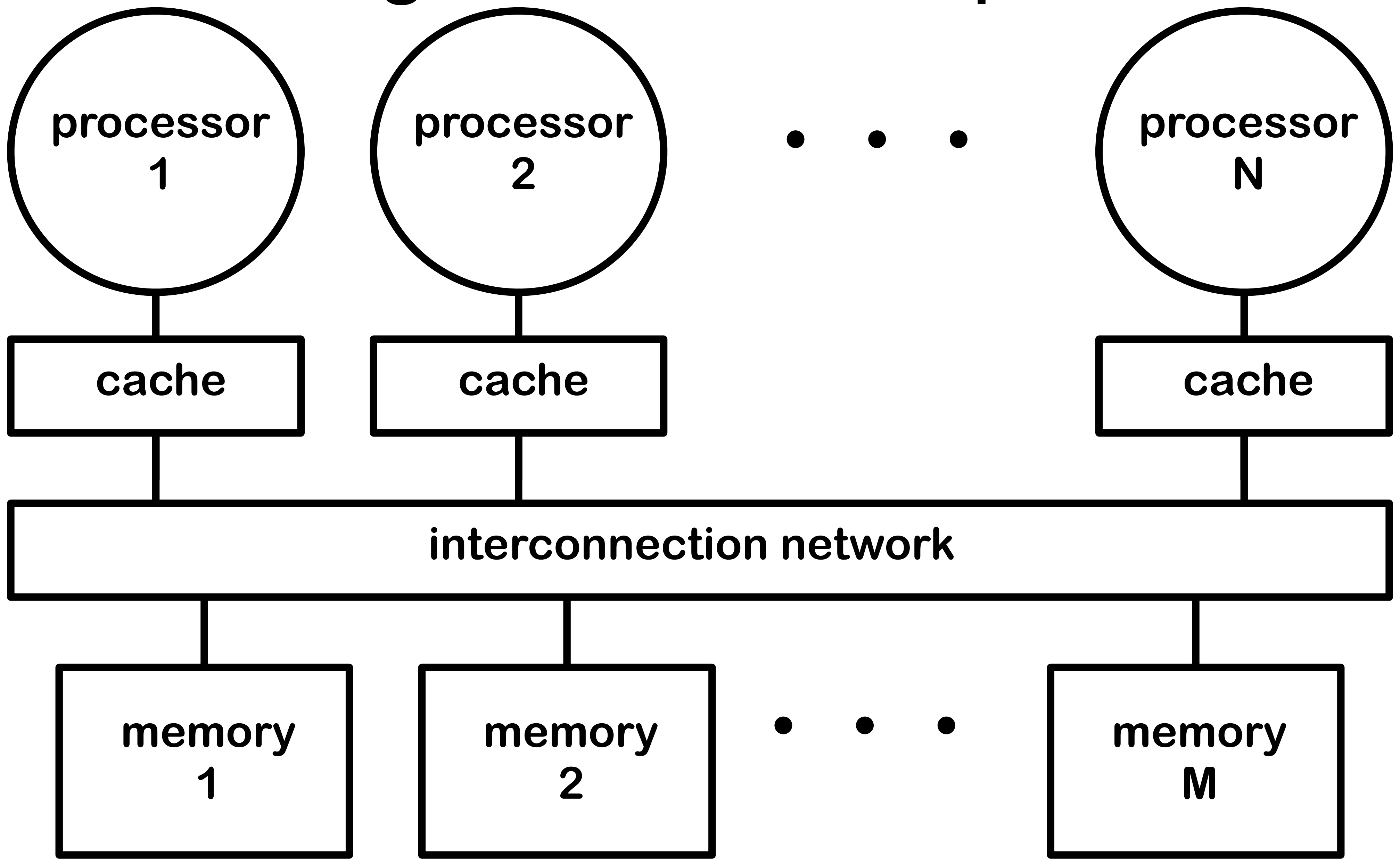
- Separate address space for each processor
- Processors communicate via message passing
- Processors have private memories
- Focuses attention on costly non-local operations

Shared Memory

- Processors communicate with shared address space
- Processors communicate by memory read/write
- Easy on small-scale machines
- Lower latency
- SMP or NUMA

Shared-Memory Processors

- Single copy of the OS (although some parts might be parallel)
- Relatively easy to program and port sequential code to
- Difficult to scale to large numbers of processors



UMA machine block diagram

Types of Shared-Memory Architectures

UMA

- Uniform Memory Access
- Access to all memory occurred at the same speed for all processors.

NUMA

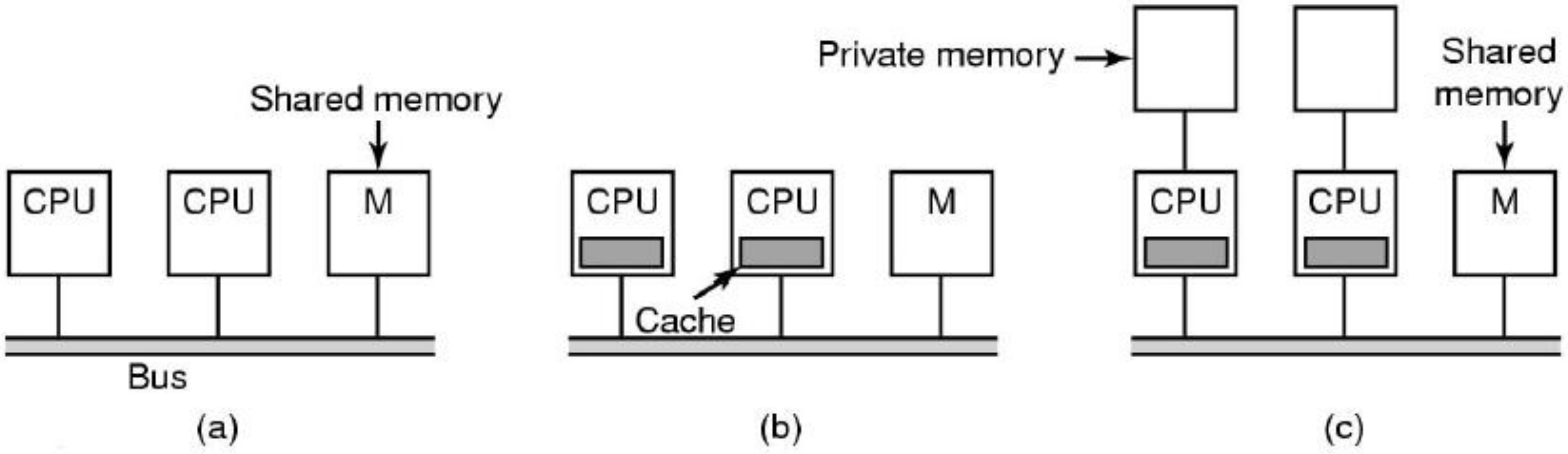
- Non-Uniform Memory Access
- a.k.a. “Distributed Shared Memory”.
- Typically interconnection is grid or hypercube.
- Access to some parts of memory is faster for some processors than other parts of memory.
- Harder to program, but scales to more processors

Bus Based UMA

(a) Simplest MP:
More than one processor on a single bus connect to memory, bus bandwidth becomes a bottleneck.

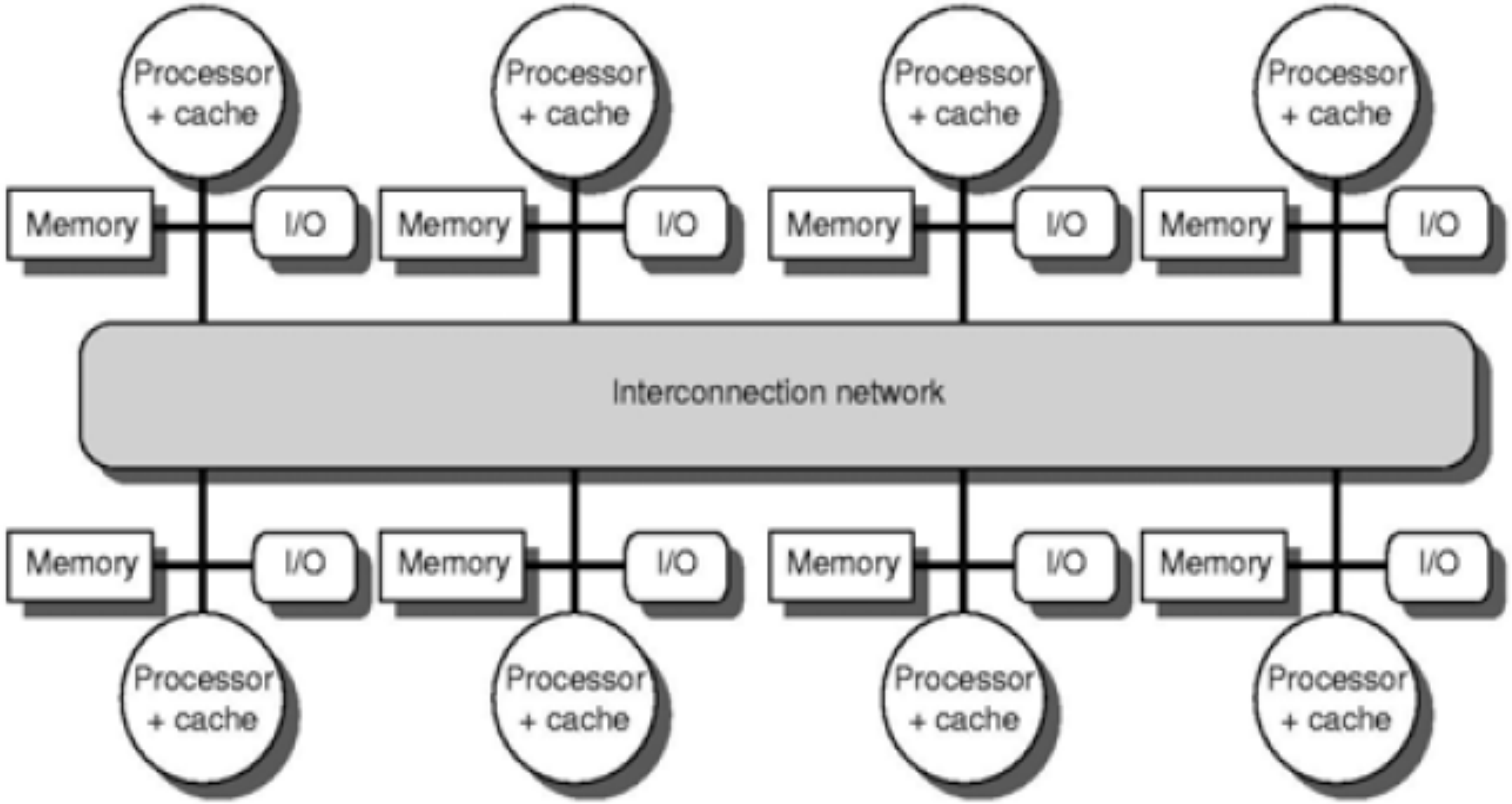
(b) Each processor has a cache to reduce the need to access to memory.

(c) To further scale the number of processors, each processor is given private local memory.



NUMA

- All memories can be addressed by all processors, but access to a processor's own local memory is faster than access to another processor's remote memory, i.e. each processor has a private connection to its own workspace, but a shared connection to all the others
- Looks like a distributed machine, but the interconnection network is usually custom-designed switches and/or buses.



