MODERN OPERATING SYSTEMS

Third Edition

ANDREW S. TANENBAUM

Chapter 8
Multiple Processor Systems

Multiple Processor Systems

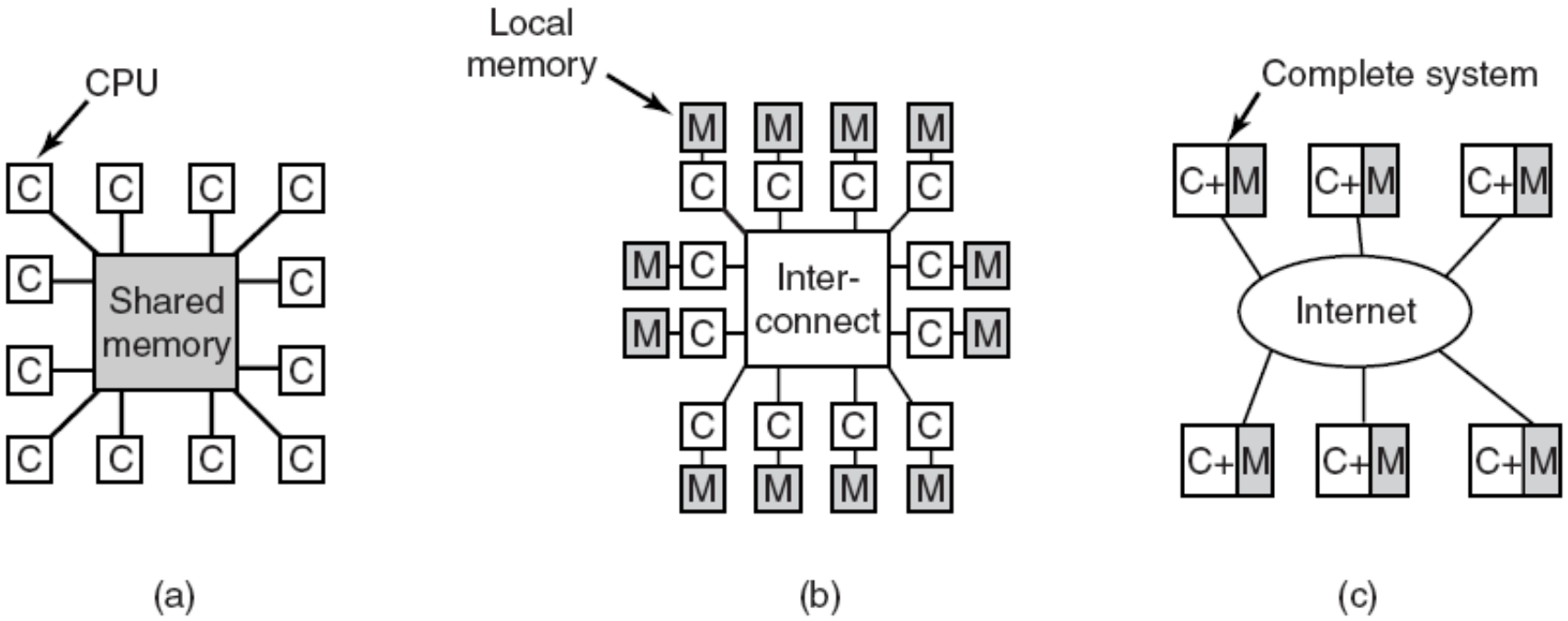


Figure 8-1. (a) A shared-memory multiprocessor. (b) A message-passing multicomputer. (c) A wide area distributed system.

UMA Multiprocessors with Bus-Based Architectures

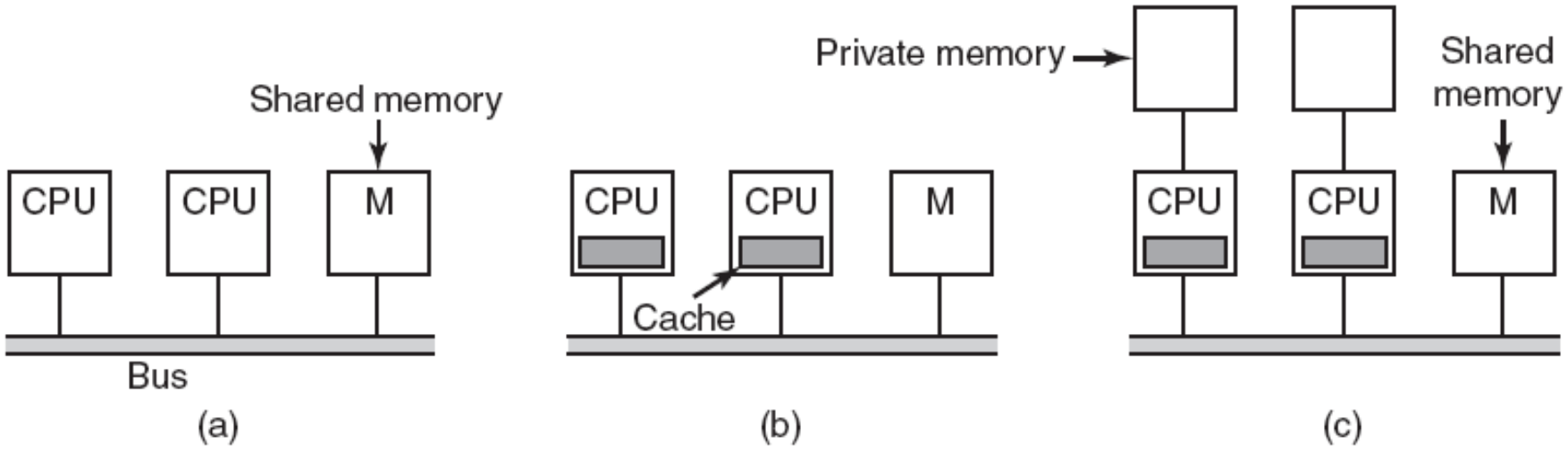


Figure 8-2. Three bus-based multiprocessors. (a) Without caching. (b) With caching. (c) With caching and private memories.

UMA Multiprocessors Using Crossbar Switches

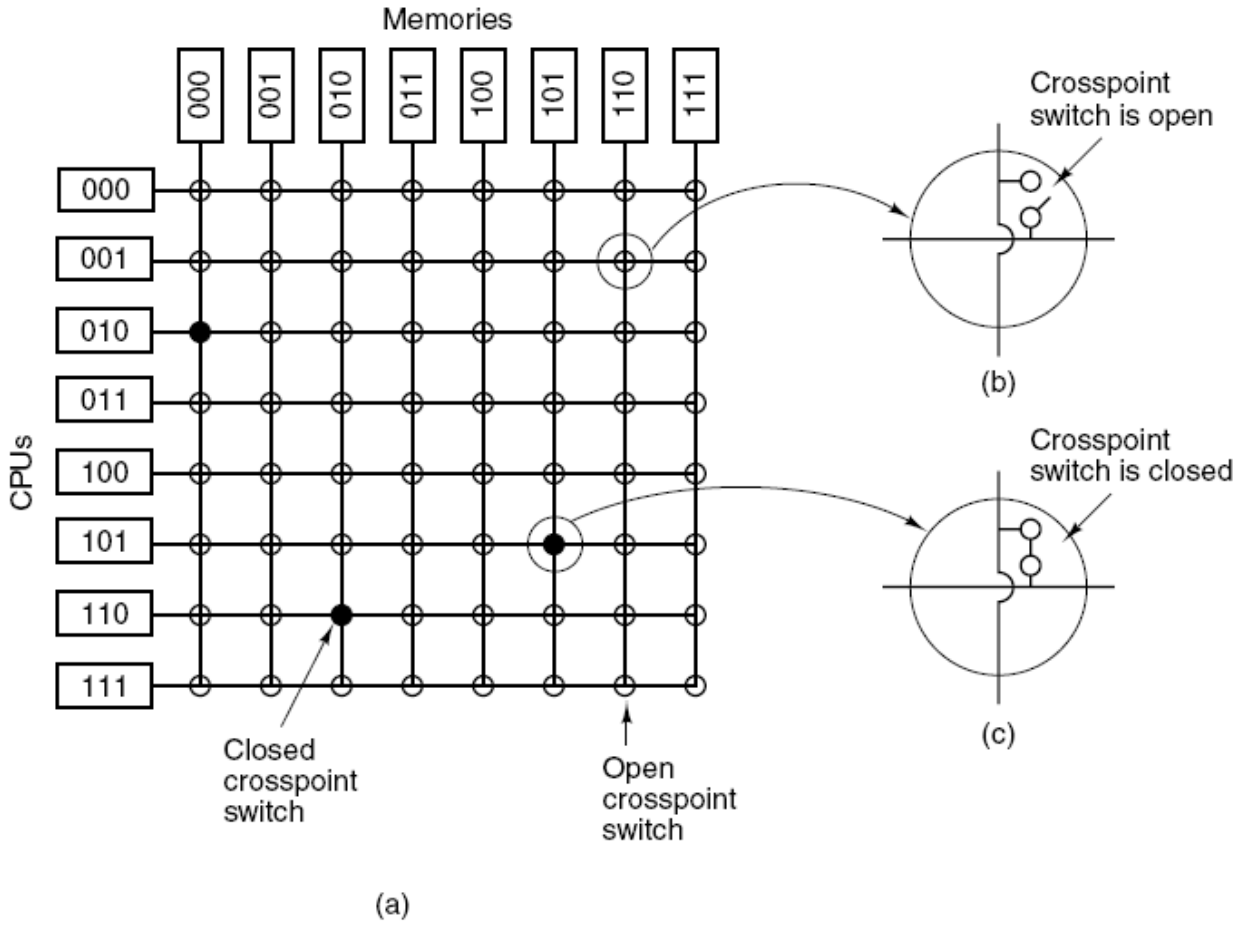


Figure 8-3. (a) An 8 × 8 crossbar switch. (b) An open crosspoint. (c) A closed crosspoint.

UMA Multiprocessors Using Multistage Switching Networks (1)

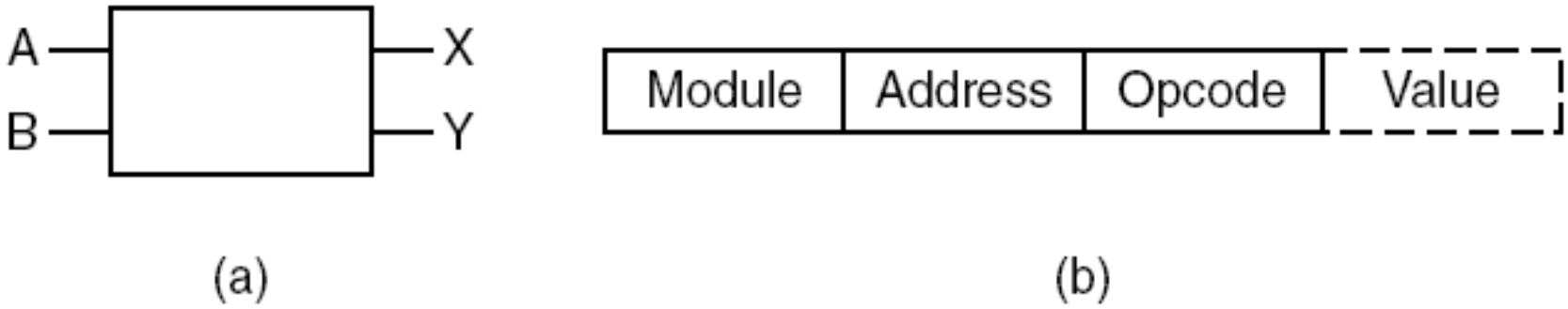


Figure 8-4. (a) A 2 × 2 switch with two input lines, A and B, and two output lines, X and Y. (b) A message format.

UMA Multiprocessors Using Multistage Switching Networks (2)

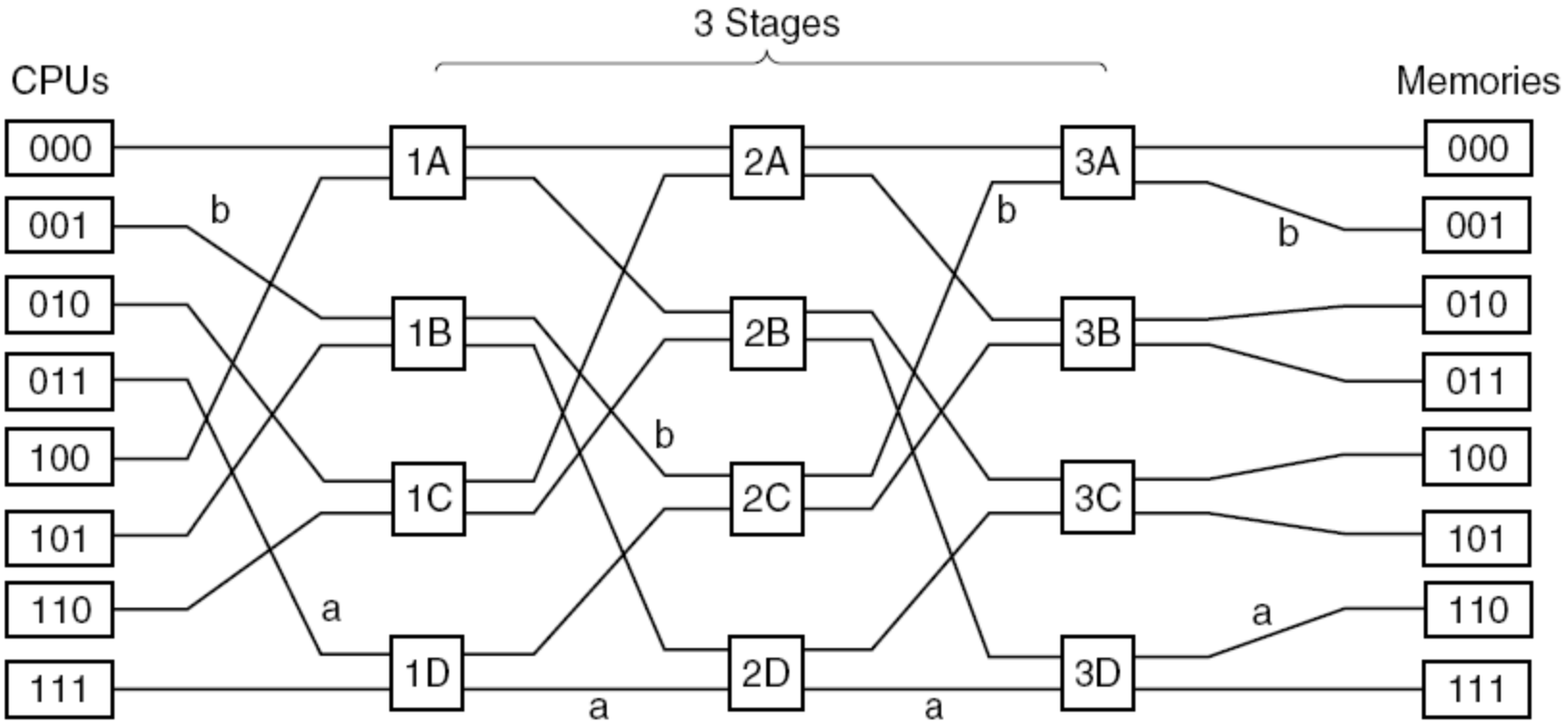


Figure 8-5. An omega switching network.

NUMA Multiprocessors (1)

Characteristics of NUMA machines:

1. There is a single address space visible to all CPUs.
2. Access to remote memory is via LOAD and STORE instructions.
3. Access to remote memory is slower than access to local memory.