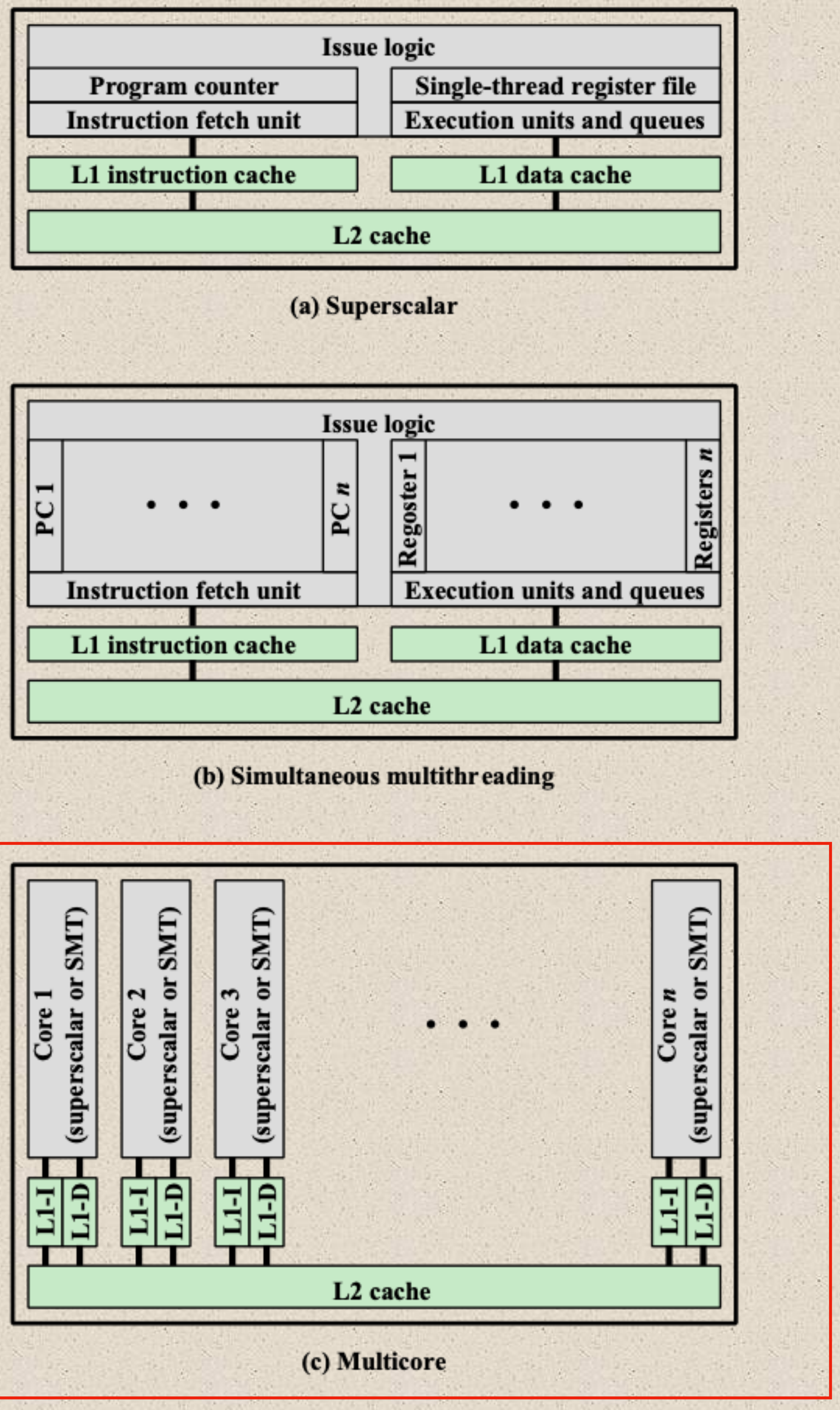
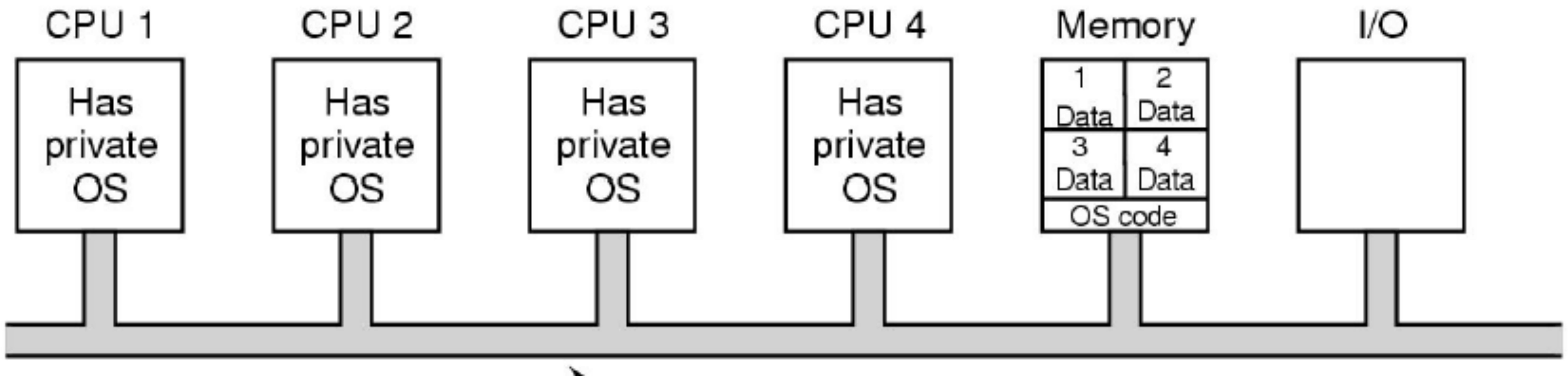


Figure 18.1 Alternative Chip Organizations

OS Option 1

Each CPU has its own OS

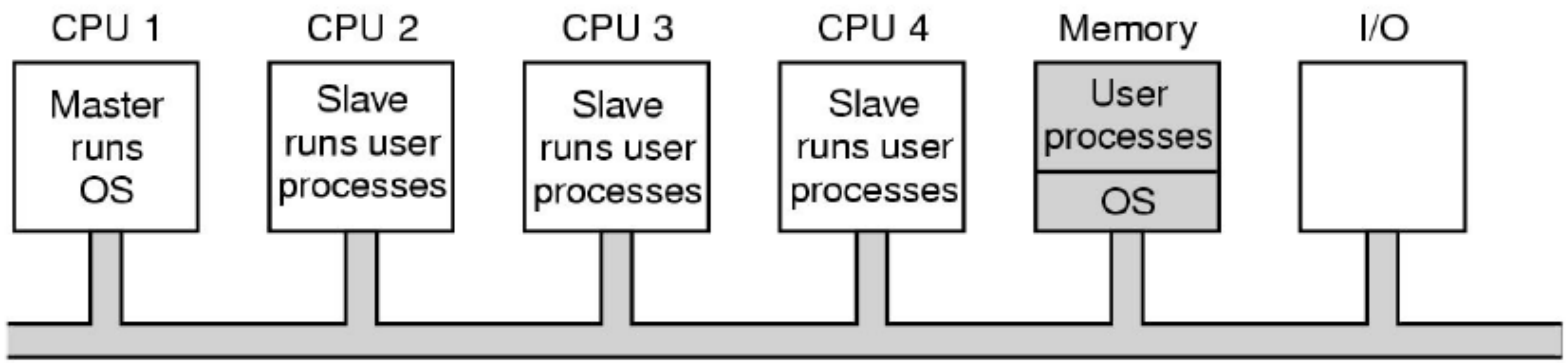
- Statically allocate physical memory to each CPU
- Each CPU runs its own independents OS
- Share peripherals
- Each CPU handles its processes system calls
- Used in early multiprocessor systems
- Simple to implement
- Avoids concurrency issues by not sharing
- Issues: 1. Each processor has its own scheduling queue.
  2. Each processor has its own memory partition.
  3. Consistency is an issue with independent disk buffer caches and potentially shared files.



OS Option 2

Master-Slave Multiprocessors

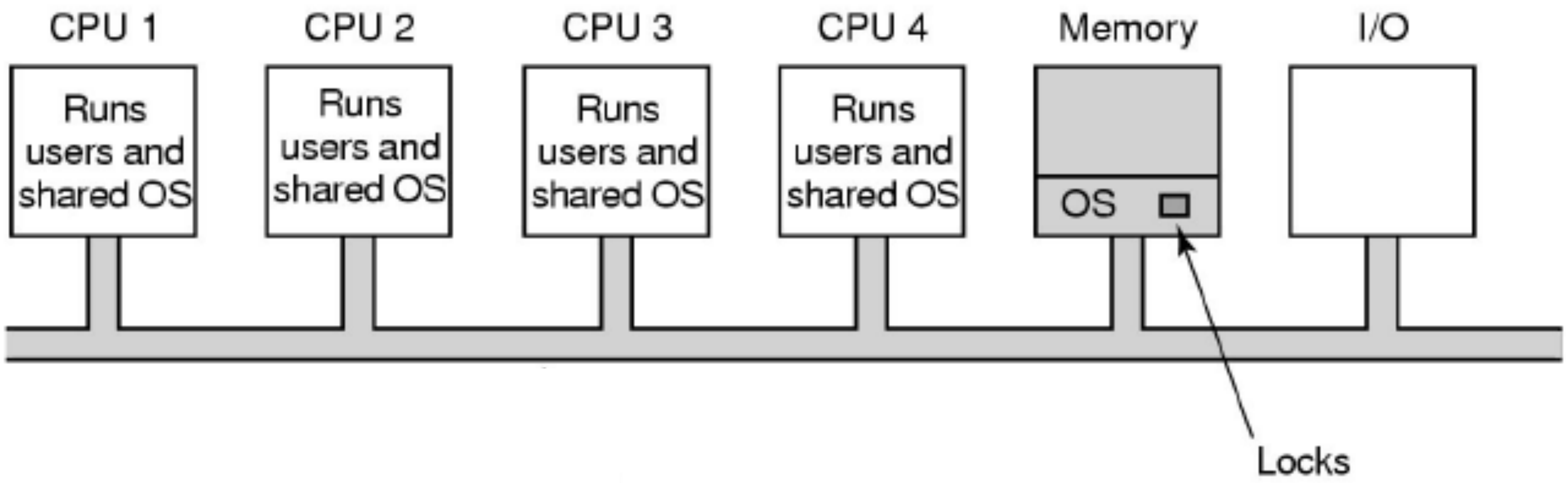
- OS mostly runs on a single fixed CPU.
- User-level applications run on the other CPUs.
- All system calls are passed to the Master CPU for processing
- Very little synchronisation required
- Single to implement
- Single centralised scheduler to keep all processors busy
- Memory can be allocated as needed to all CPUs.
- Issues: Master CPU becomes the bottleneck.



OS Option 3

Symmetric Multiprocessors (SMP)

- OS kernel runs on all processors, while load and resources are balanced between all processors.
- One alternative: A single mutex (mutual exclusion object) that make the entire kernel a large critical section; Only one CPU can be in the kernel at a time; Only slight better than master-slave
- Better alternative: Identify independent parts of the kernel and make each of them their own critical section, which allows parallelism in the kernel
- Issues: A difficult task; Code is mostly similar to uniprocessor code; hard part is identifying independent parts that don’t interfere with each other



Effective Applications for Multicore Processors

- **Multi-threaded native applications**
  - Thread-level parallelism
  - Characterized by having a small number of highly threaded processes
- **Multi-process applications**
  - Process-level parallelism
  - Characterized by the presence of many single-threaded processes
- **Java applications**
  - Embrace threading in a fundamental way
  - Java Virtual Machine is a multi-threaded process that provides scheduling and memory management for Java applications
- **Multi-instance applications**
  - If multiple application instances require some degree of isolation, virtualization technology can be used to provide each of them with its own separate and secure environment

Earlier Example
Quad-Processor Pentium Pro

- SMP, bus interconnection.
- 4 x 200 MHz Intel Pentium Pro processors.
- 8 + 8 Kb L1 cache per processor.
- 512 Kb L2 cache per processor.
- Snoopy cache coherence.
- Employed in Compaq, HP, IBM, NetPower.
- OS: Windows NT, Solaris, Linux, etc.

Example
HP Integrity Superdome

- The HP Integrity Superdome is HP's high-end addition to the family of industry-standard Itanium®-based solutions. The Superdome offers several configurations from 2-way multiprocessing all the way to 128 CPUs supporting multiple operating systems such as HP-UX 11iV2, Microsoft Windows Server 2003 Datacentre Edition, Linux and OpenVMS.
- **Processor:** 2 to 64 Intel Itanium 2 processors (1.6 GHz with 9 MB cache)
- **Memory:** Up to 1TB DDR memory
- 1-16 cell boards (each cell: 2 or 4 processors and 2 to 32 GB memory)
-
- 48 - 96 PCI-X internal hot-plug I/O card slots (Optional Server Expansion Unit)
- 4 -16 hardware partitions (nPars) using Server Expansion Unit

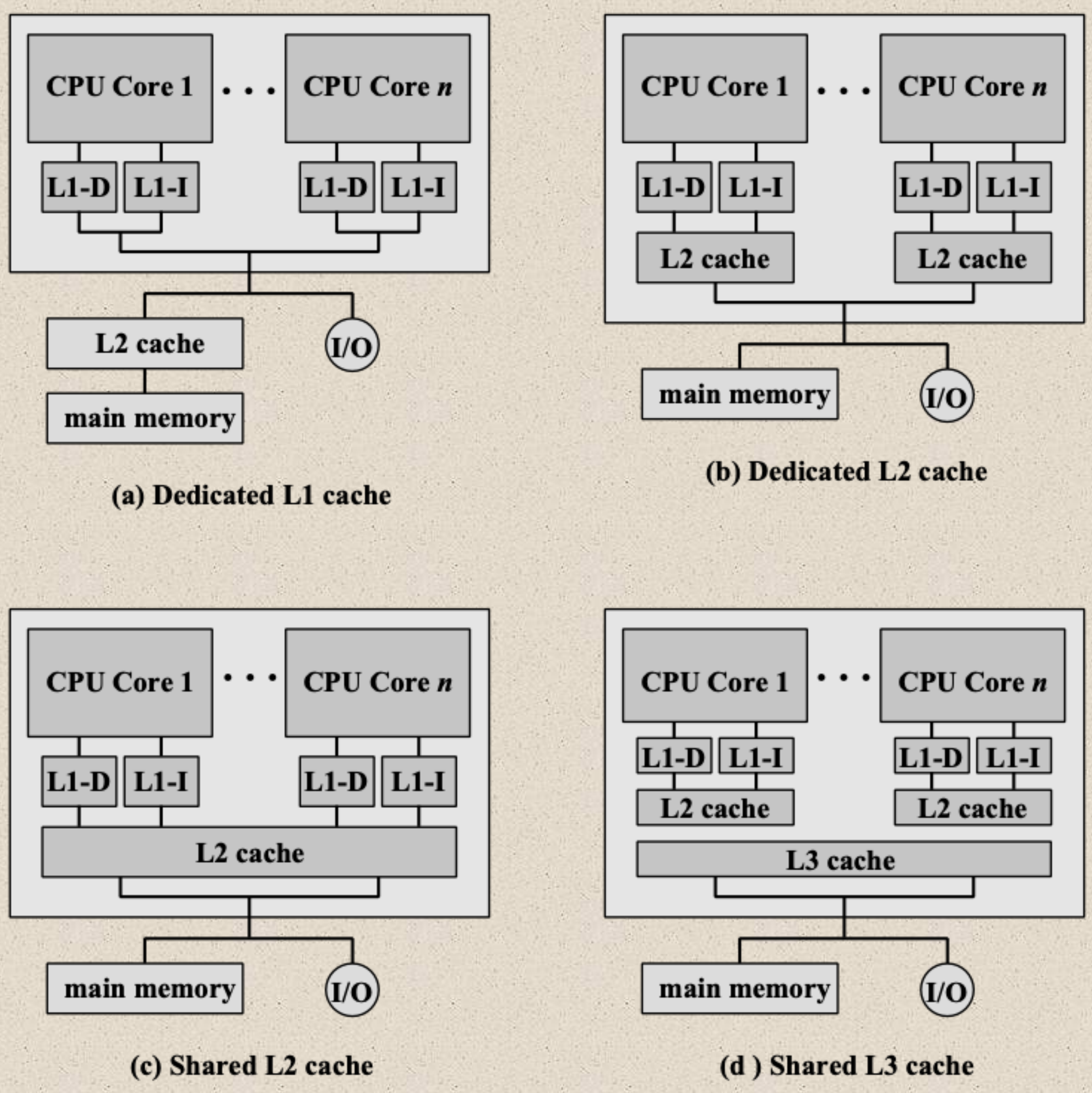


Figure 18.6 Multicore Organization Alternatives

Table 18.1

Operating Parameters of AMD 5100K Heterogeneous Multicore Processor

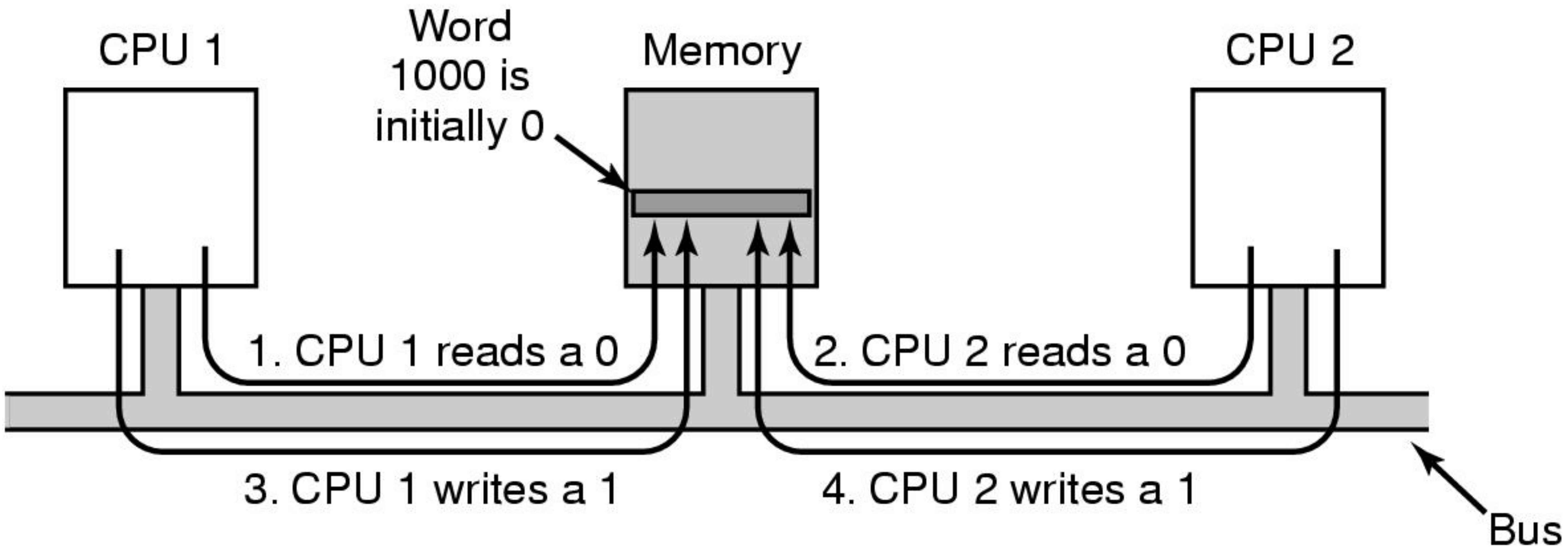
	CPU	GPU
Clock frequency (GHz)	3.8	0.8
Cores	4	384
FLOPS/core	8	2
GFLOPS	121.6	614.4

FLOPS = floating point operations per second
FLOPS/core = number of parallel floating point operations that can be performed

Conclusion

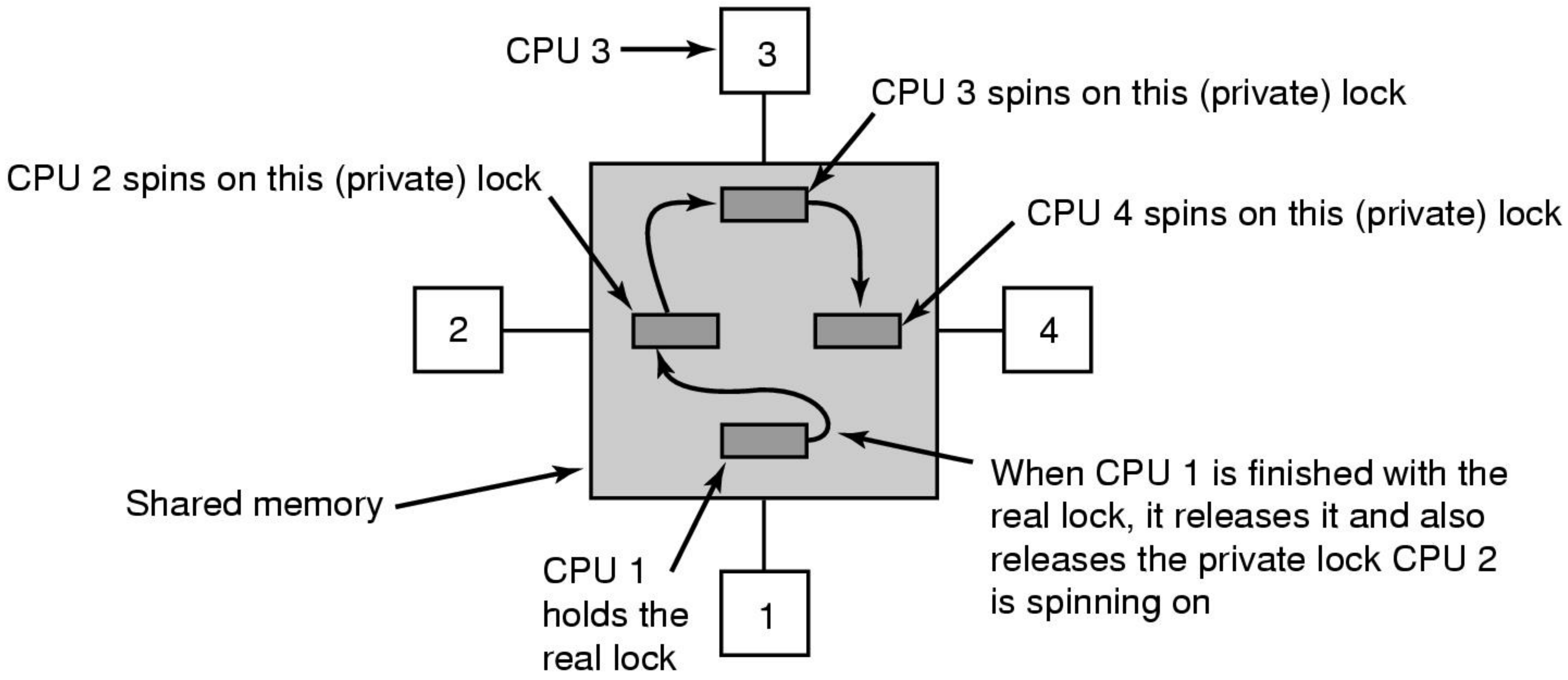
- Parallel processing is a technique for higher performance and effectiveness for multiprogrammed workloads.
- MPs combine the difficulties of building complex hardware systems and complex software systems.
- Communication, memory, affinity and throughputs presents an important influence on the systems costs and performances
- On-chip MPs (MPSoC) technology is dominating and growing

Multiprocessor Synchronization (1)



TSL instruction can fail if bus already locked

Multiprocessor Synchronization (2)



Multiple locks used to avoid cache thrashing

Multiprocessor Synchronization (3)

Busy Spinning is **a wait strategy in which one thread waits for some condition to happen which is to be set by some other thread**. ... The consumer thread while waiting holds the CPU cycles and thus there is wastage of CPU resources which can be used for some other processing by other threads.