NUMA Multiprocessors (2)

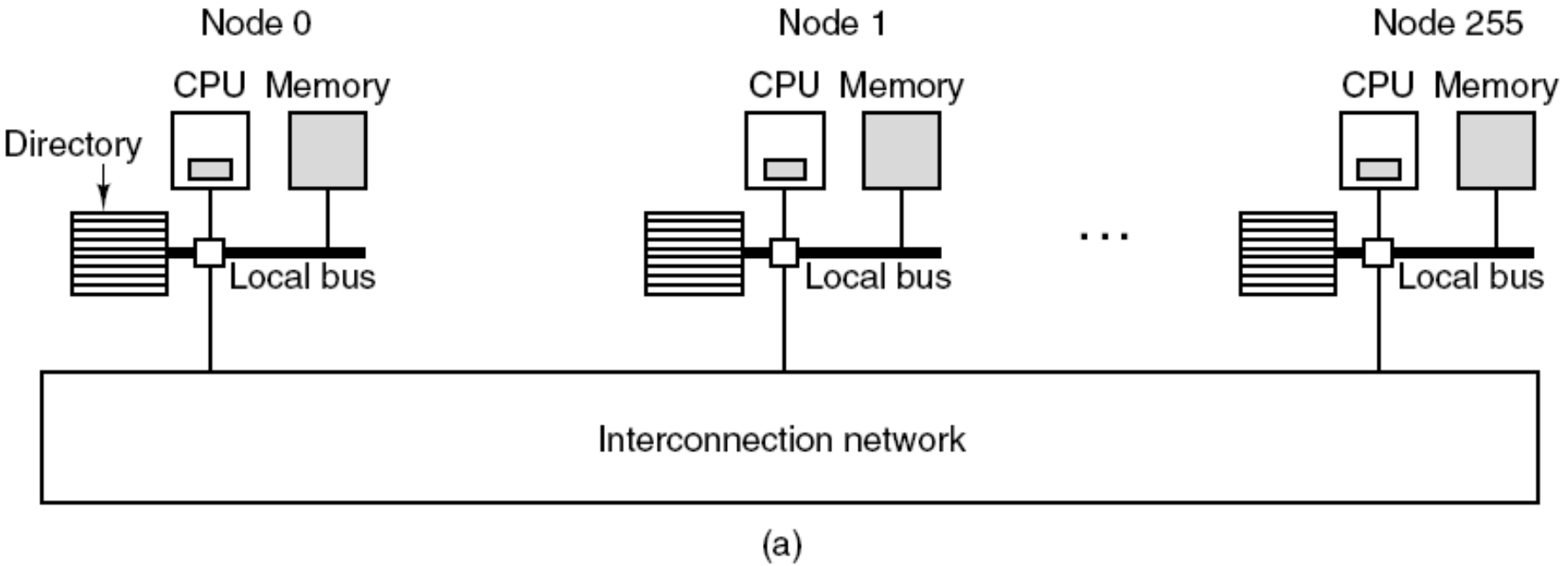


Figure 8-6. (a) A 256-node directory-based multiprocessor.

NUMA Multiprocessors (3)

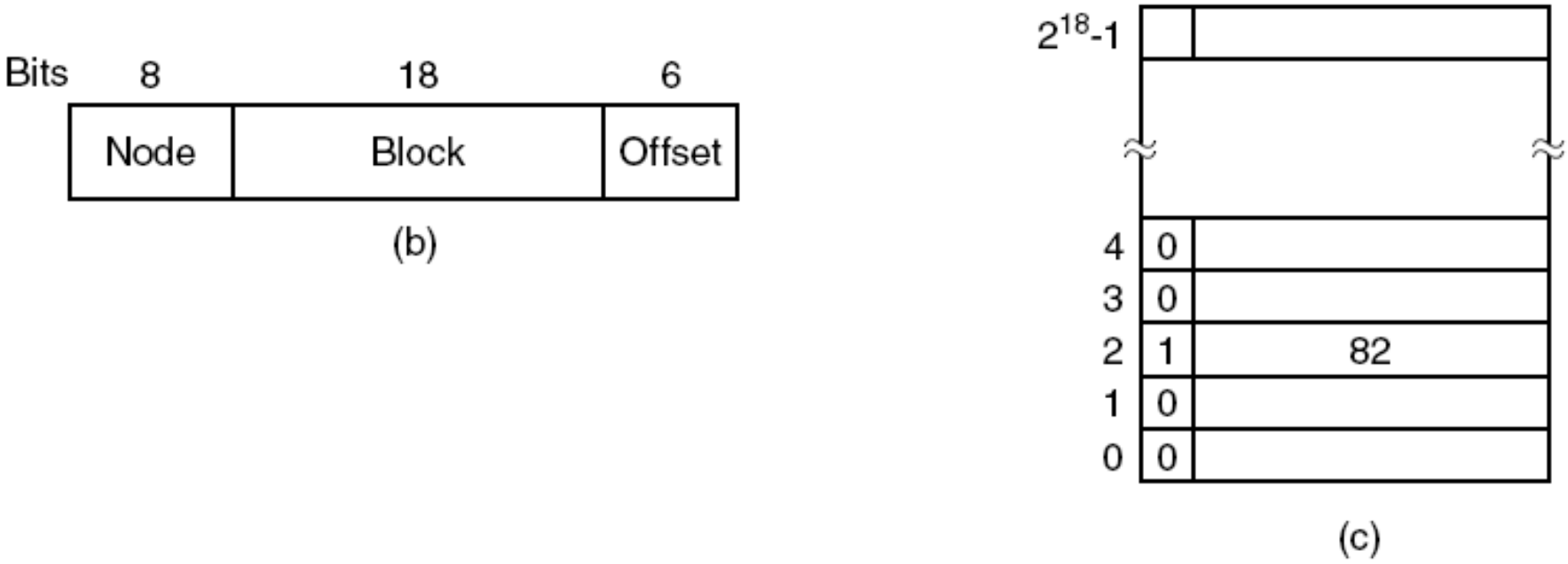


Figure 8-6. (b) Division of a 32-bit memory address into fields. (c) The directory at node 36.

Each CPU Has Its Own Operating System

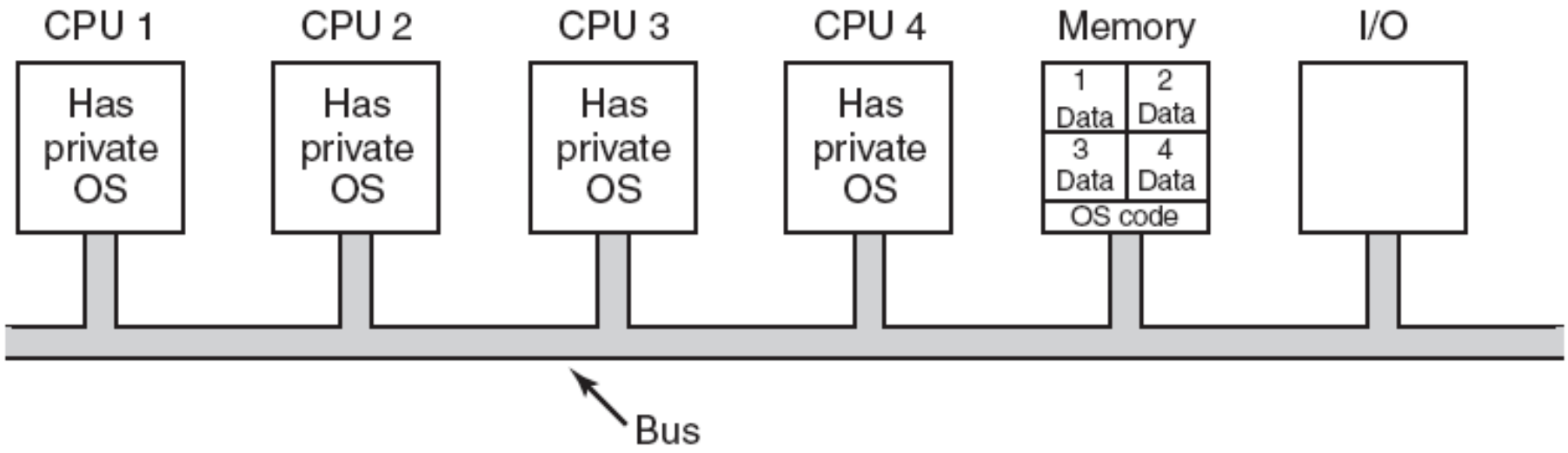


Figure 8-7. Partitioning multiprocessor memory among four CPUs, but sharing a single copy of the operating system code. The boxes marked Data are the operating system's private data for each CPU.

Master-Slave Multiprocessors

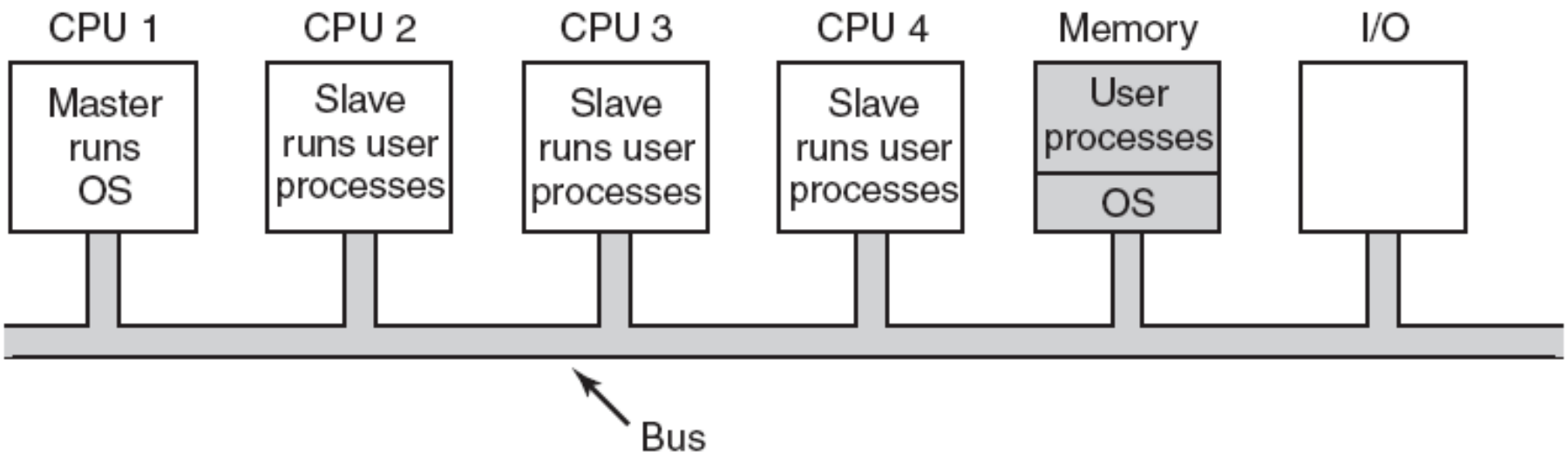


Figure 8-8. A master-slave multiprocessor model.

Symmetric Multiprocessors

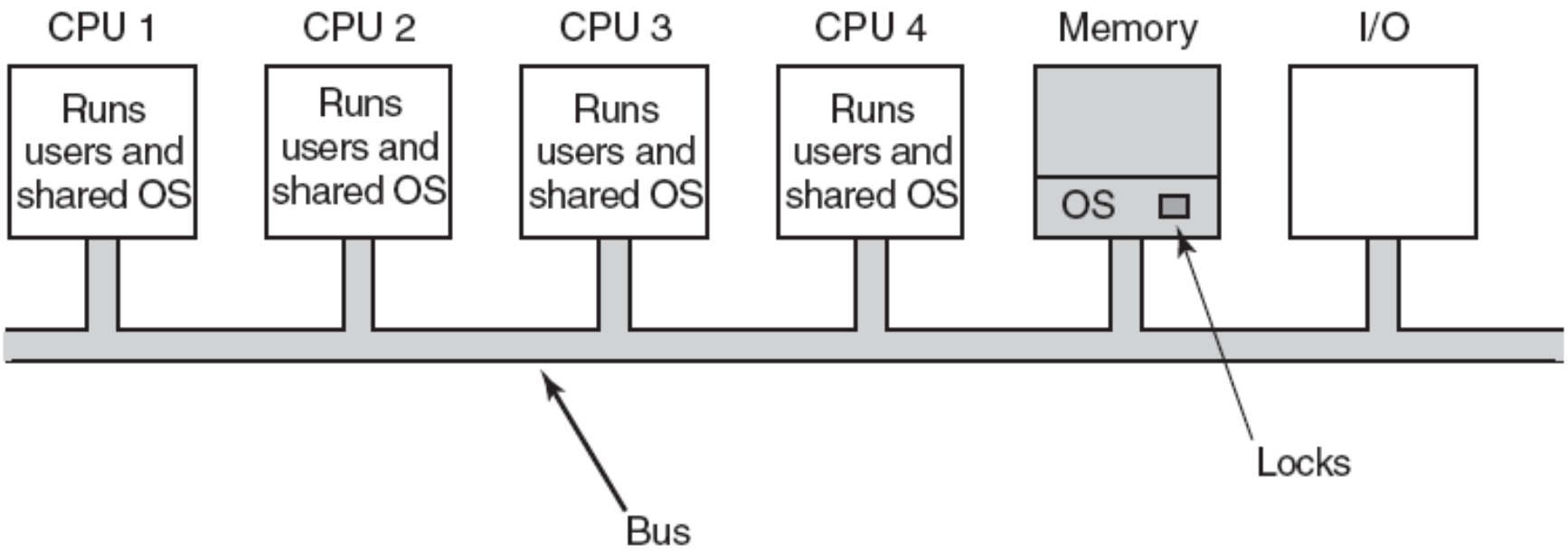


Figure 8-9. The SMP multiprocessor model.

Multiprocessor Synchronization (1)

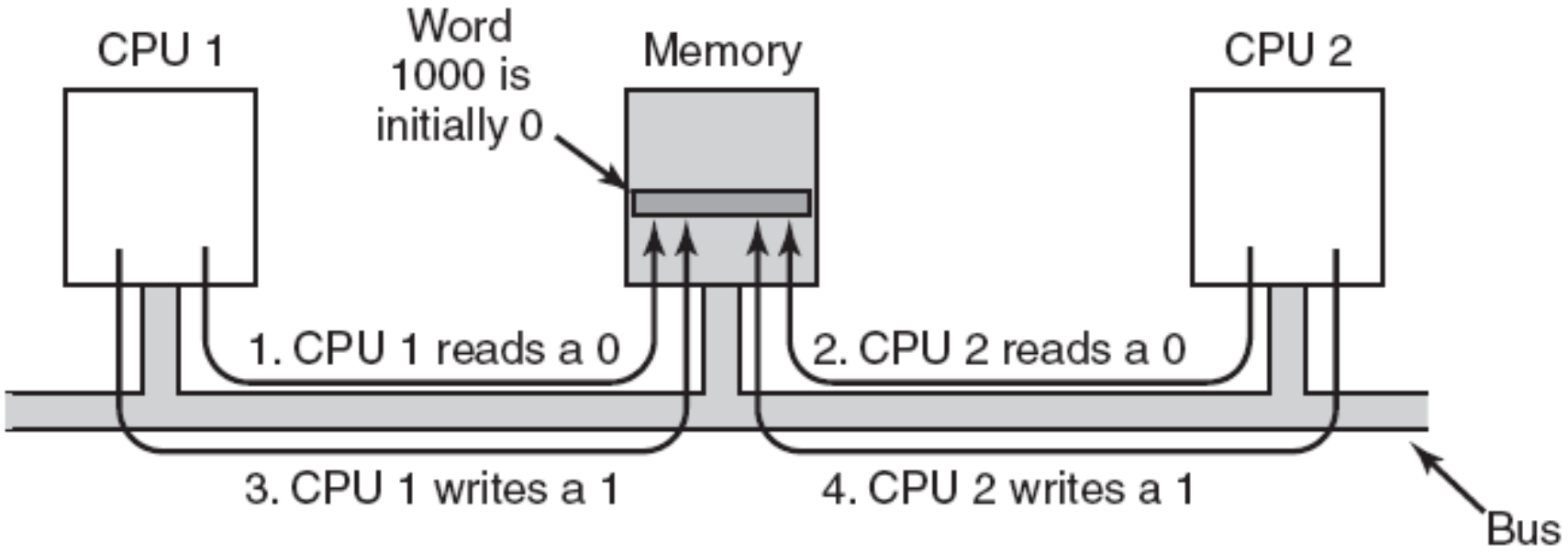


Figure 8-10. The TSL instruction can fail if the bus cannot be locked. These four steps show a sequence of events where the failure is demonstrated.