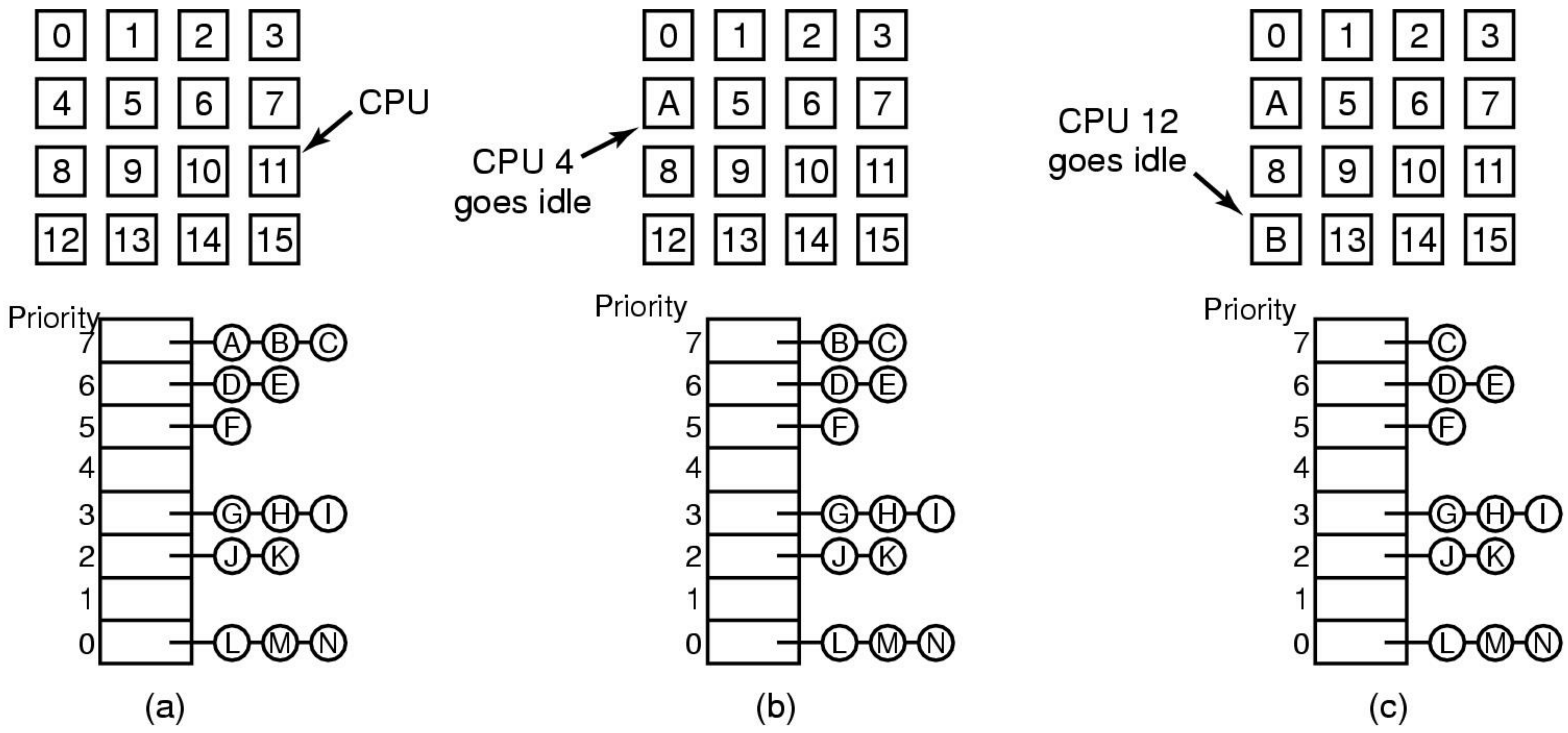
Blocked waiting (also known as sleeping waiting) is **a wait strategy** where a task sleeps until an event occurs. For blocked waiting to work, there must be some external agent that can wake up the task when the event has (or may have) occurred.

Multiprocessor Synchronization (4)

Spinning versus Switching

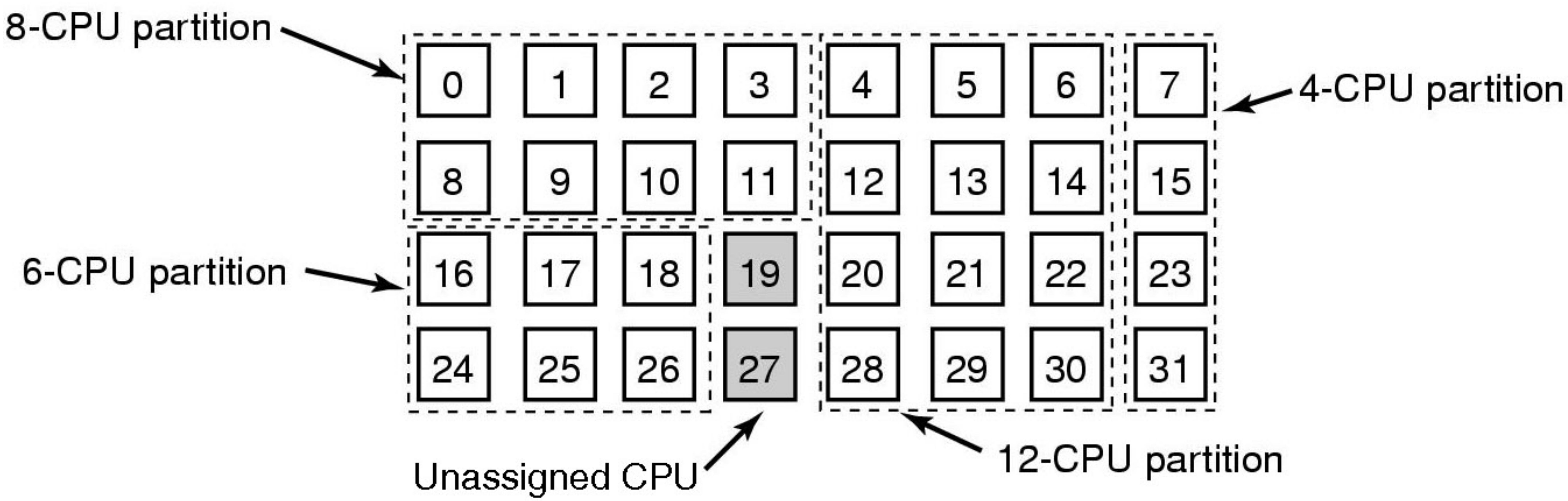
- In some cases CPU must wait
  - waits to acquire ready list
- In other cases a choice exists
  - spinning wastes CPU cycles
  - switching uses up CPU cycles also
  - possible to make separate decision each time locked mutex encountered

Multiprocessor Scheduling (1)



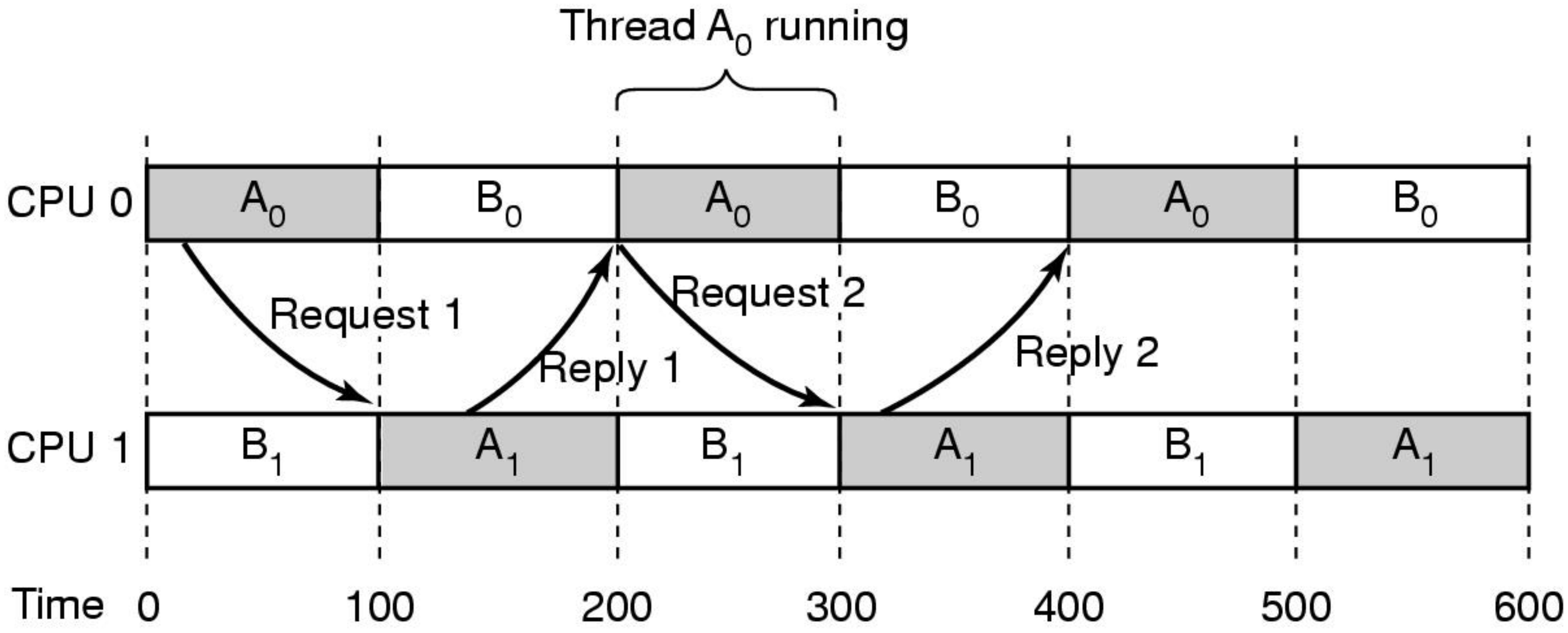
- Timesharing
  - note use of single data structure for scheduling

Multiprocessor Scheduling (2)



- Space sharing
  - multiple threads at same time across multiple CPUs

Multiprocessor Scheduling (3)



- Problem with communication between two threads
  - both belong to process A
  - both running out of phase

Multiprocessor Scheduling (4)

- **Solution: Gang Scheduling**
  1. Groups of related threads scheduled as a unit (a gang)
  2. All members of gang run simultaneously
     - on different timeshared CPUs
  3. All gang members start and end time slices together

Multiprocessor Scheduling (5)

CPU

Time slot	0	1	2	3	4	5
0	A_0	A_1	A_2	A_3	A_4	A_5
1	B_0	B_1	B_2	C_0	C_1	C_2
2	D_0	D_1	D_2	D_3	D_4	E_0
3	E_1	E_2	E_3	E_4	E_5	E_6
4	A_0	A_1	A_2	A_3	A_4	A_5
5	B_0	B_1	B_2	C_0	C_1	C_2
6	D_0	D_1	D_2	D_3	D_4	E_0
7	E_1	E_2	E_3	E_4	E_5	E_6

Gang Scheduling

Multicomputers

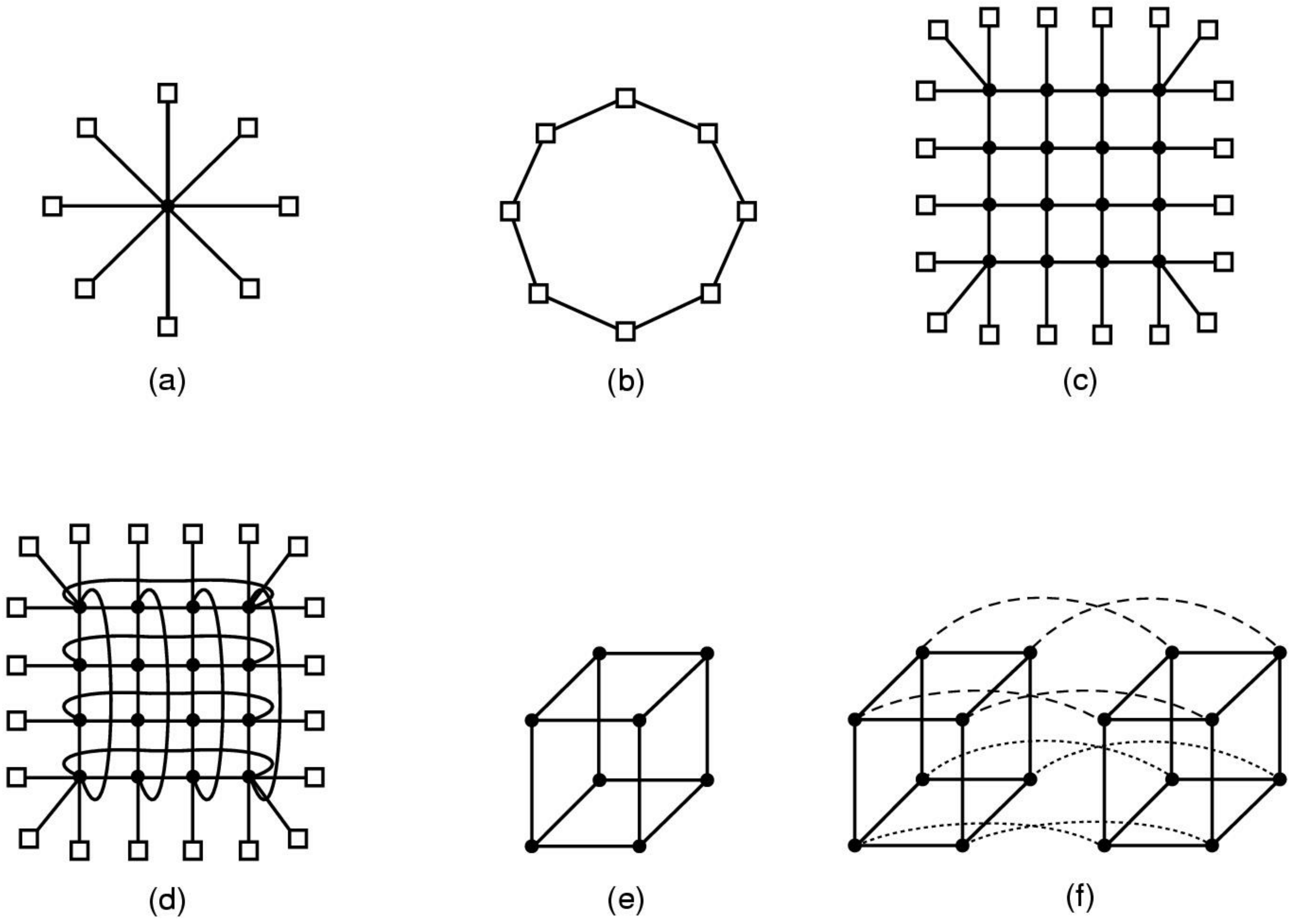
- Definition:
  Tightly-coupled CPUs that do not share memory

- Also known as
  - cluster computers
  - clusters of workstations (COWs)

https://www.youtube.com/watch?v=So9SR3qpWsM

10:00 -

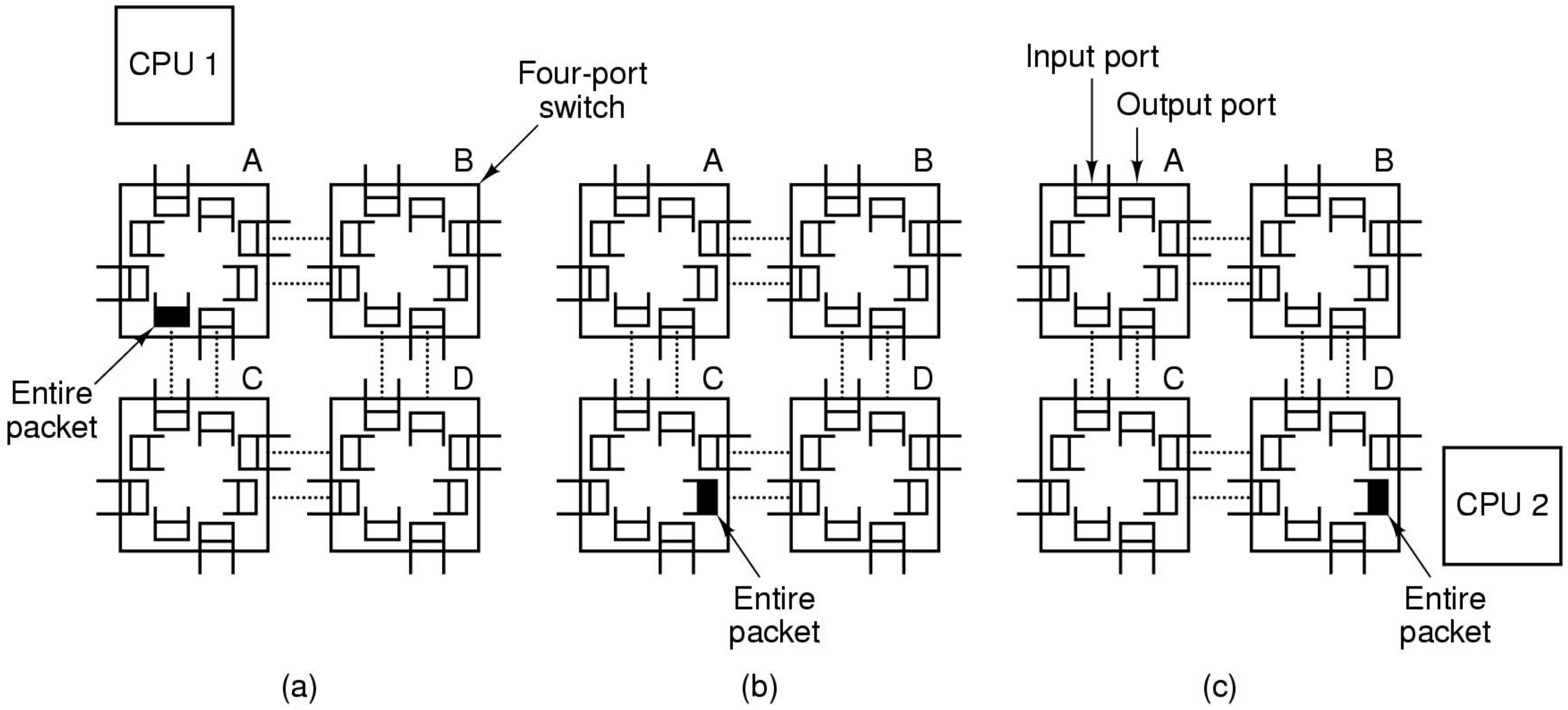
Multicomputer Hardware (1)



- Interconnection topologies

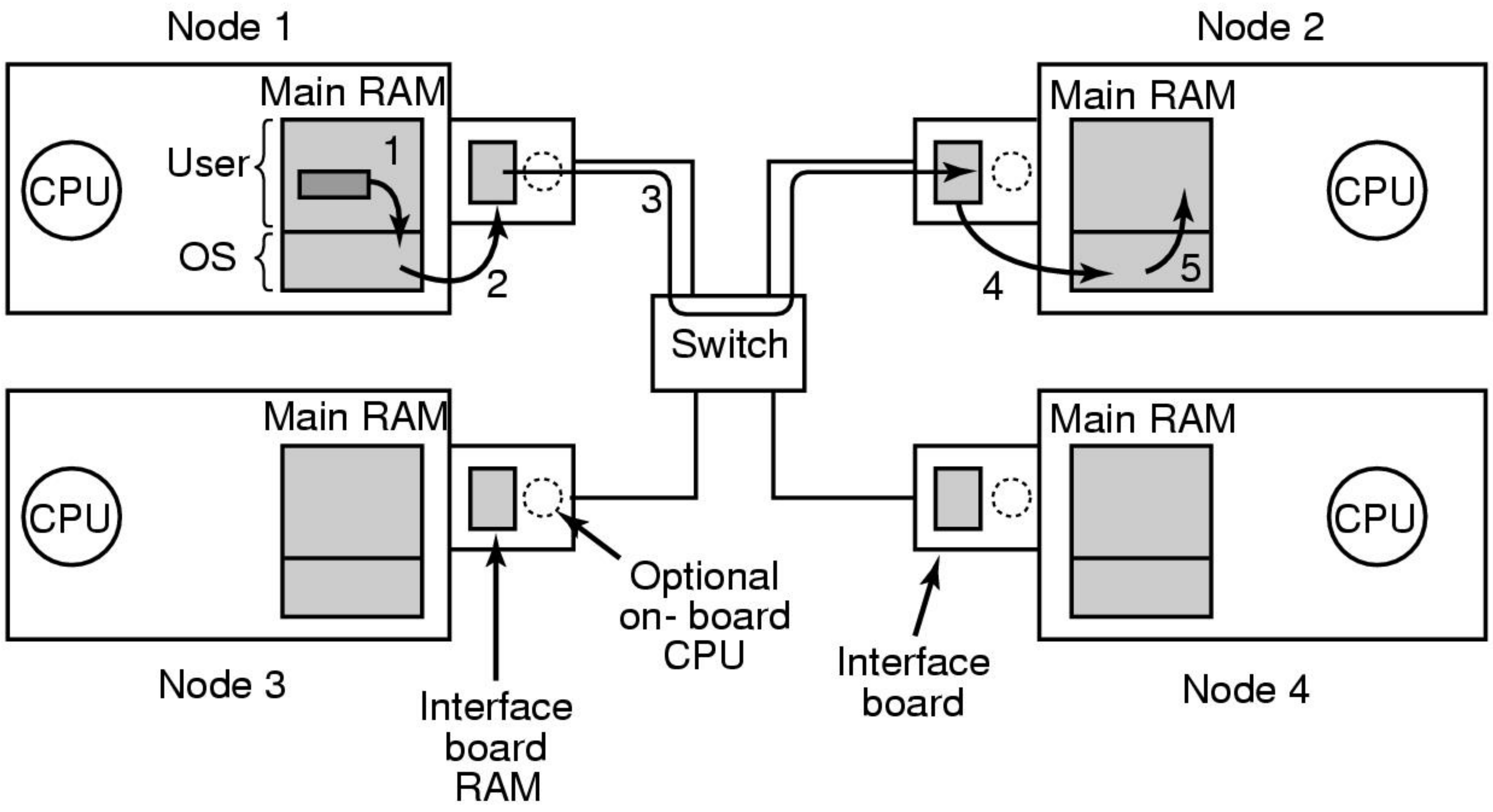
(a) single switch
(b) ring
(c) grid
(d) double torus
(e) cube
(f) hypercube

Multicomputer Hardware (2)



- Switching scheme
  - store-and-forward packet switching

Multicomputer Hardware (3)



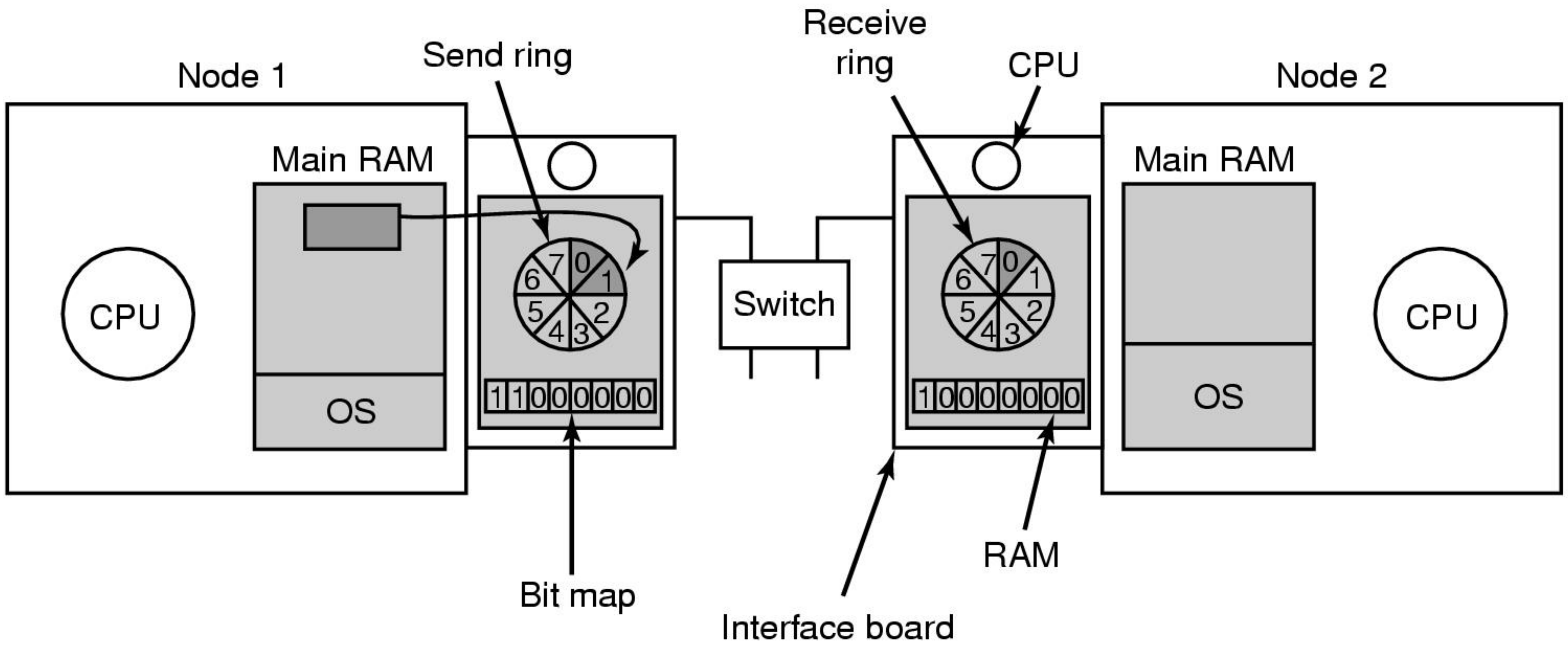
Network interface boards in a multicomputer