Multiprocessor Synchronization (2)

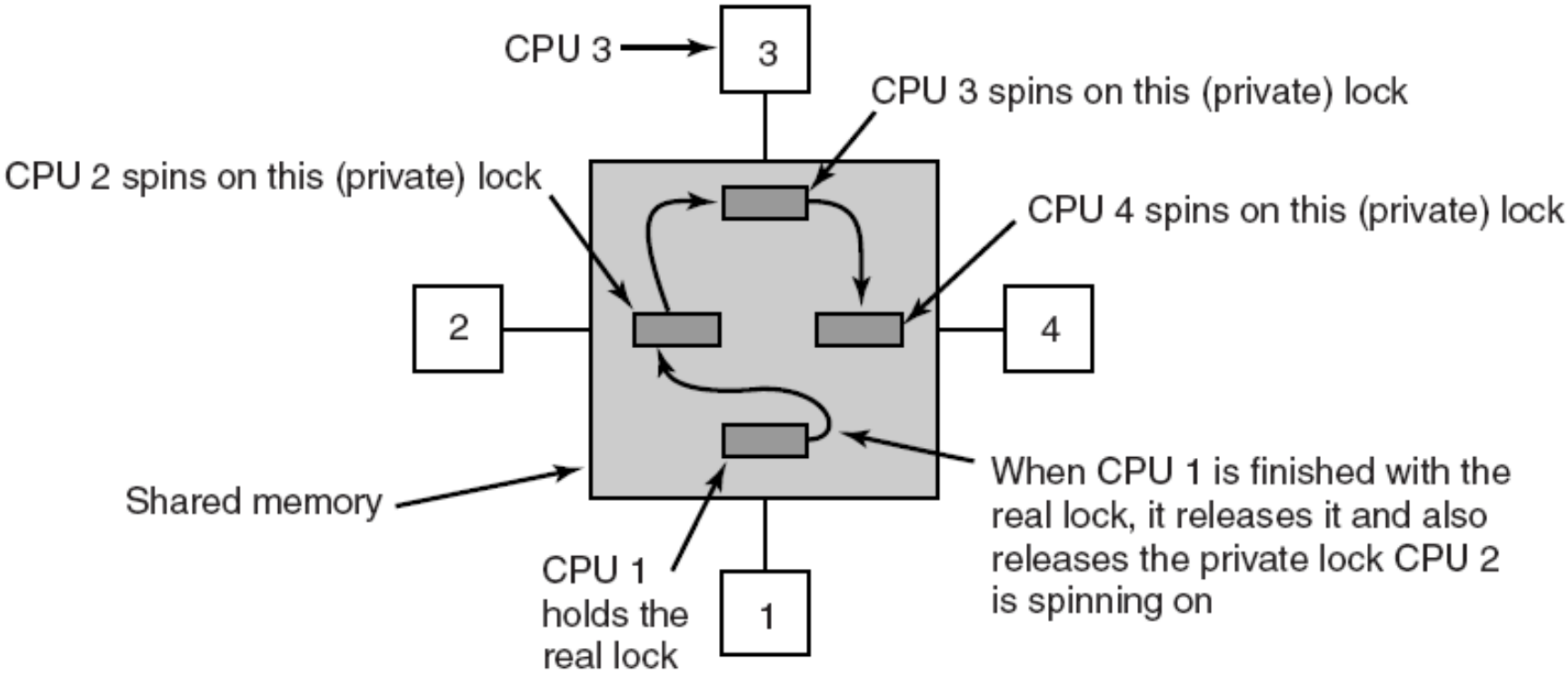


Figure 8-11. Use of multiple locks to avoid cache thrashing.

Timesharing

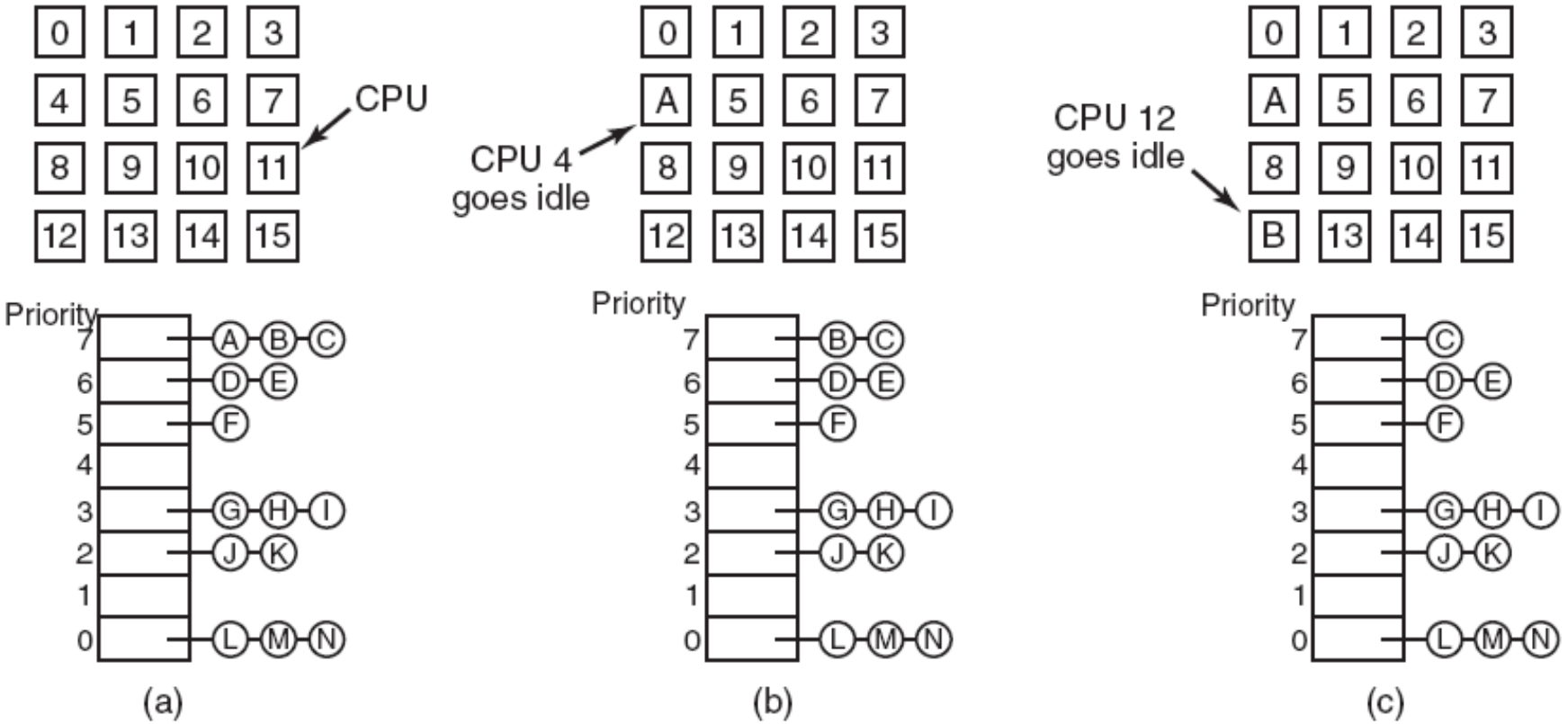


Figure 8-12. Using a single data structure for scheduling a multiprocessor.

Space Sharing

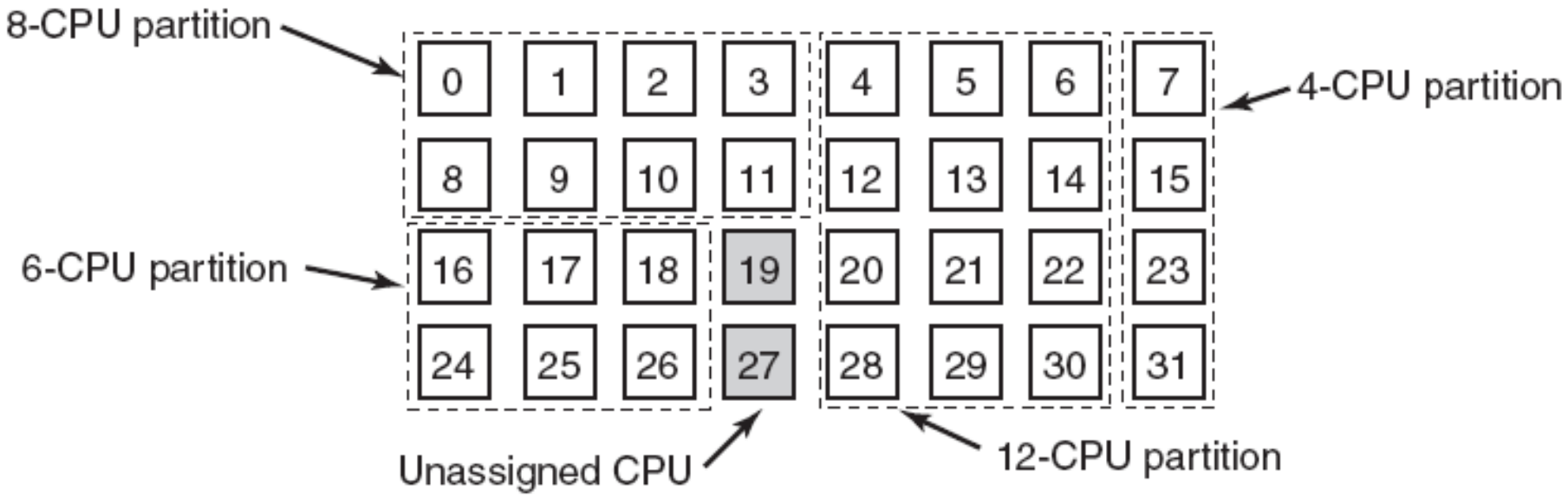


Figure 8-13. A set of 32 CPUs split into four partitions, with two CPUs available.

Gang Scheduling (1)

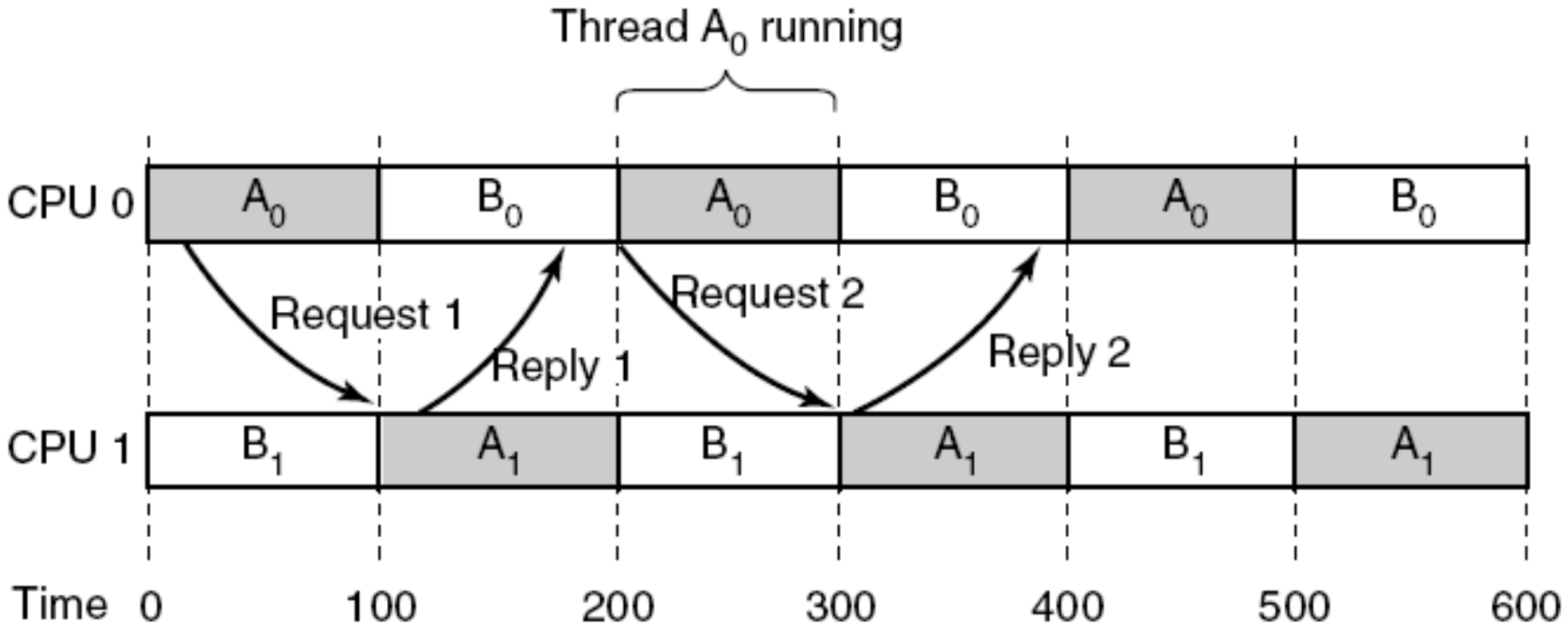


Figure 8-14. Communication between two threads belonging to thread A that are running out of phase.

Gang Scheduling (2)

The three parts of gang scheduling:

1. Groups of related threads are scheduled as a unit, a gang.
2. All members of a gang run simultaneously, on different timeshared CPUs.
3. All gang members start and end their time slices together.

Gang Scheduling (3)

Time slot	CPU 0	CPU 1	CPU 2	CPU 3	CPU 4	CPU 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

Figure 8-15. Gang scheduling.

Interconnection Technology (1)

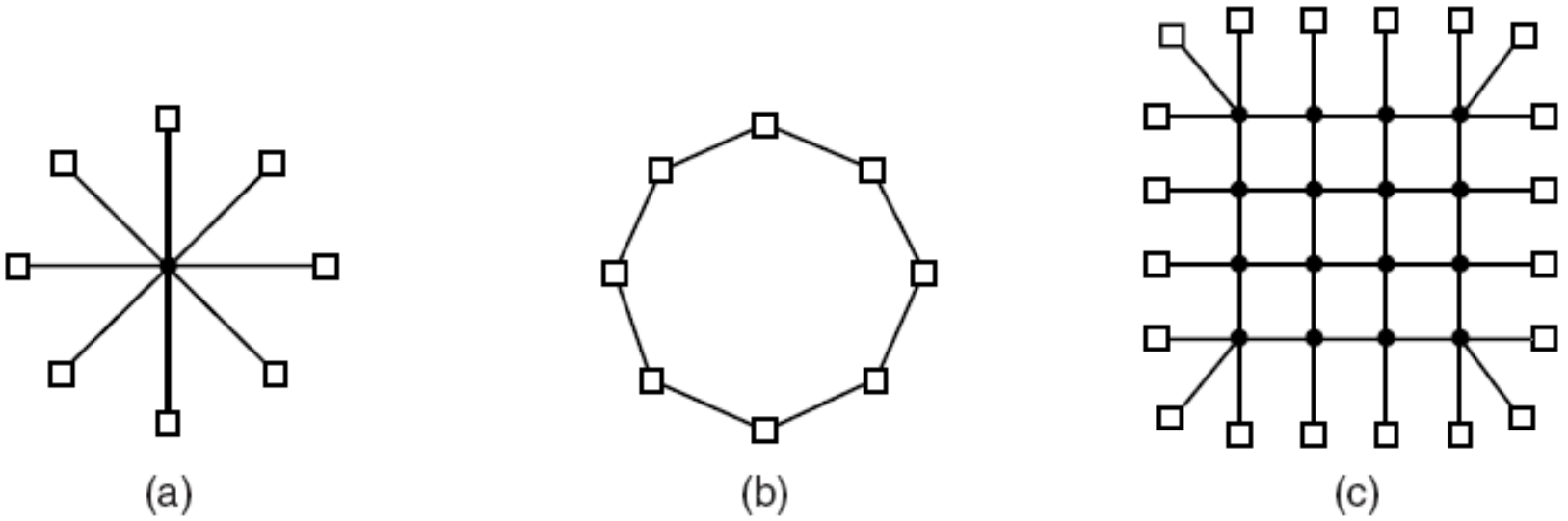


Figure 8-16. Various interconnect topologies. (a) A single switch. (b) A ring. (c) A grid.