Low-Level Communication Software (1)

- If several processes running on node
  - need network access to send packets …
- Map interface board to all process that need it

- If kernel needs access to network …
- Use two network boards
  - one to user space, one to kernel

Low-Level Communication Software (2)



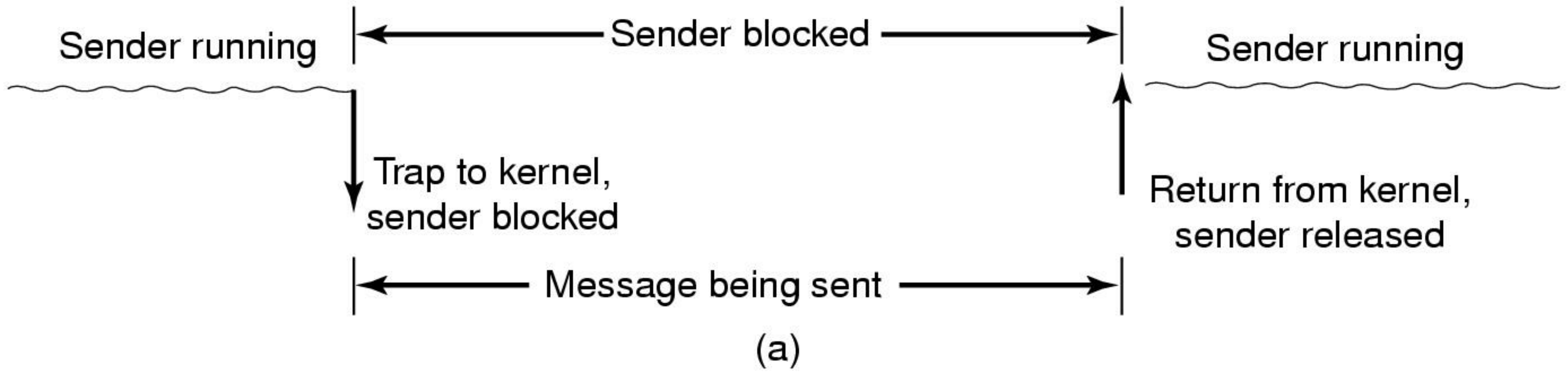
Node to Network Interface Communication

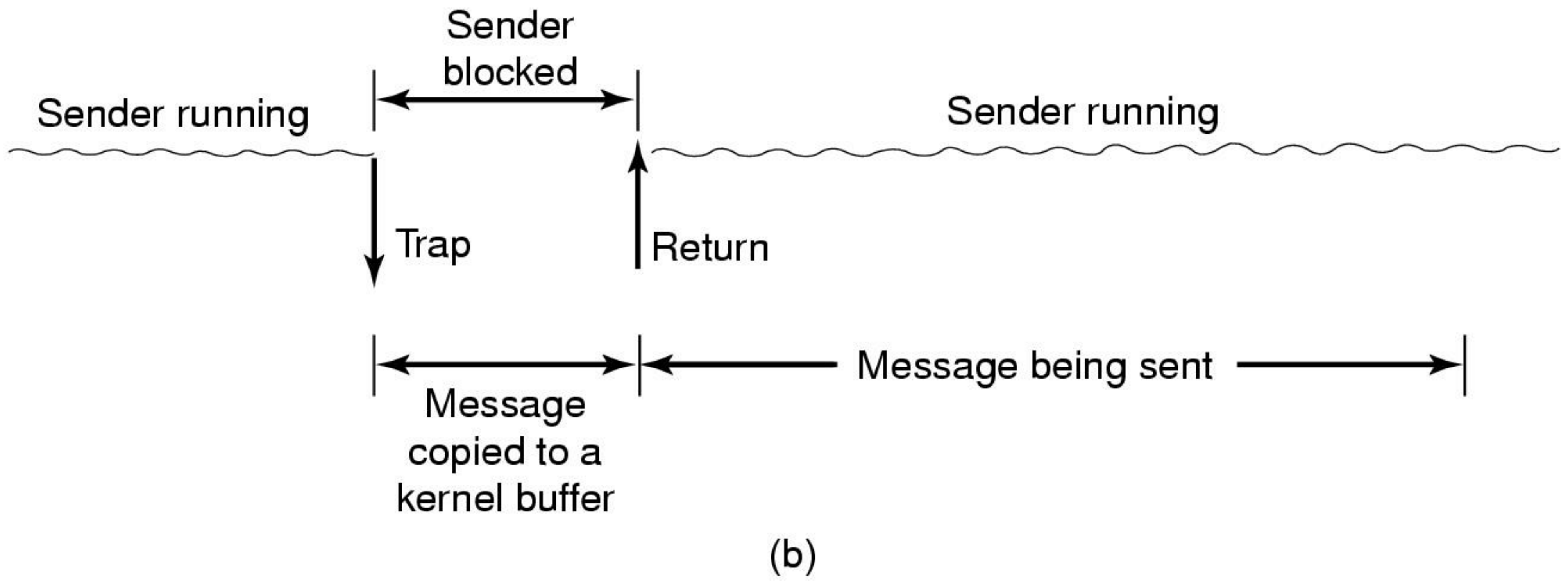
- Use send & receive rings
- coordinates main CPU with on-board CPU

User Level Communication Software

- Minimum services provided
  - send and receive commands
- These are blocking (synchronous) calls

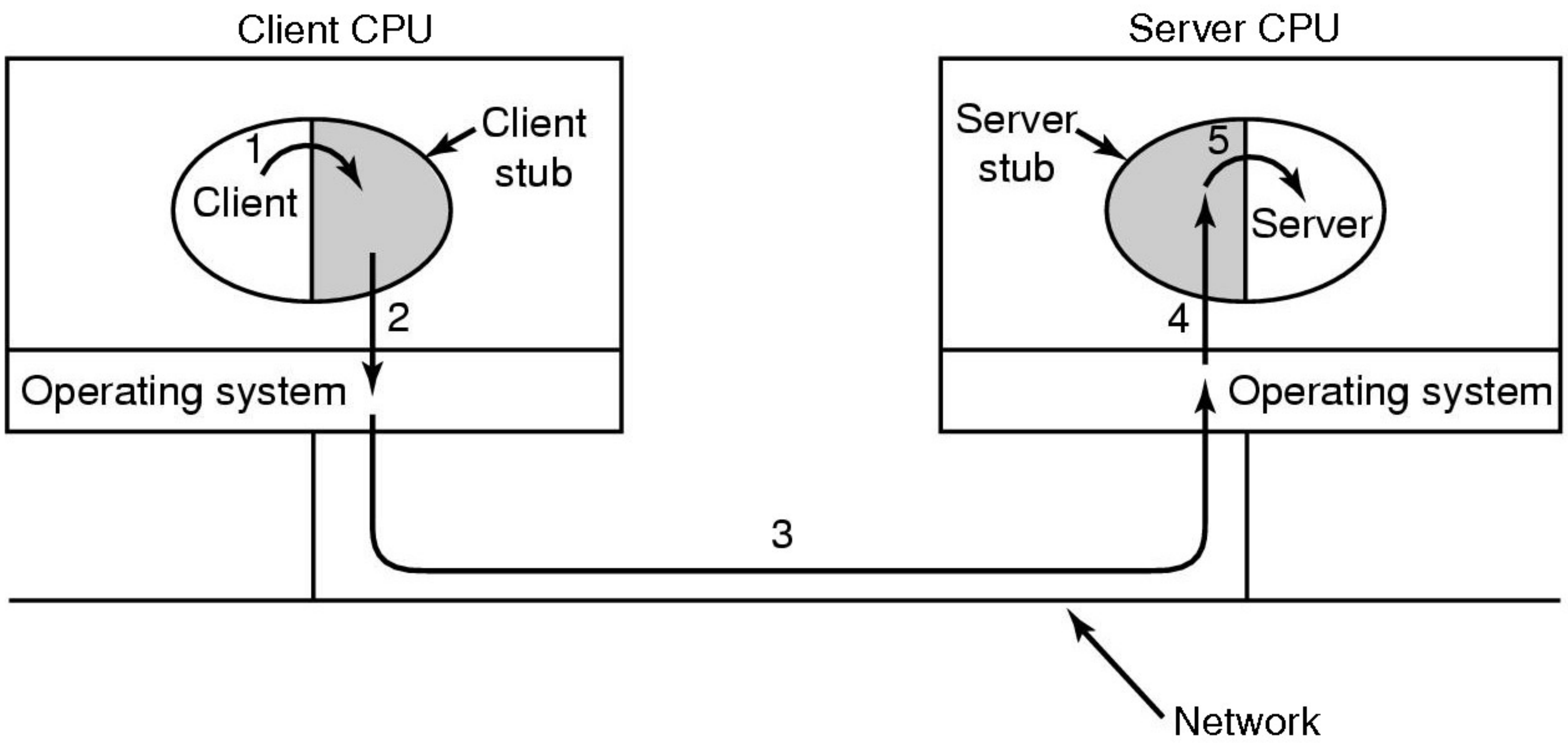
(a) Blocking send call





(b) Nonblocking send call

Remote Procedure Call (1)



- Steps in making a remote procedure call
  - the stubs are shaded gray

Remote Procedure Call (1a)

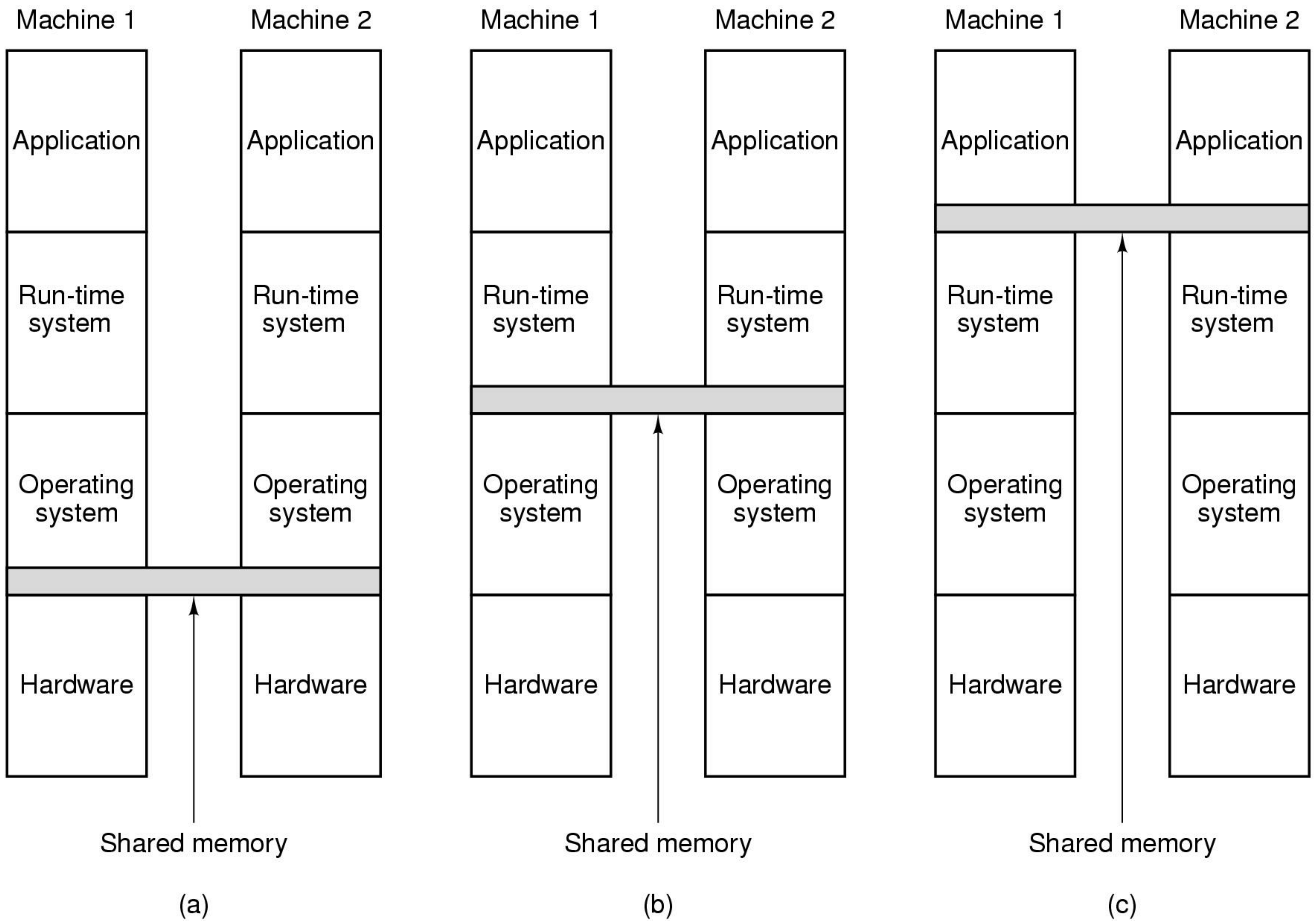
A client stub is **responsible for conversion (marshalling) of parameters used in a function call** and deconversion of results passed from the server after execution of the function. It uses an interface description language (IDL) to define the interface between client and server.

Remote Procedure Call (2)

Implementation Issues

- Cannot pass pointers
  - call by reference becomes copy-restore (but might fail)
- Weakly typed languages
  - client stub cannot determine size
- Not always possible to determine parameter types
- Cannot use global variables
  - may get moved to remote machine

Distributed Shared Memory (1)



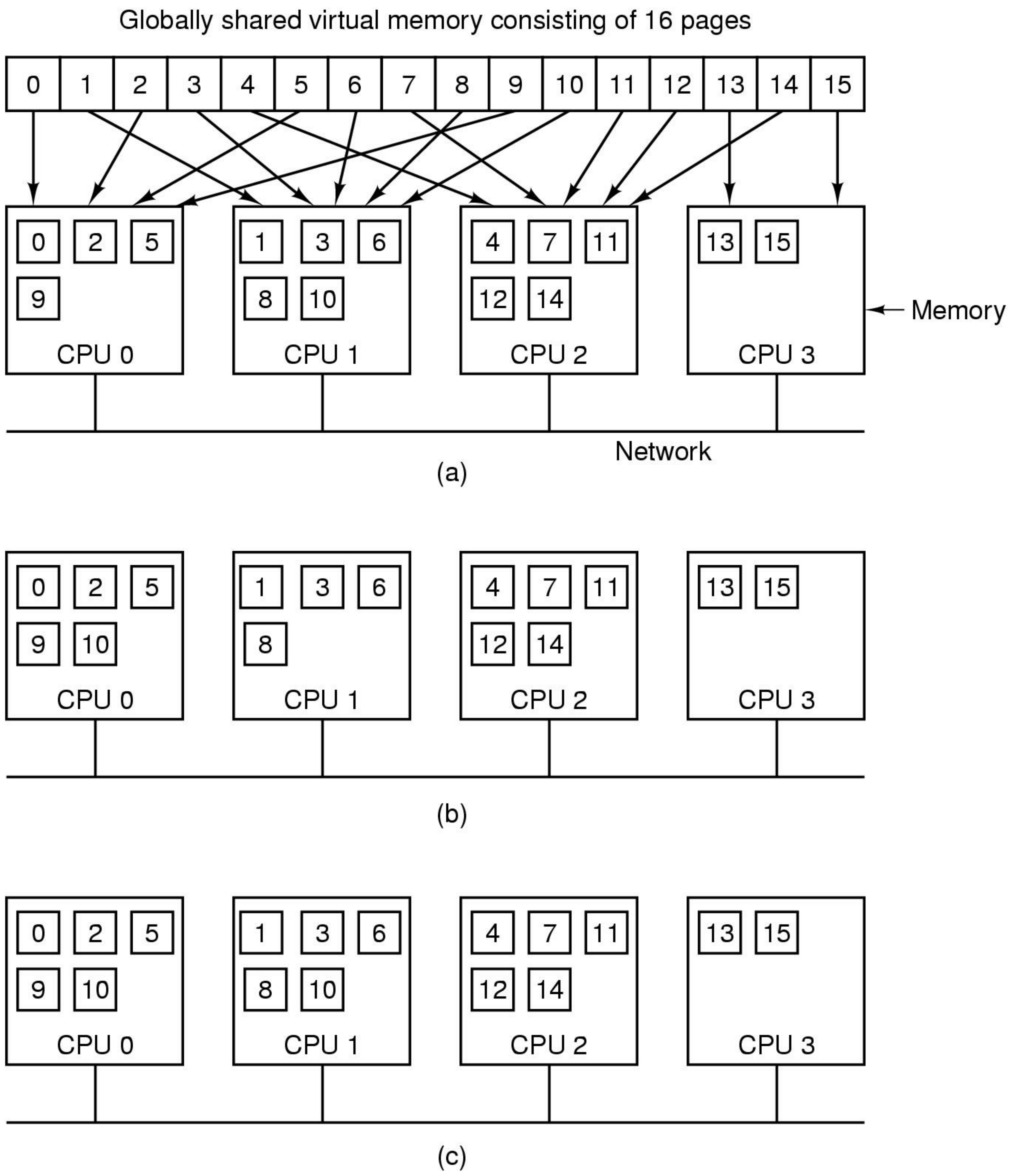
- Note layers where it can be implemented
  - hardware
  - operating system

Distributed Shared Memory (2)

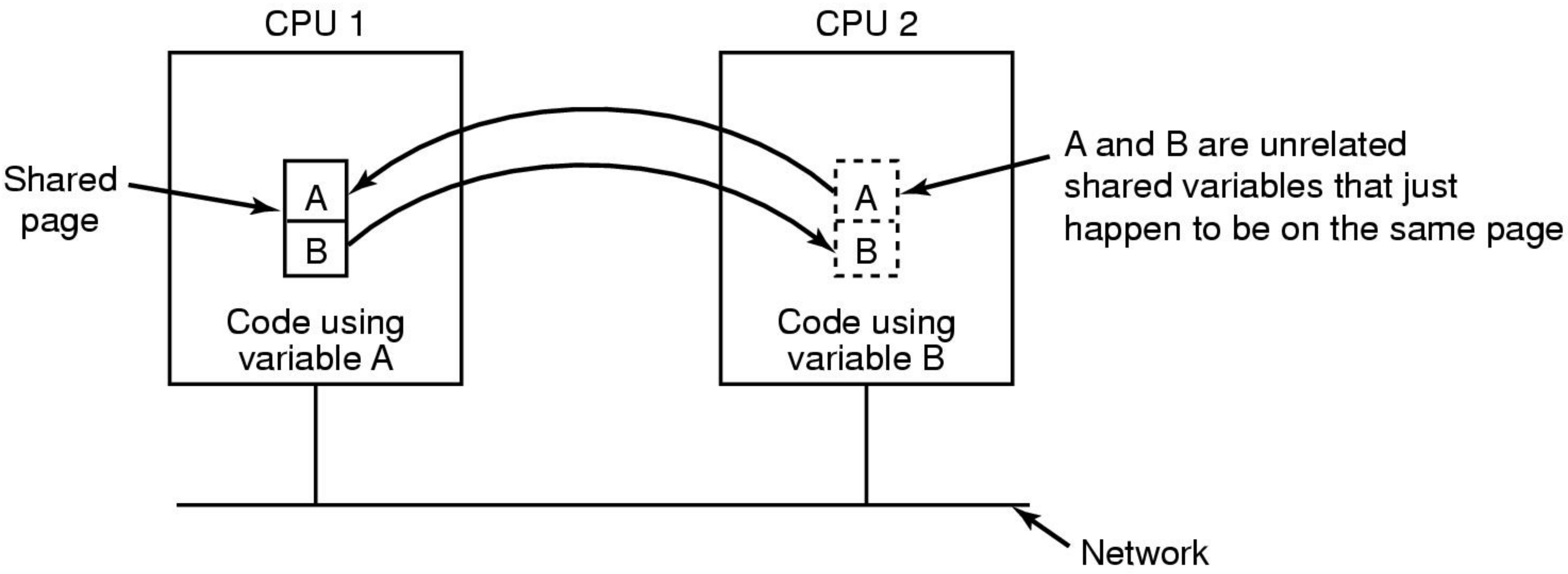
Replication

(a) Pages distributed on 4 machines

(b) CPU 0 reads page 10

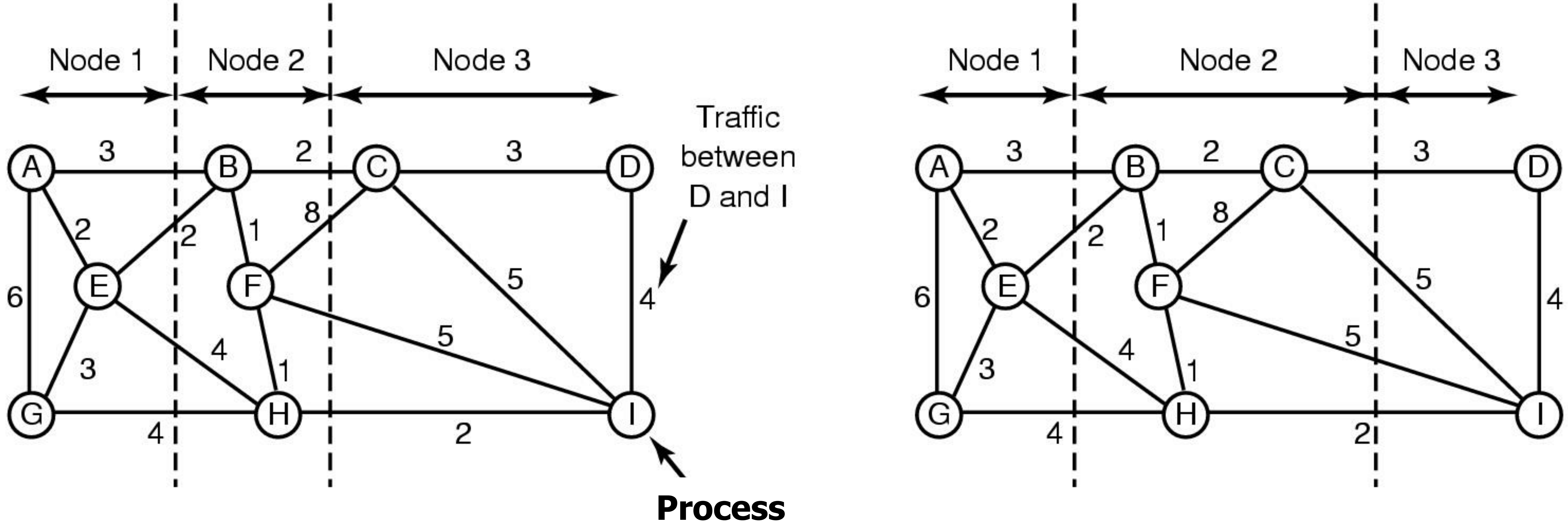


Distributed Shared Memory (3)