Interconnection Technology (2)

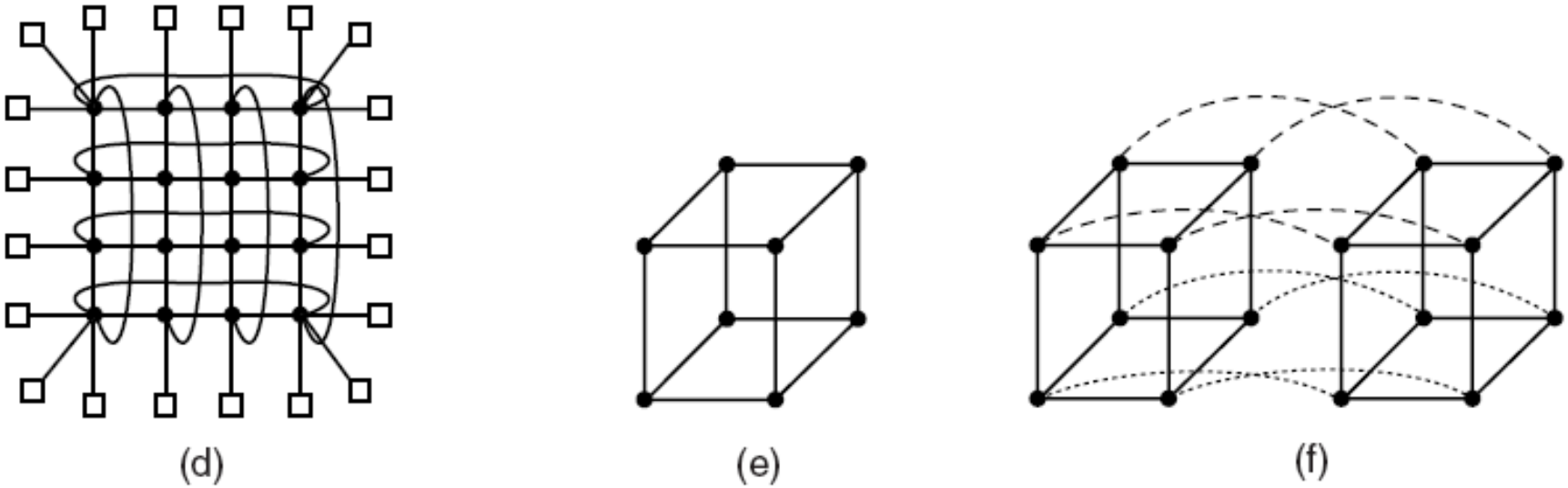


Figure 8-16. Various interconnect topologies. (d) A double torus. (e) A cube. (f) A 4D hypercube.

Interconnection Technology (3)

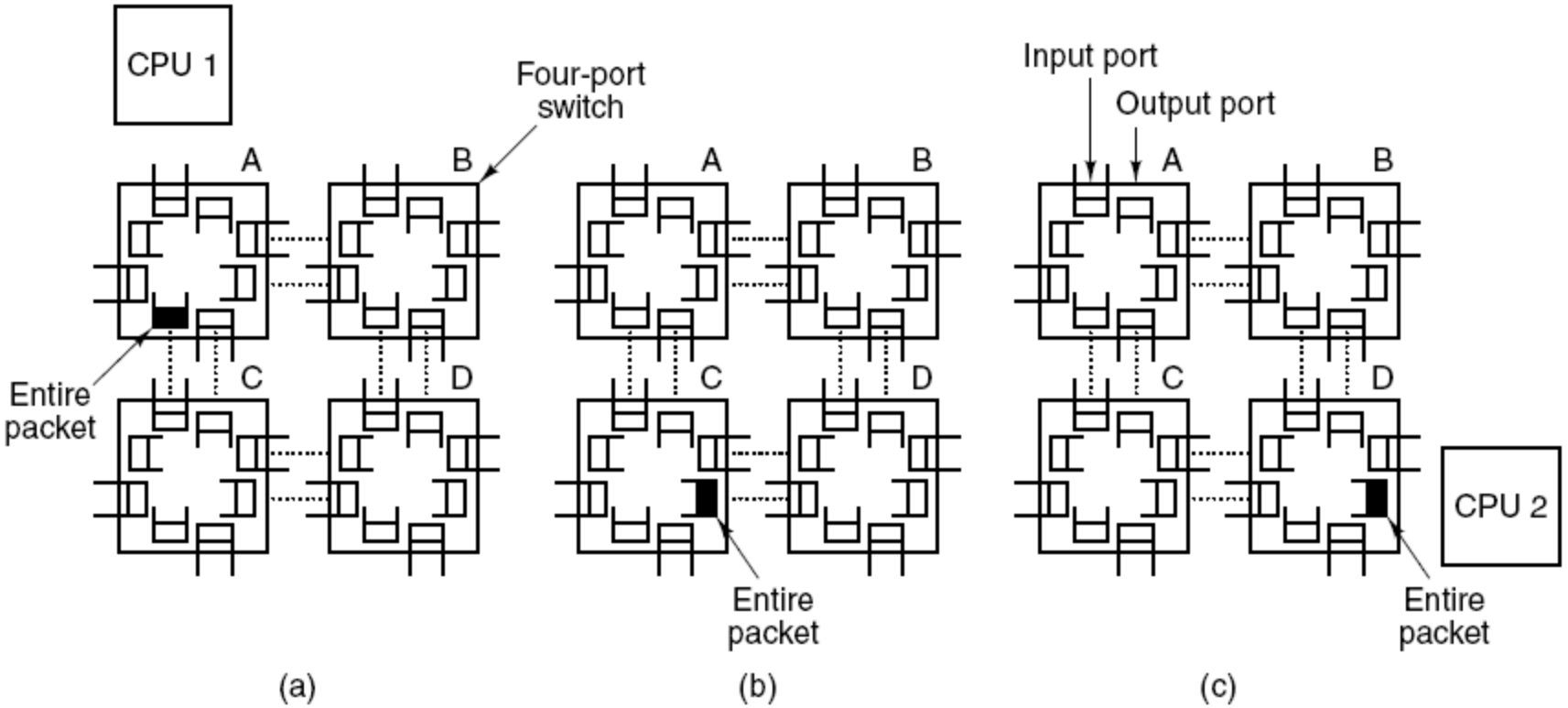


Figure 8-17. Store-and-forward packet switching.

Network Interfaces

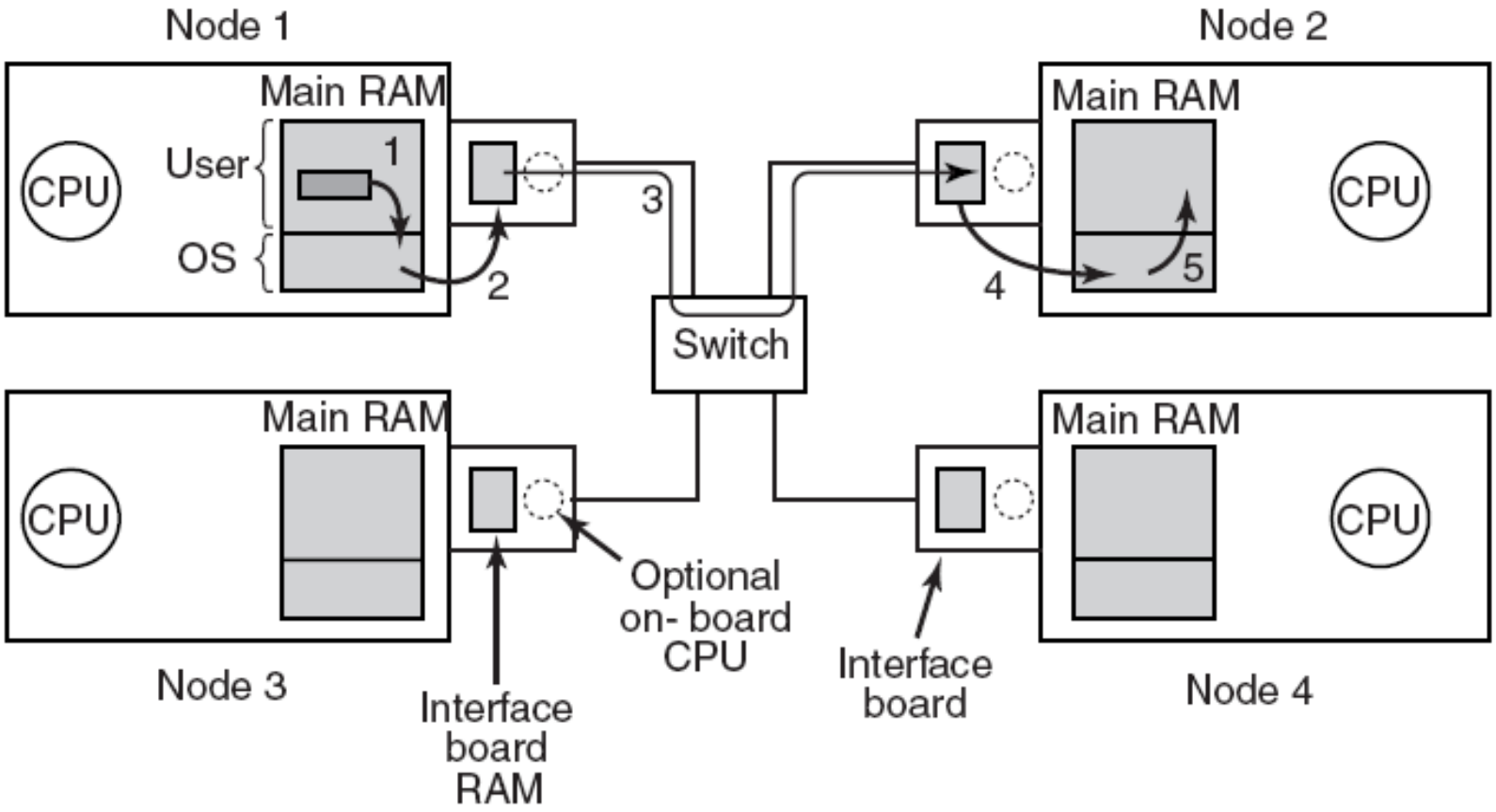


Figure 8-18. Position of the network interface boards in a multicomputer.

Blocking versus Nonblocking Calls (1)

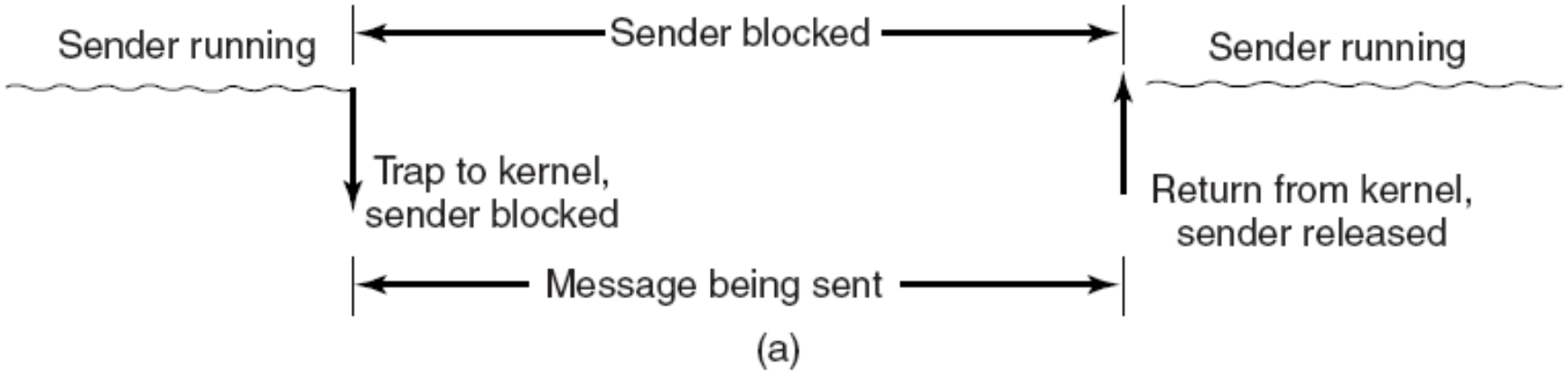


Figure 8-19. (a) A blocking send call.

Blocking versus Nonblocking Calls (2)

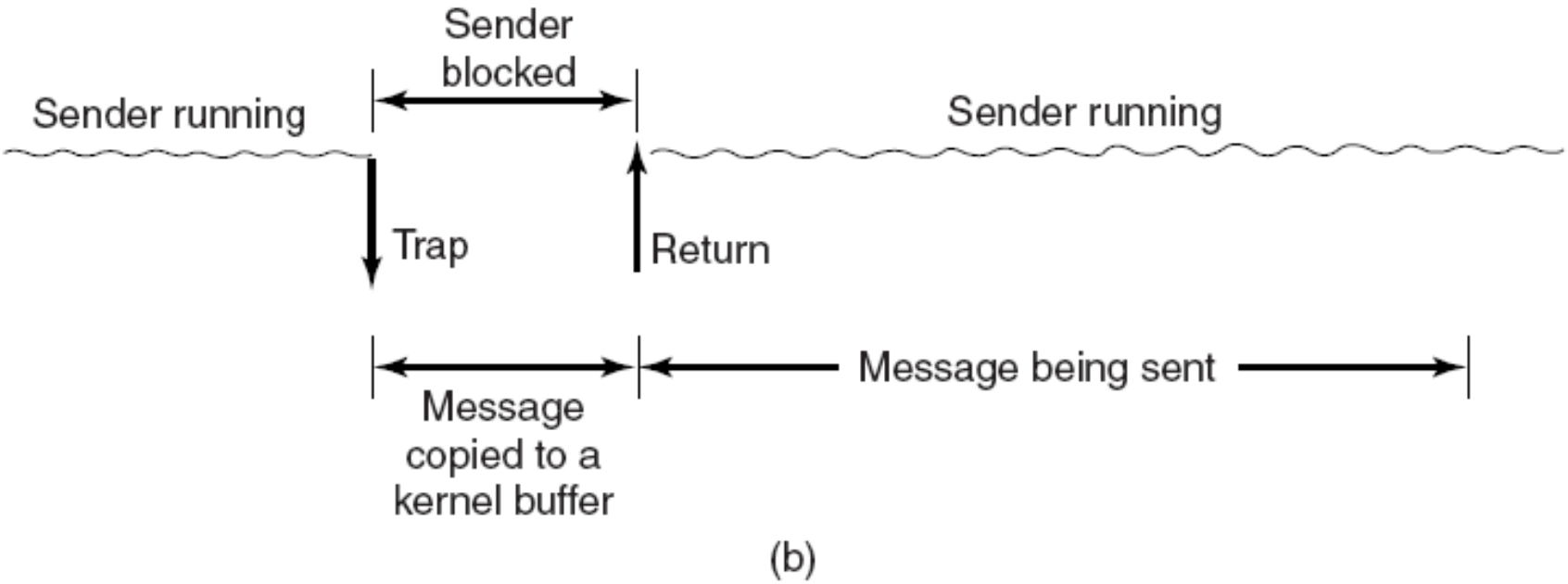


Figure 8-19. (b) A nonblocking send call.

Blocking versus Nonblocking Calls (3)

Choices on the sending side:

1. Blocking send (CPU idle during message transmission).
2. Nonblocking send with copy (CPU time wasted for the extra copy).
3. Nonblocking send with interrupt (makes programming difficult).
4. Copy on write (extra copy probably needed eventually).

Remote Procedure Call

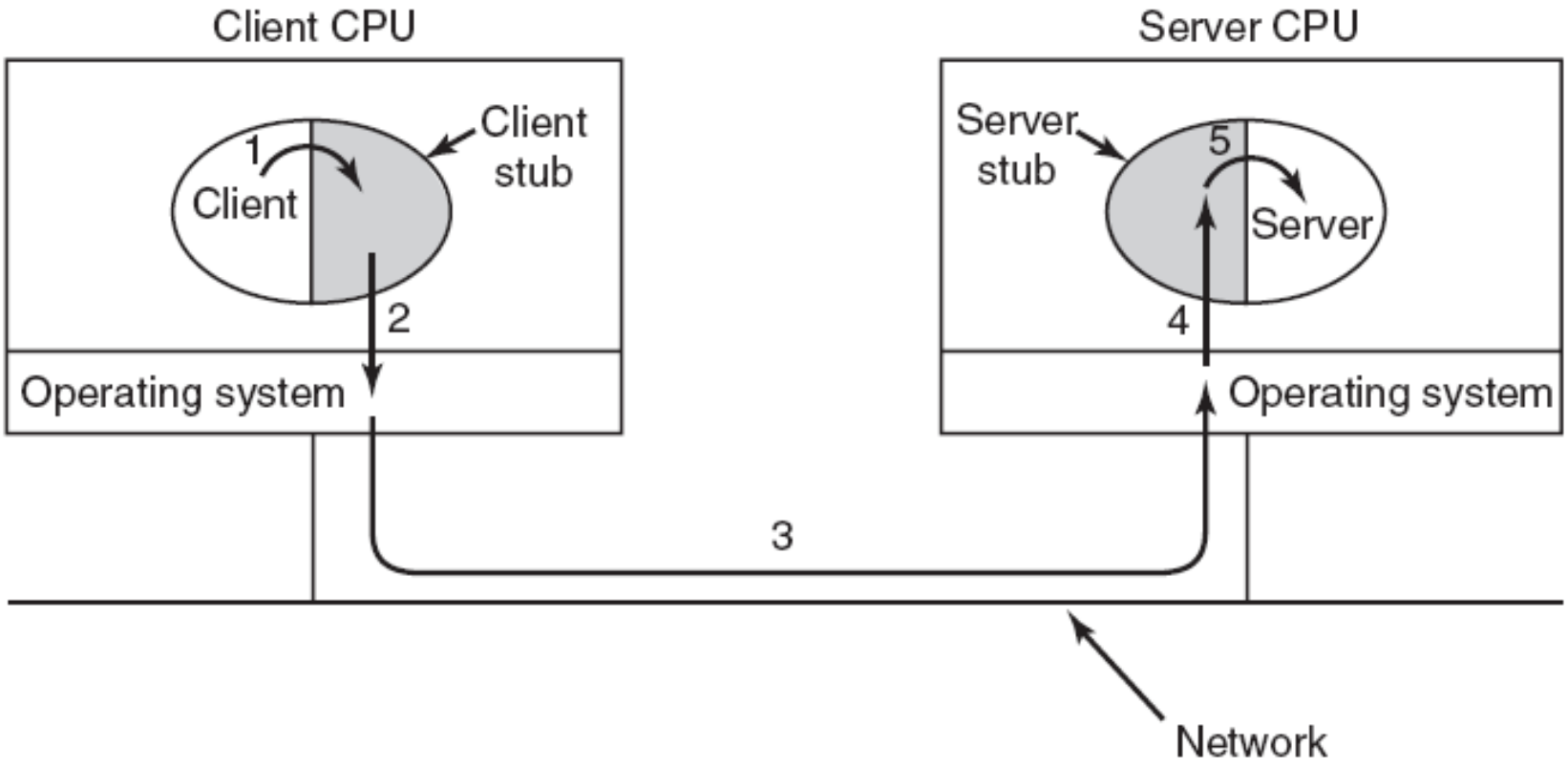
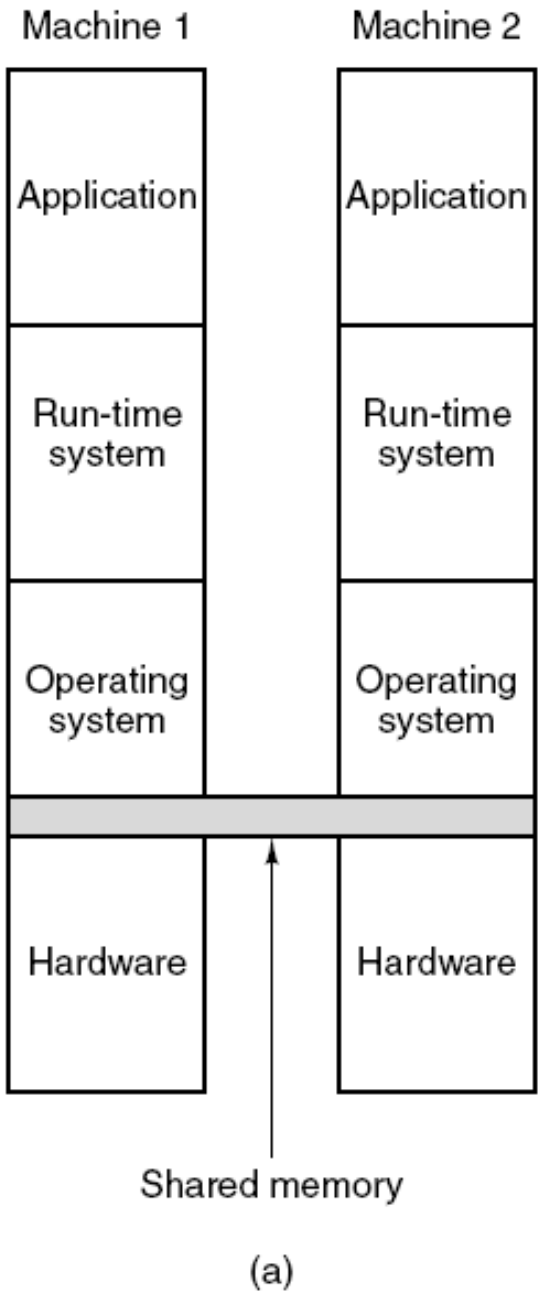


Figure 8-20. Steps in making a remote procedure call. The stubs are shaded gray.

Distributed Shared Memory (1)

Figure 8-21. Various layers where shared memory can be implemented. (a) The hardware.



Distributed Shared Memory (2)

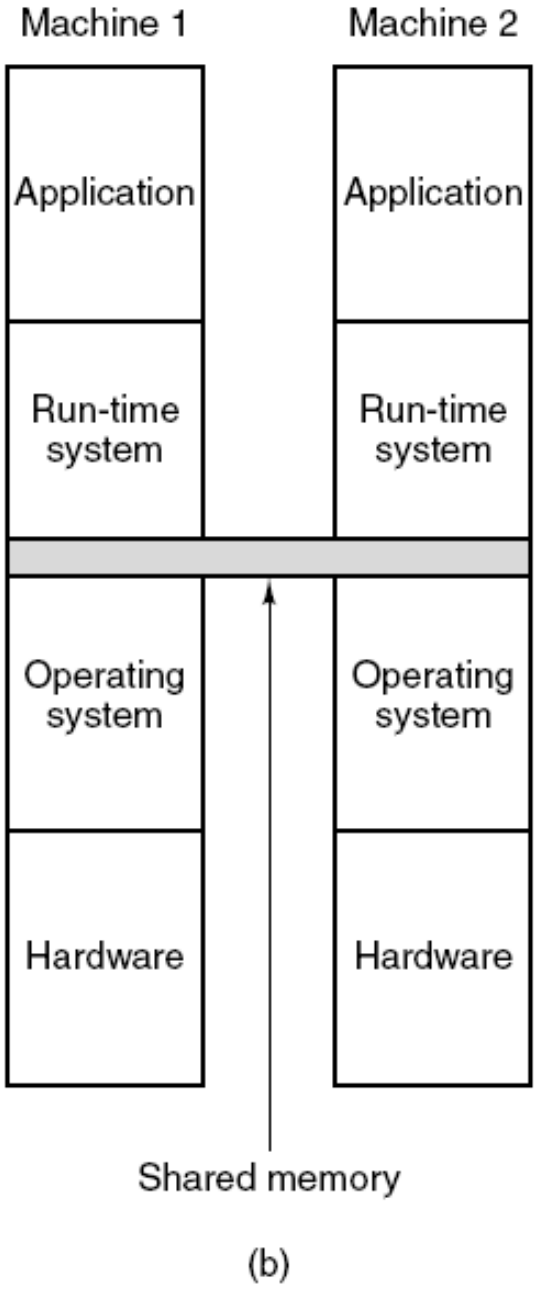
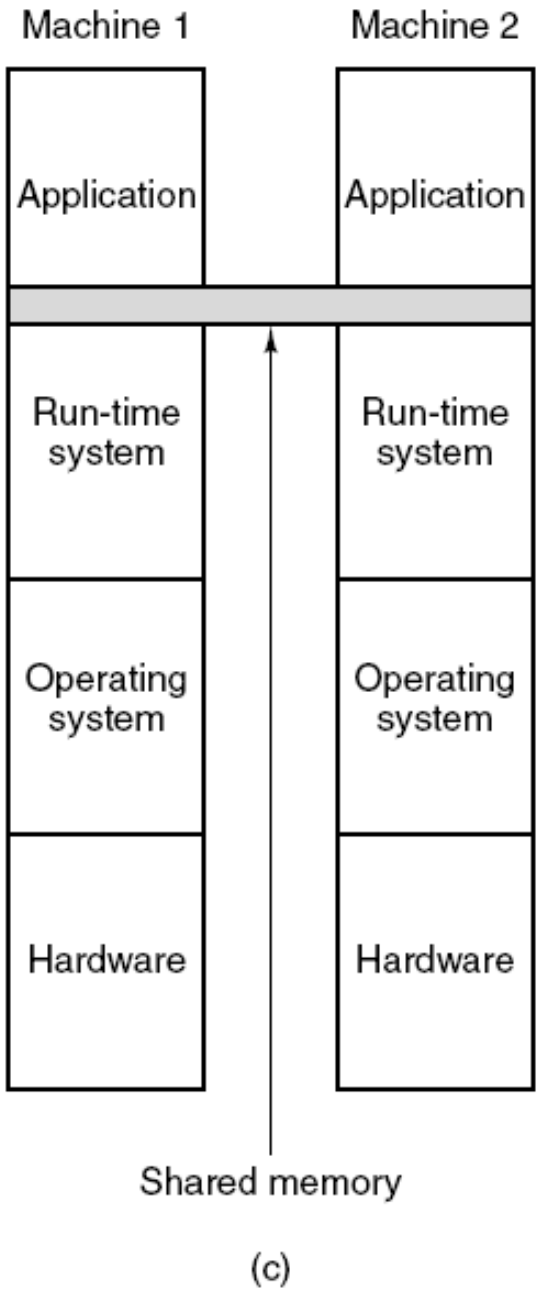


Figure 8-21. Various layers where shared memory can be implemented. (b) The operating system.

Distributed Shared Memory (3)

Figure 8-21. Various layers where shared memory can be implemented. (c) User-level software.



Distributed Shared Memory (4)

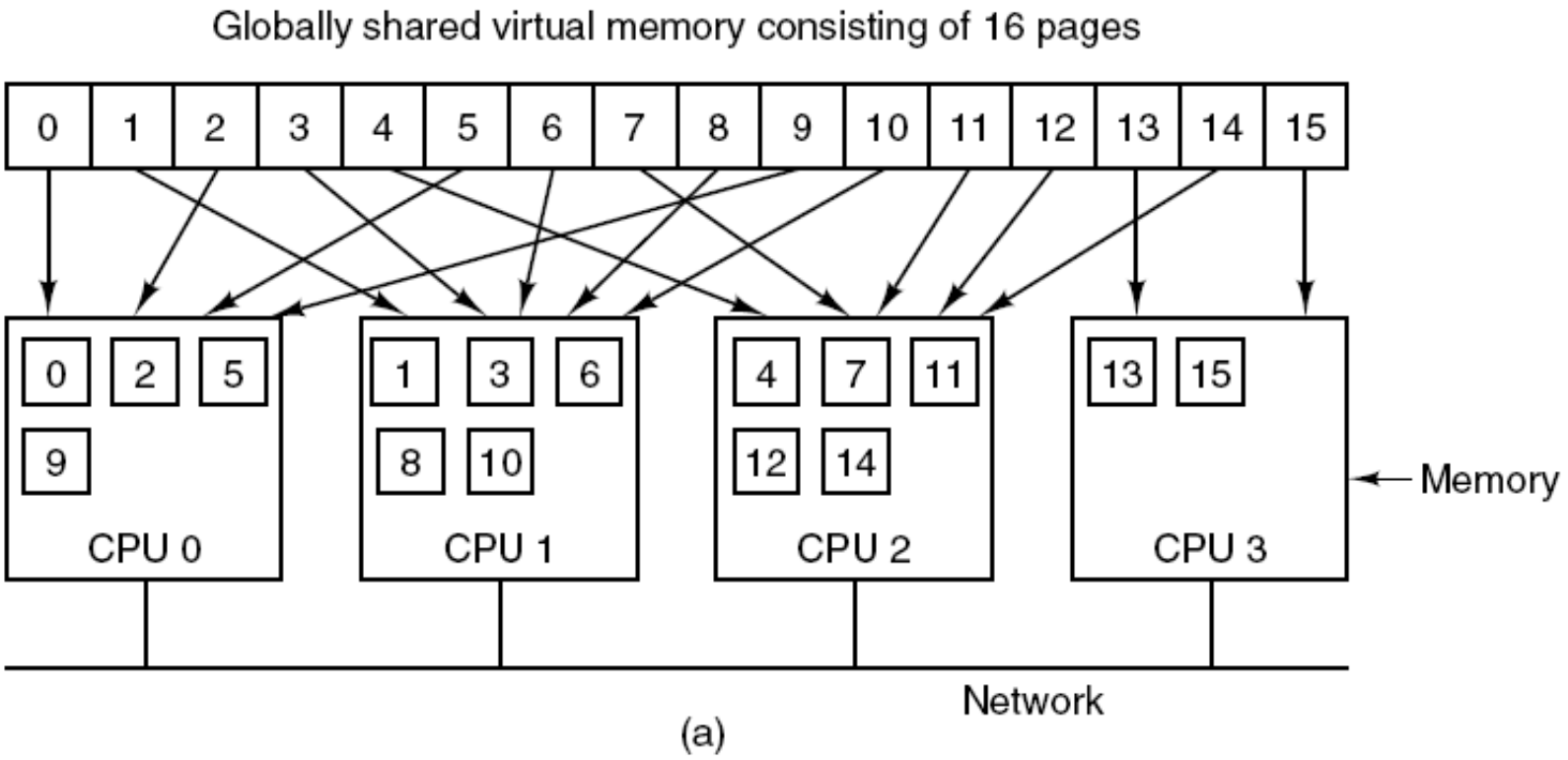


Figure 8-22. (a) Pages of the address space distributed among four machines.

Distributed Shared Memory (5)

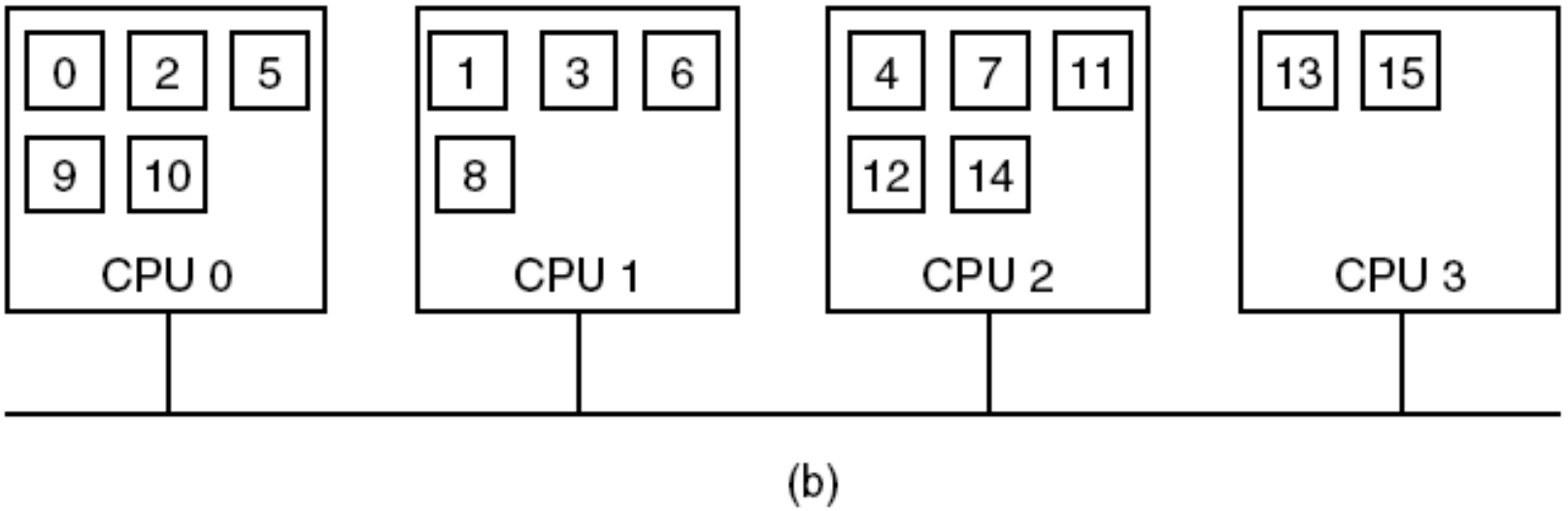


Figure 8-22. (b) Situation after CPU 1 references page 10 and the page is moved there.