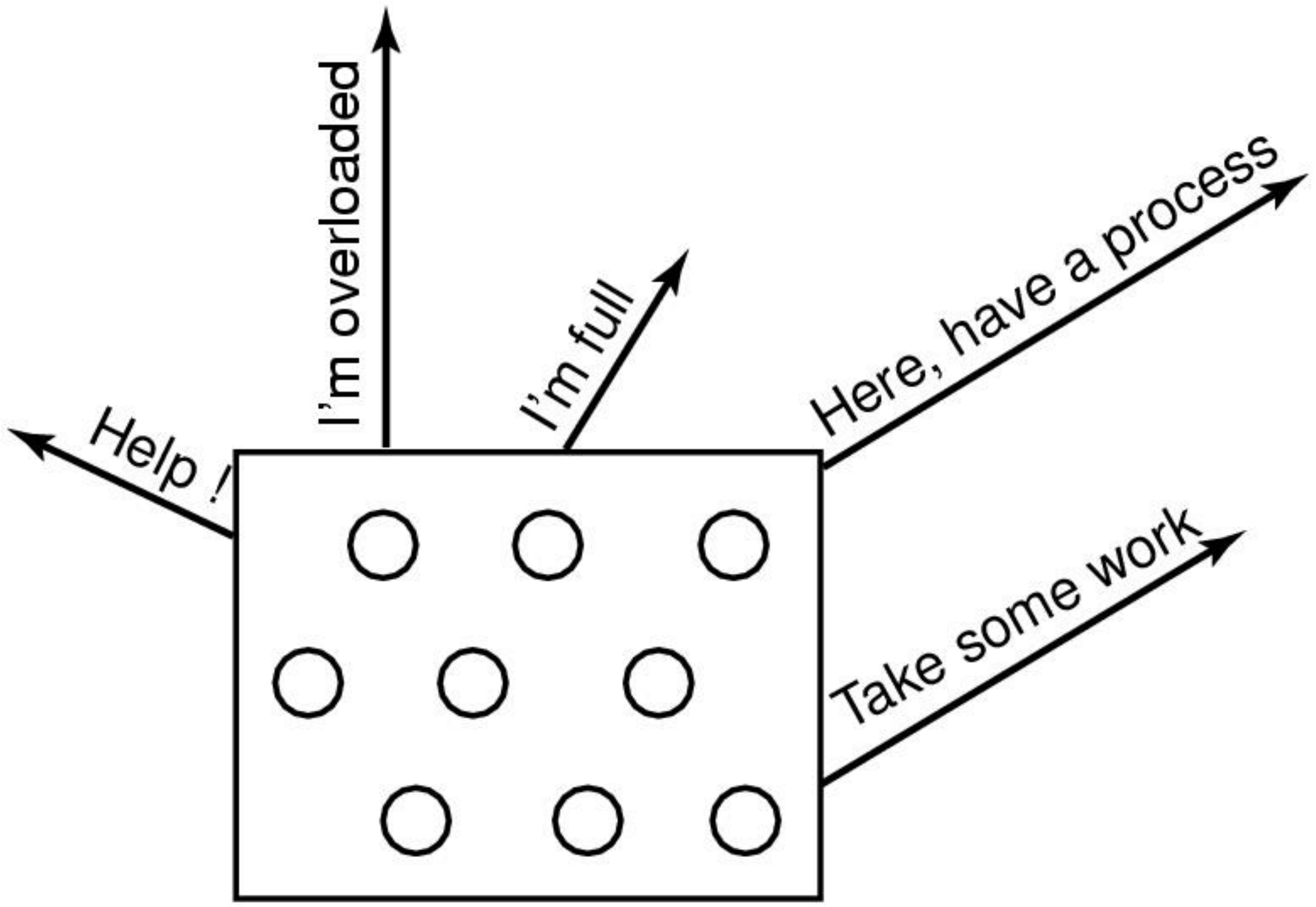
- False Sharing
- Must also achieve sequential consistency

Multicomputer Scheduling
Load Balancing (1)



- Graph-theoretic deterministic algorithm

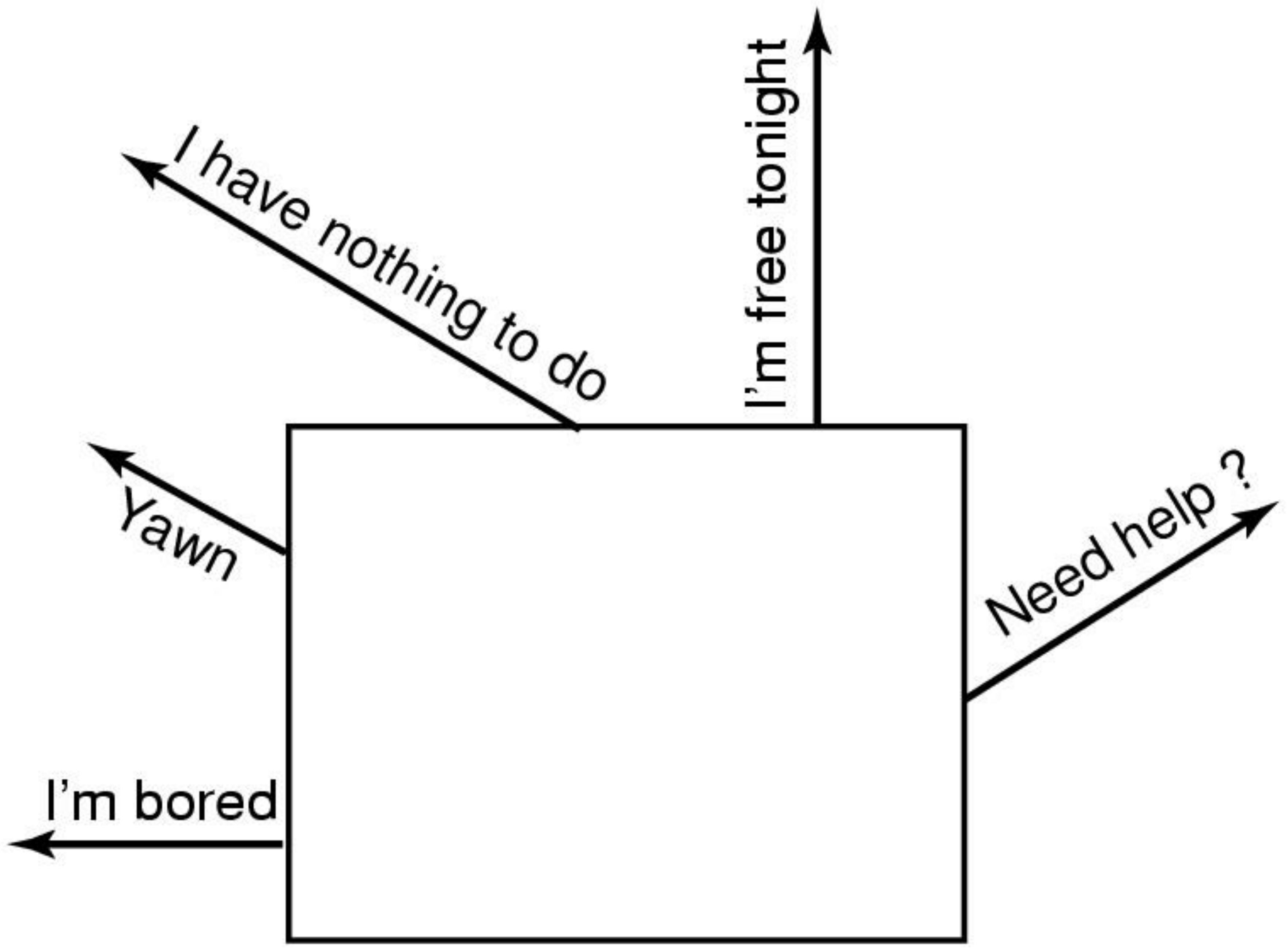
Load Balancing (2)



(a)

- Sender-initiated distributed heuristic algorithm
  - overloaded sender

Load Balancing (3)



(b)

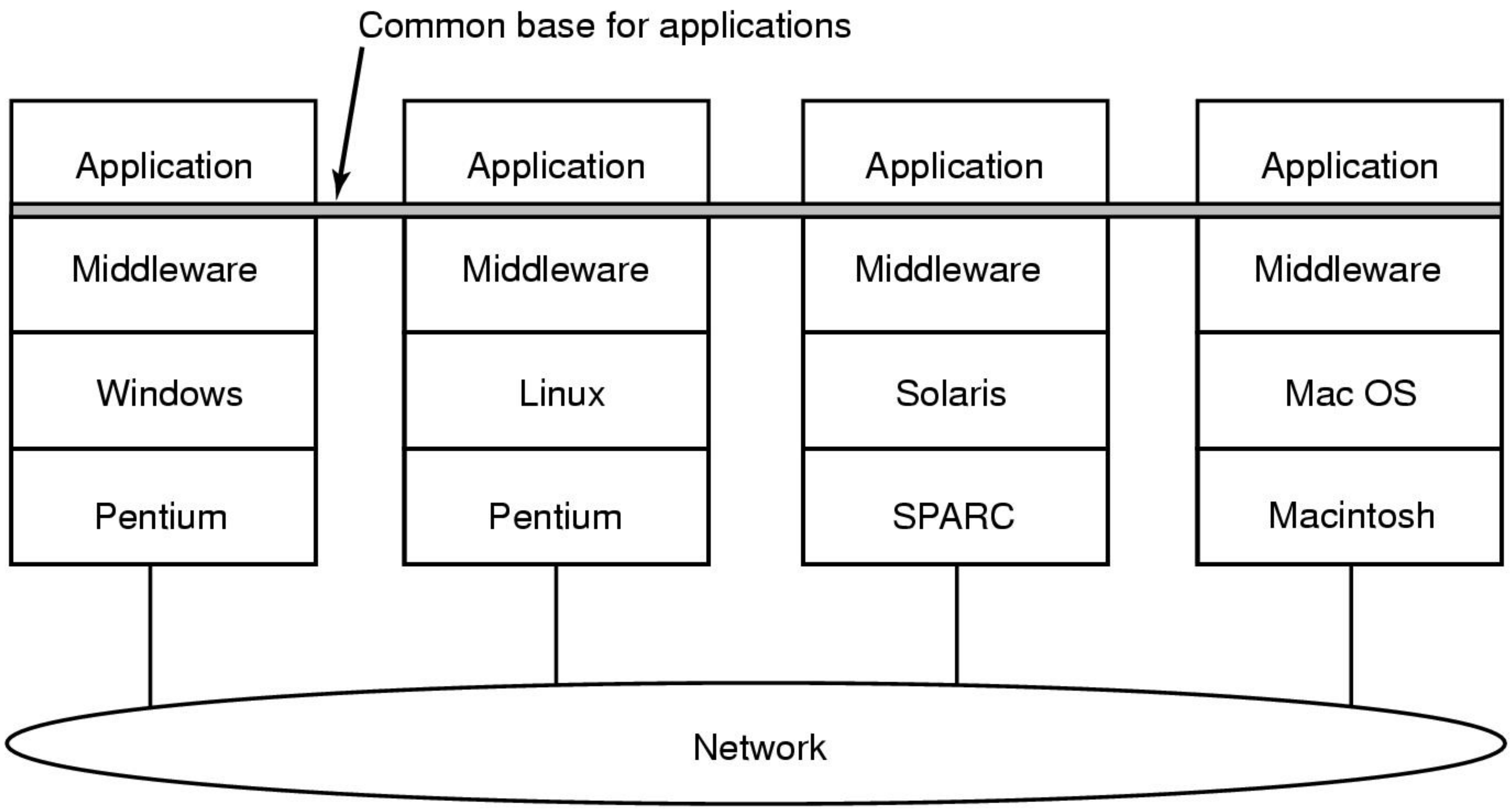
- Receiver-initiated distributed heuristic algorithm
  - under loaded receiver

Distributed Systems (1)

Item	Multiprocessor	Multicomputer	Distributed System
Node configuration	CPU	CPU, RAM, net interface	Complete computer
Node peripherals	All shared	Shared exc. maybe disk	Full set per node
Location	Same rack	Same room	Possibly worldwide
Internode communication	Shared RAM	Dedicated interconnect	Traditional network
Operating systems	One, shared	Multiple, same	Possibly all different
File systems	One, shared	One, shared	Each node has own
Administration	One organization	One organization	Many organizations

Comparison of three kinds of multiple CPU systems

Distributed Systems (2)



Achieving uniformity with middleware

Multiprocessors Review

https://www.youtube.com/watch?v=TIcmpXjt2vE

1:57