Distributed Shared Memory (6)

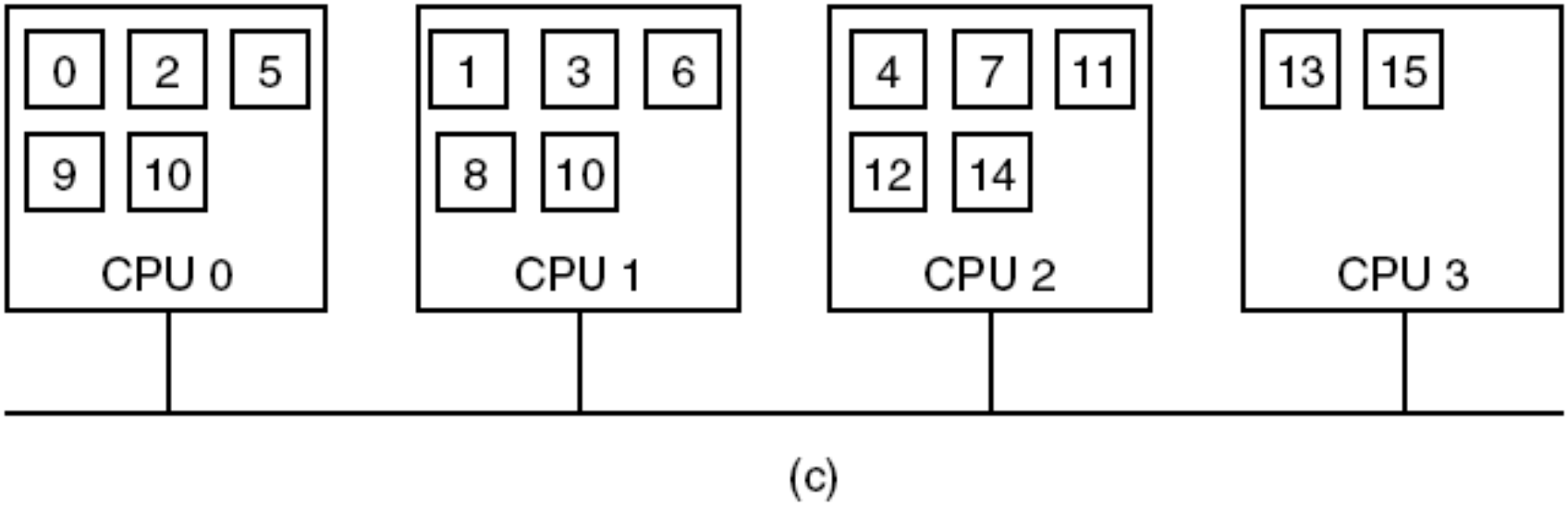


Figure 8-22. (c) Situation if page 10 is read only and replication is used.

False Sharing

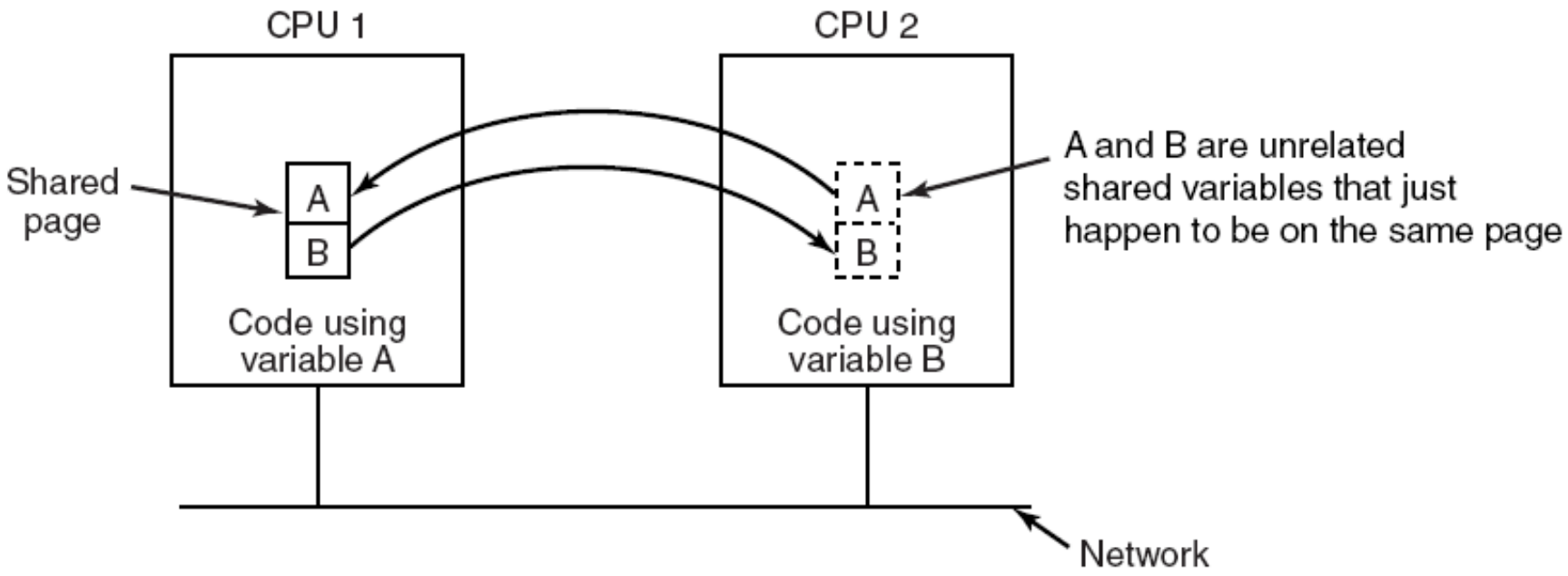


Figure 8-23. False sharing of a page containing two unrelated variables.

A Graph-Theoretic Deterministic Algorithm

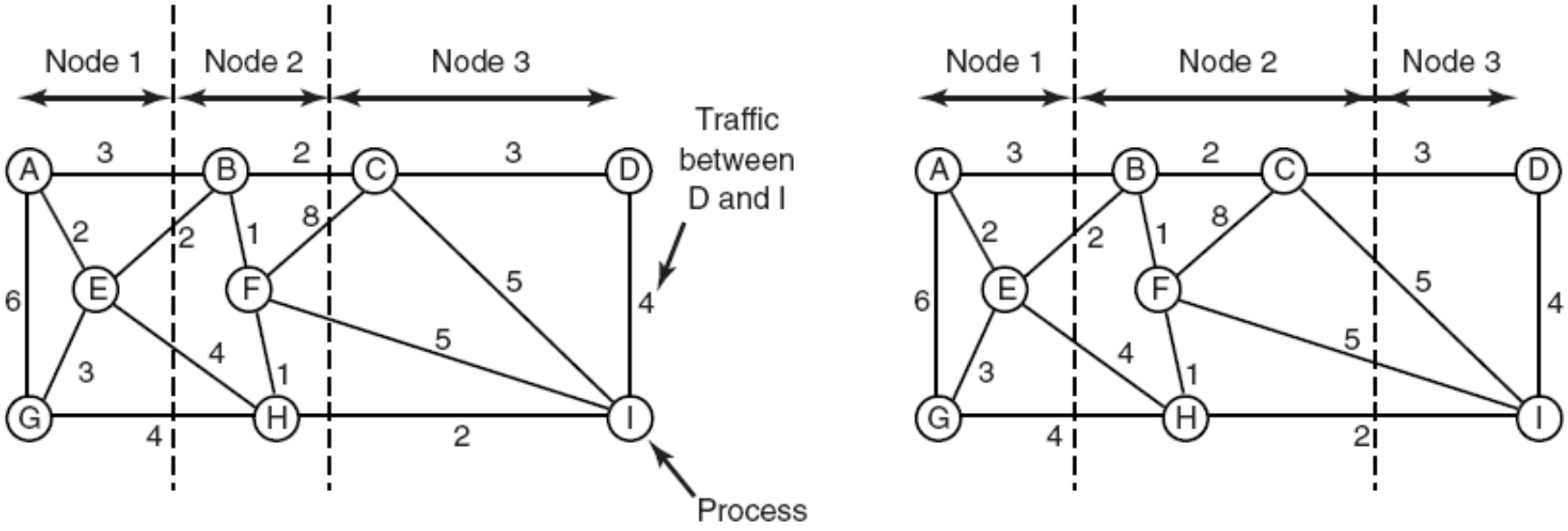


Figure 8-24. Two ways of allocating nine processes to three nodes.

A Sender-Initiated Distributed Heuristic Algorithm

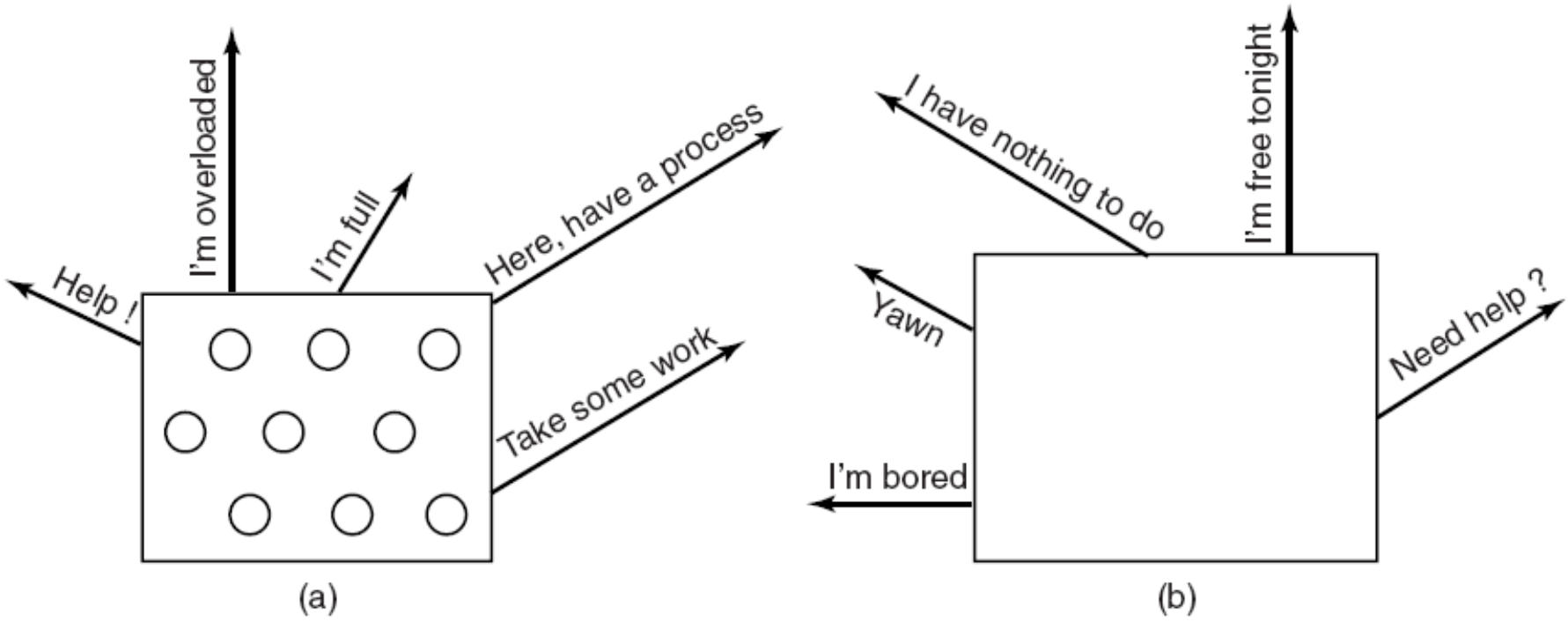


Figure 8-25. (a) An overloaded node looking for a lightly loaded node to hand off processes to. (b) An empty node looking for work to do.

Type 1 Hypervisors

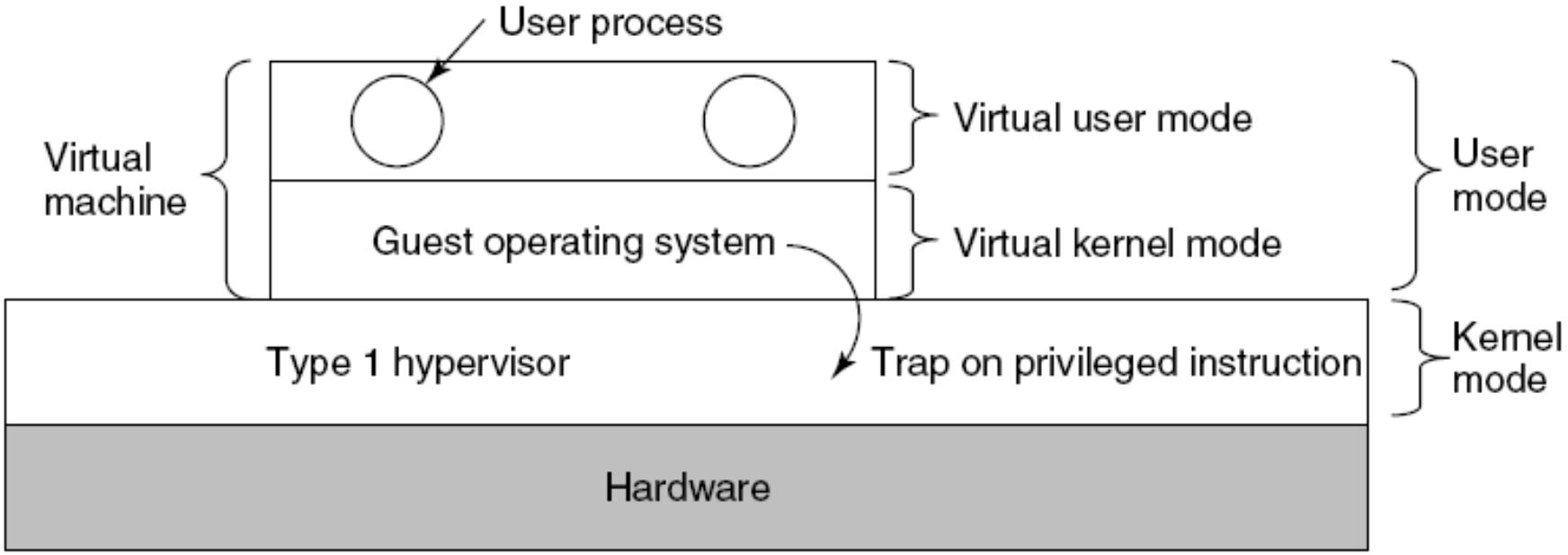


Figure 8-26. When the operating system in a virtual machine executes a kernel-only instruction, it traps to the hypervisor if virtualization technology is present.

Paravirtualization (1)

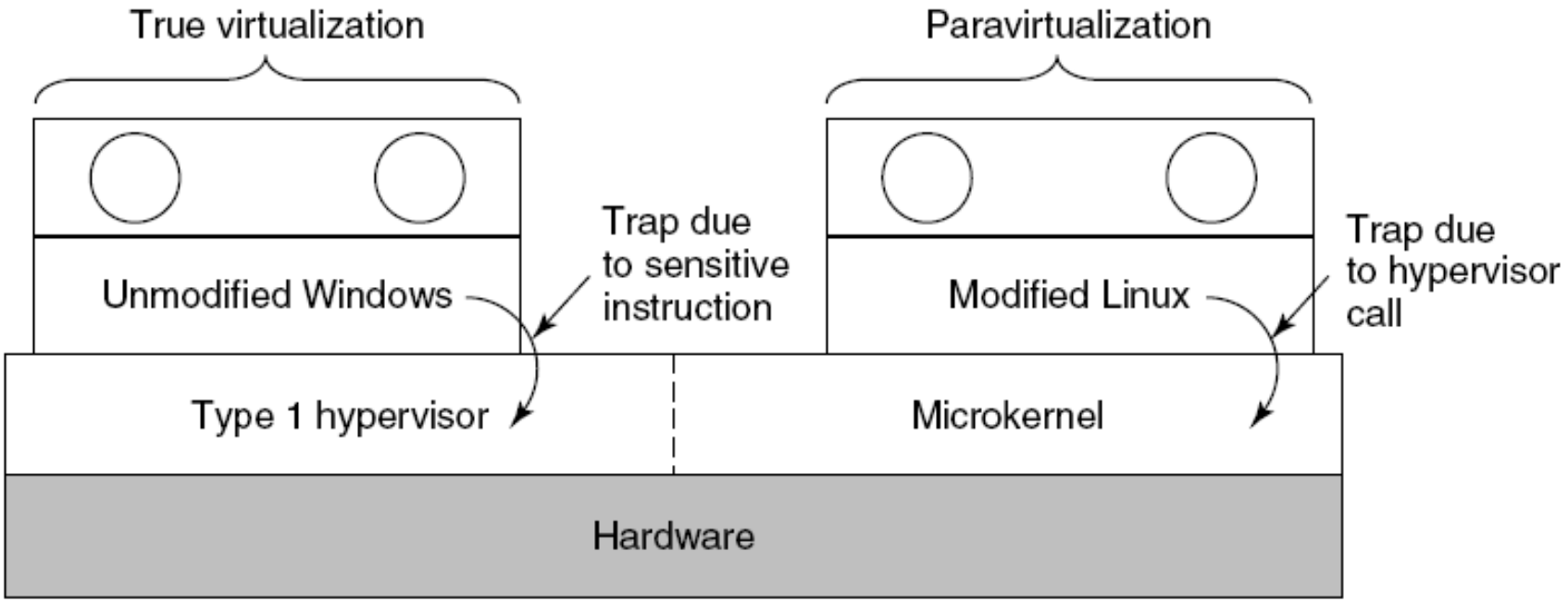


Figure 8-27. A hypervisor supporting both true virtualization and paravirtualization.

Paravirtualization (2)

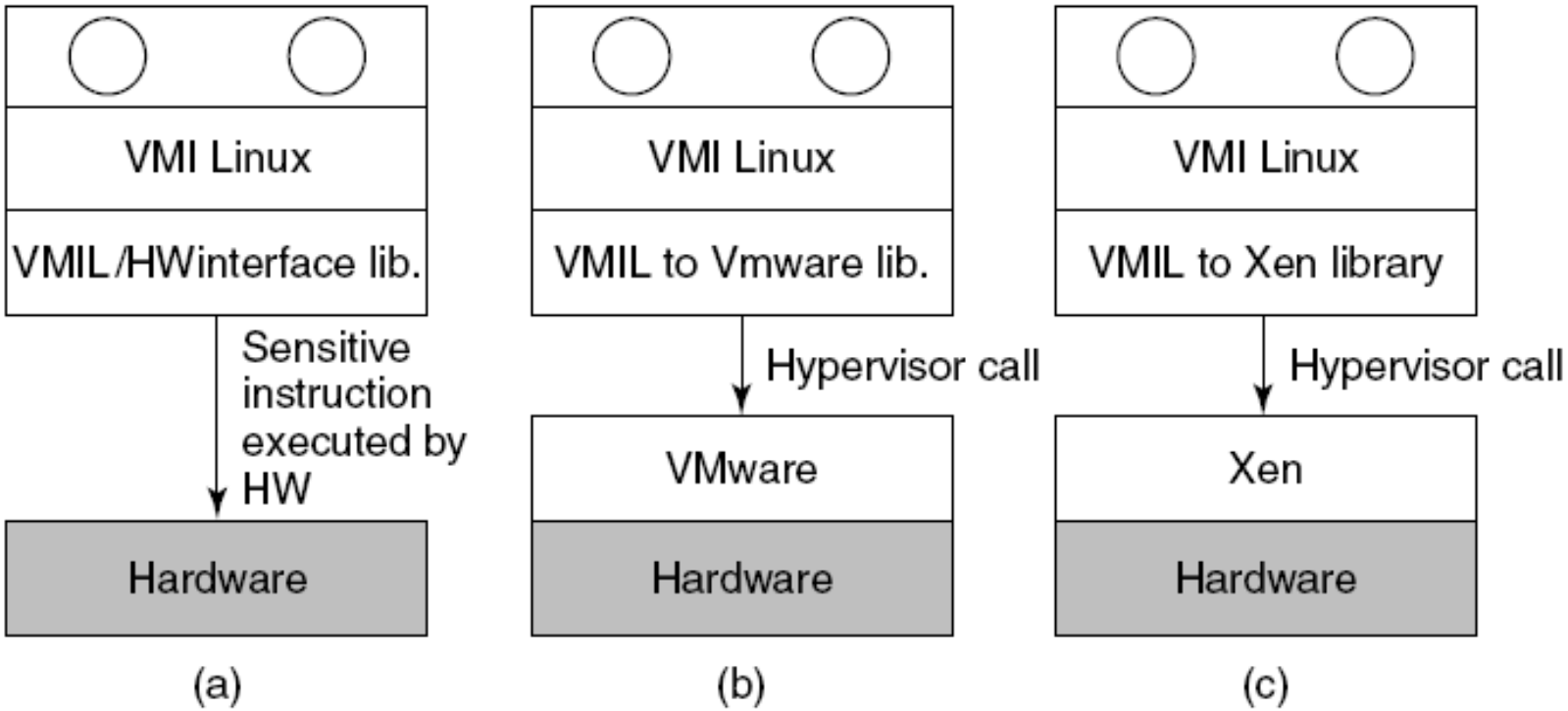


Figure 8-28. VMI Linux running on (a) the bare hardware (b) VMware (c) Xen.

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

Figure 8-29. Comparison of three kinds of multiple CPU systems.

Distributed Systems (2)

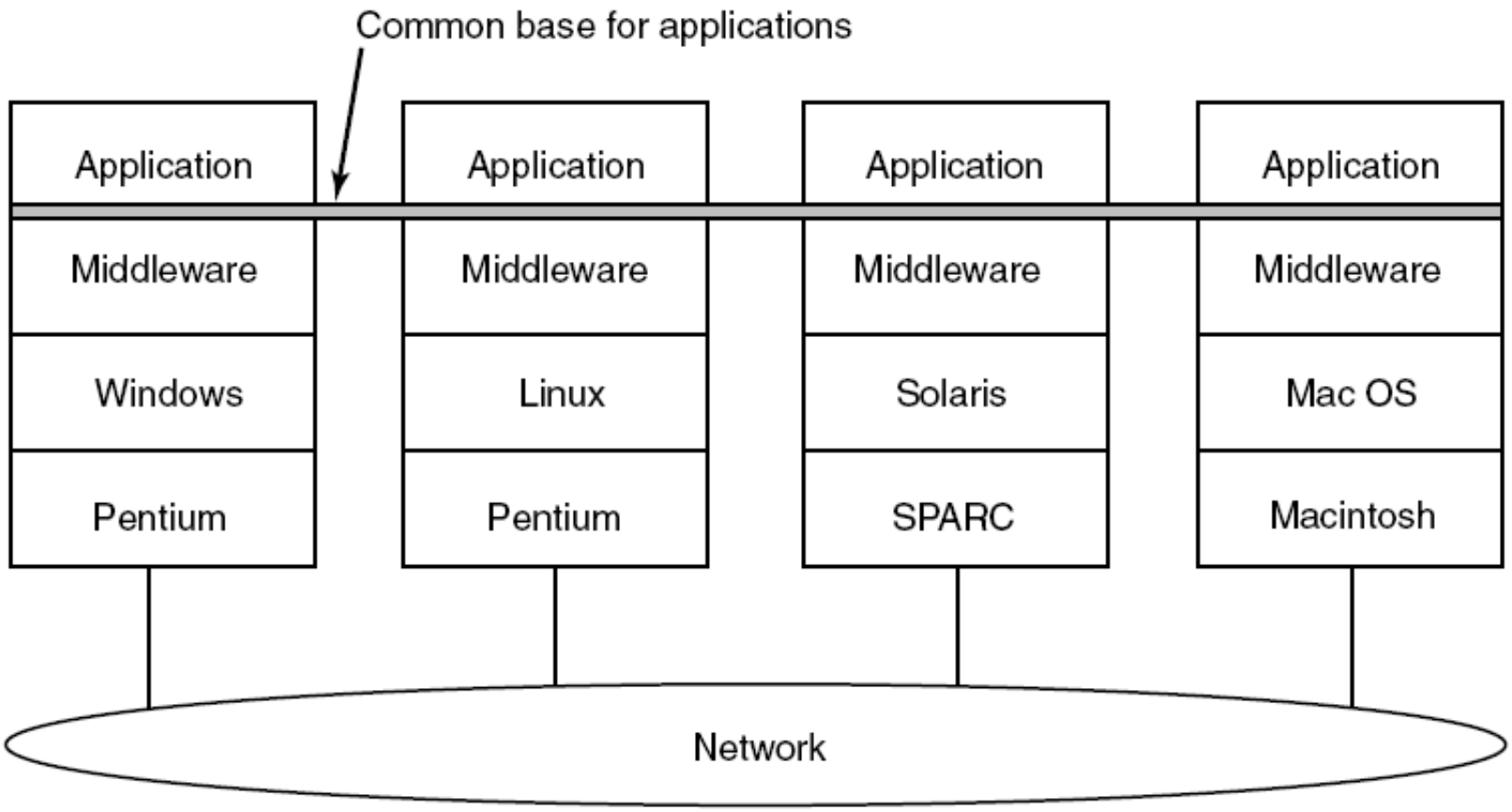


Figure 8-30. Positioning of middleware in a distributed system.

Ethernet

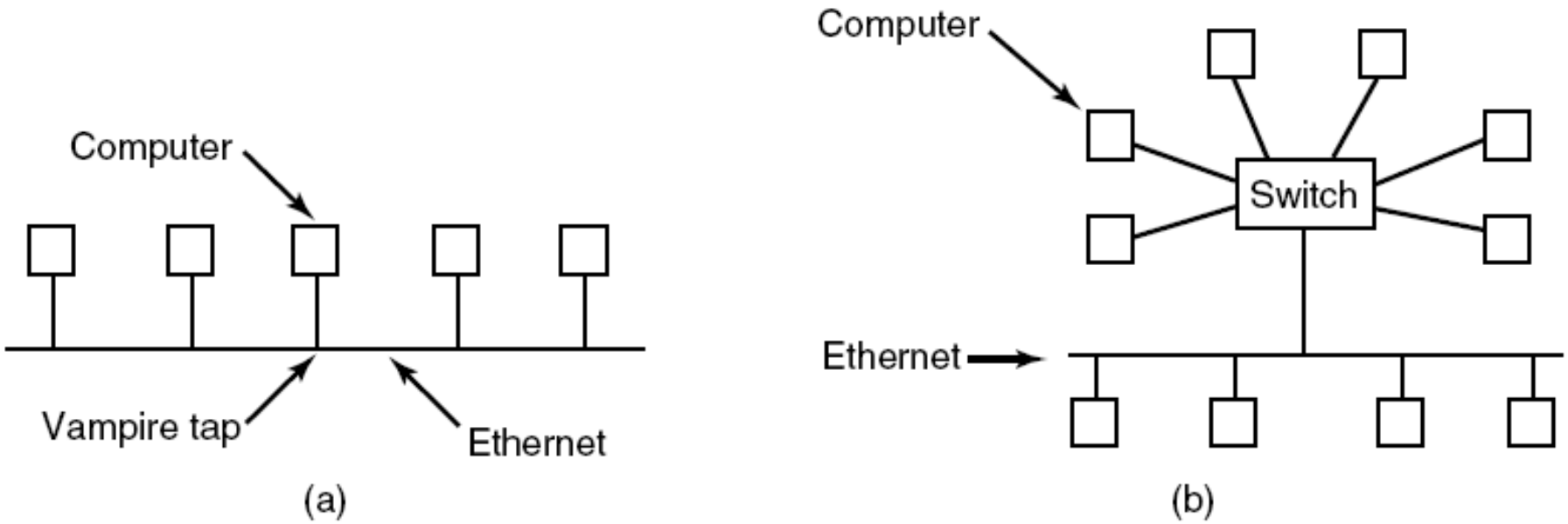


Figure 8-31. (a) Classic Ethernet. (b) Switched Ethernet.

The Internet

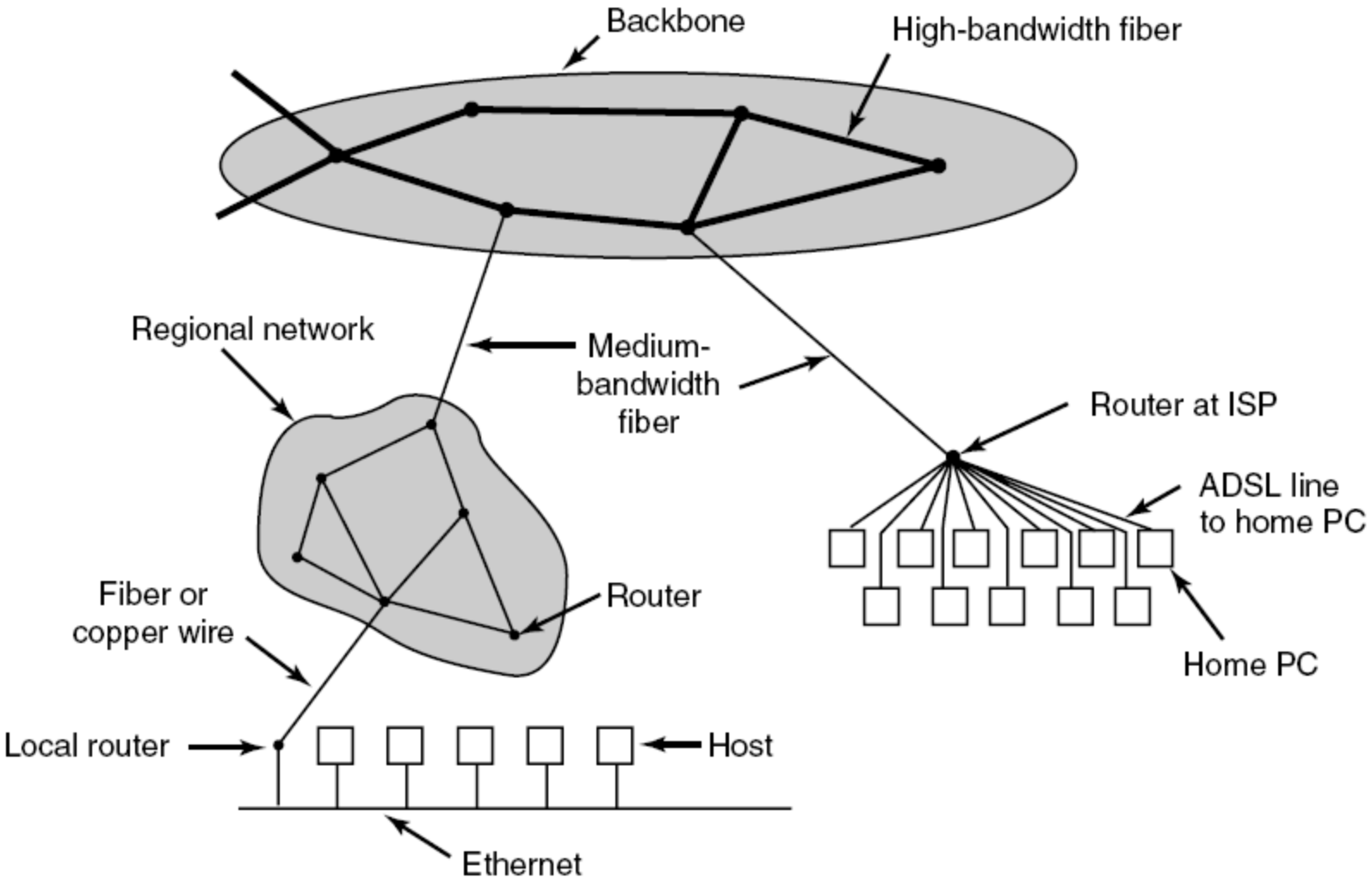


Figure 8-32. A portion of the Internet.

Network Protocols (1)

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages of a book
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connectionless	Unreliable datagram	Network test packets
	Acknowledged datagram	Registered mail
	Request-reply	Database query

Figure 8-33. Six different types of network service.

Network Protocols (2)

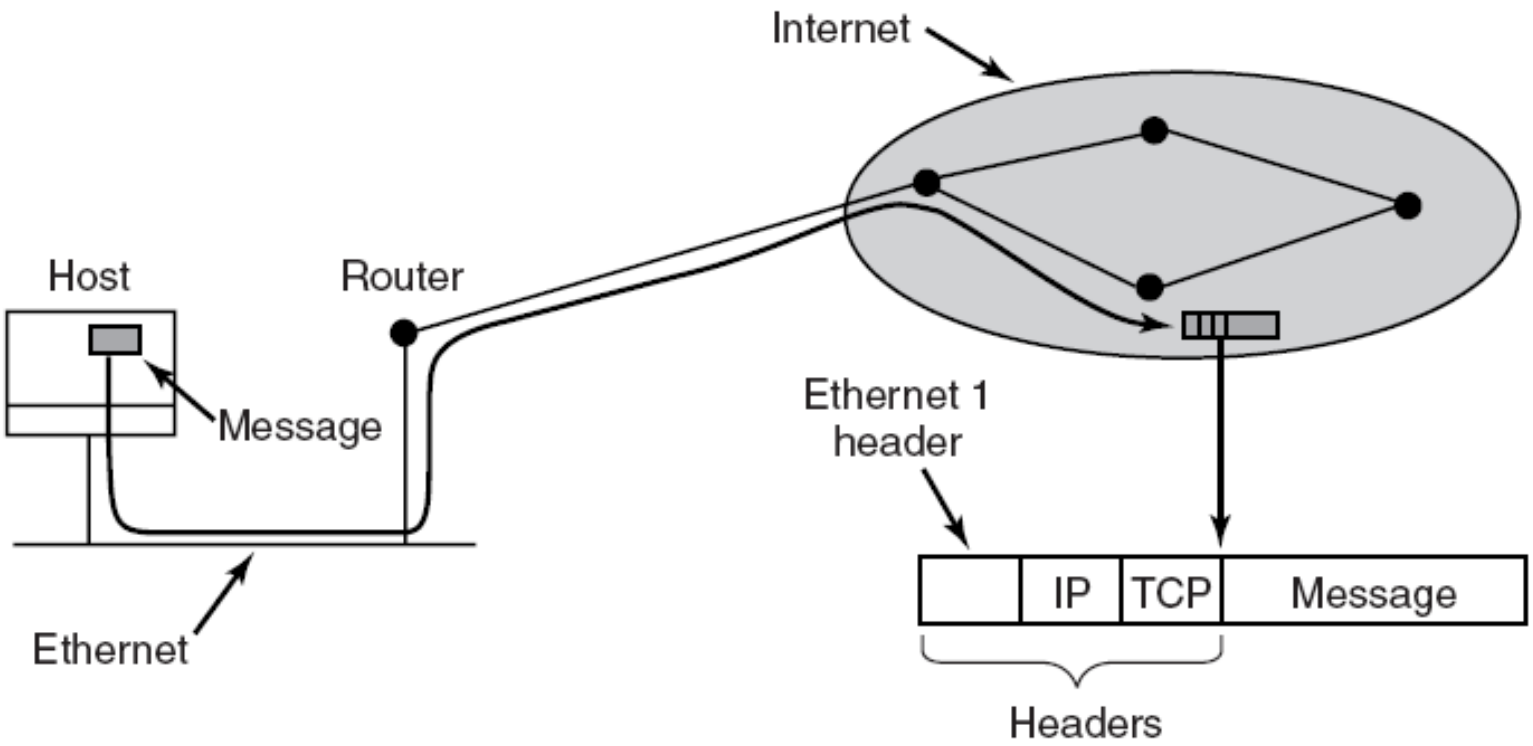


Figure 8-34. Accumulation of packet headers.

Document-Based Middleware (1)

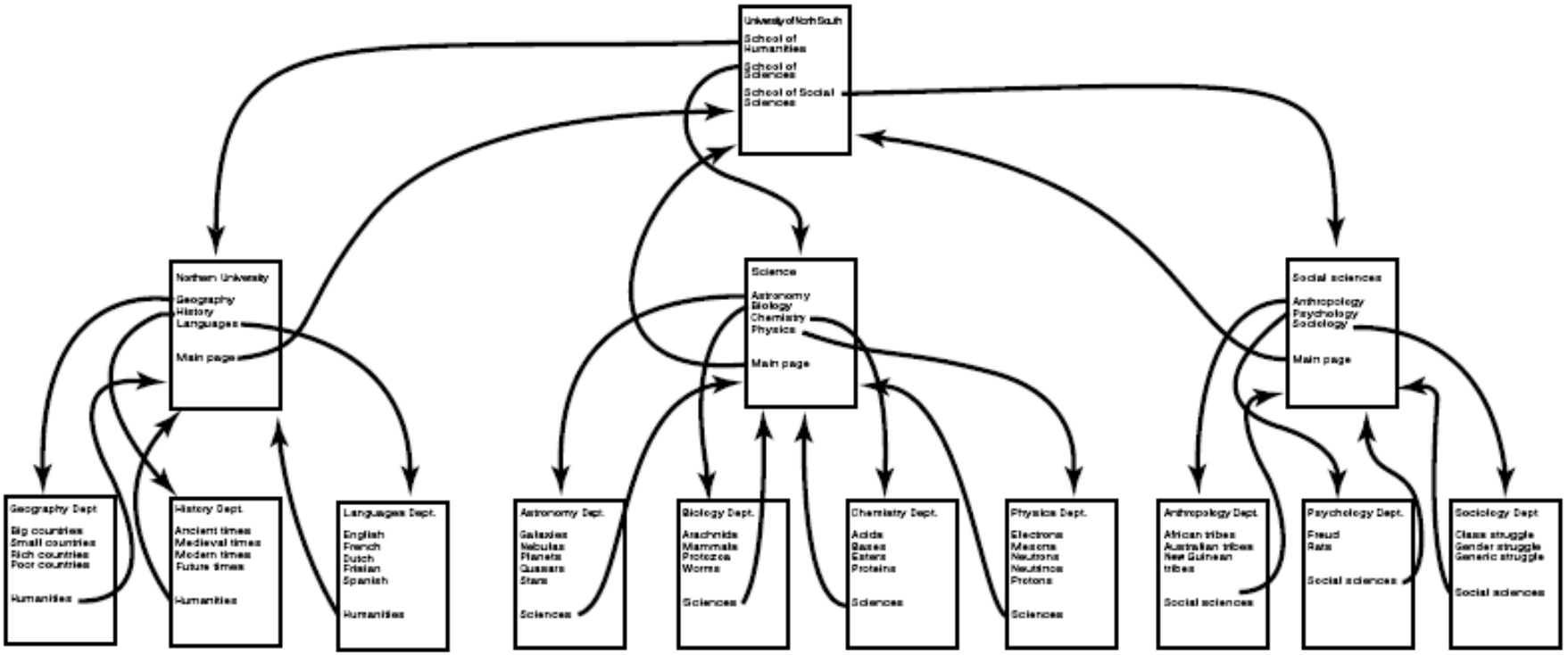


Figure 8-35. The Web is a big directed graph of documents.

Document-Based Middleware (2)

When the browser gets the page *http://www.minix3.org/doc/faq.html.*

1. *The browser asks DNS for the IP address of www.minix3.org.*
2. *DNS replies with 130.37.20.20.*
3. *The browser makes a TCP connection to port 80 on 130.37.20.20.*
4. *It then sends a request asking for the file doc/faq.html.*

. . .

Document-Based Middleware (3)

. . .

5. *The www.acm.org server sends the file doc/faq.html.*
6. *The TCP connection is released.*
7. *The browser displays all the text in doc/faq.html.*
8. *The browser fetches and displays all images in doc/faq.html.*

File-System-Based Middleware Transfer Model

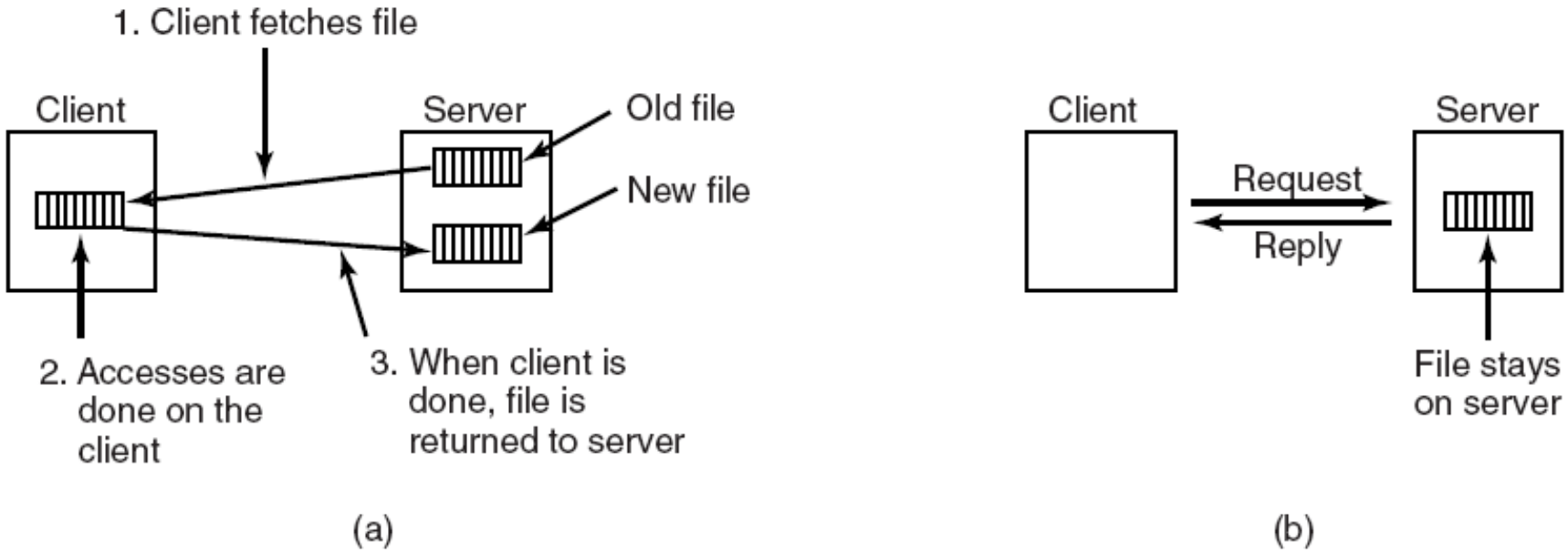


Figure 8-36. (a) The upload/download model. (b) The remote access model.

The Directory Hierarchy (1)

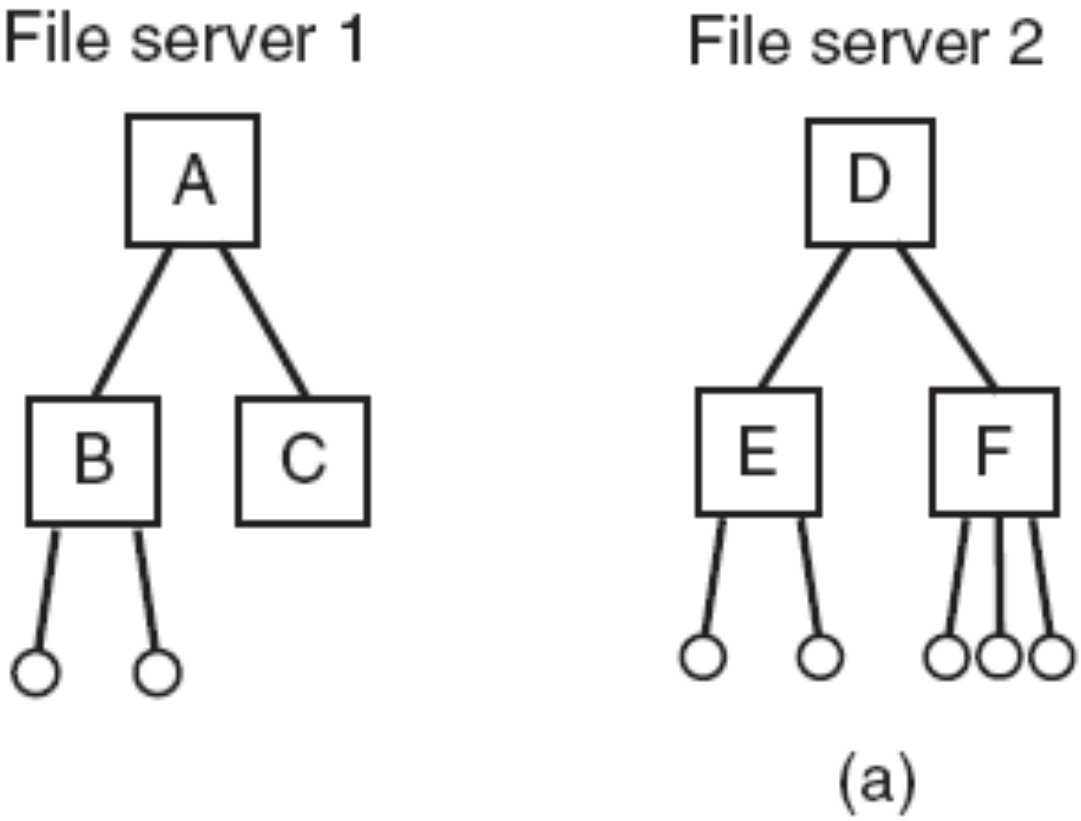


Figure 8-37. (a) Two file servers. The squares are directories and the circles are files.

The Directory Hierarchy (2)

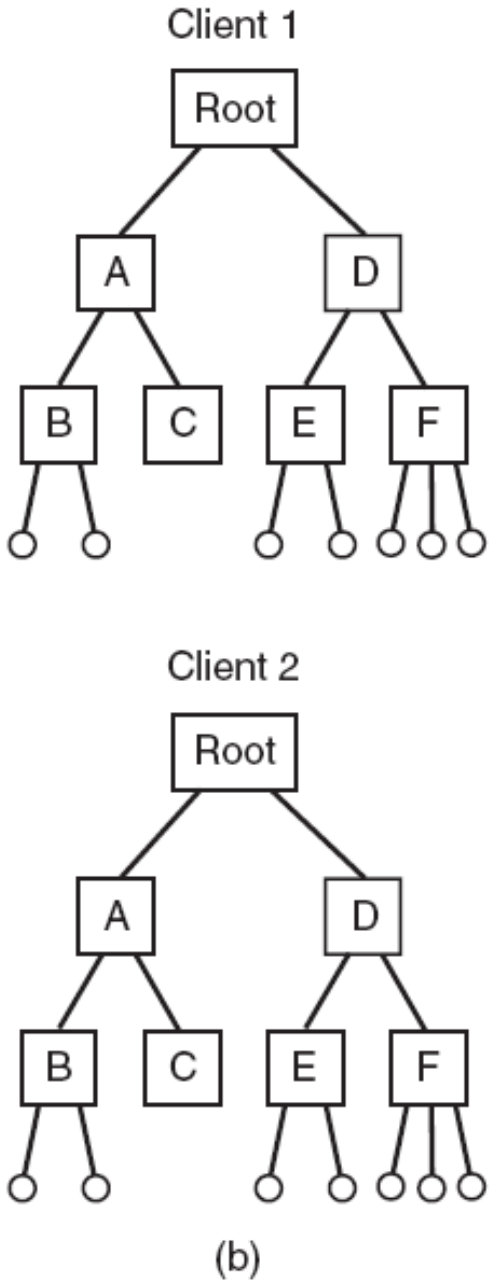


Figure 8-37. (b) A system in which all clients have the same view of the file system.

The Directory Hierarchy (3)

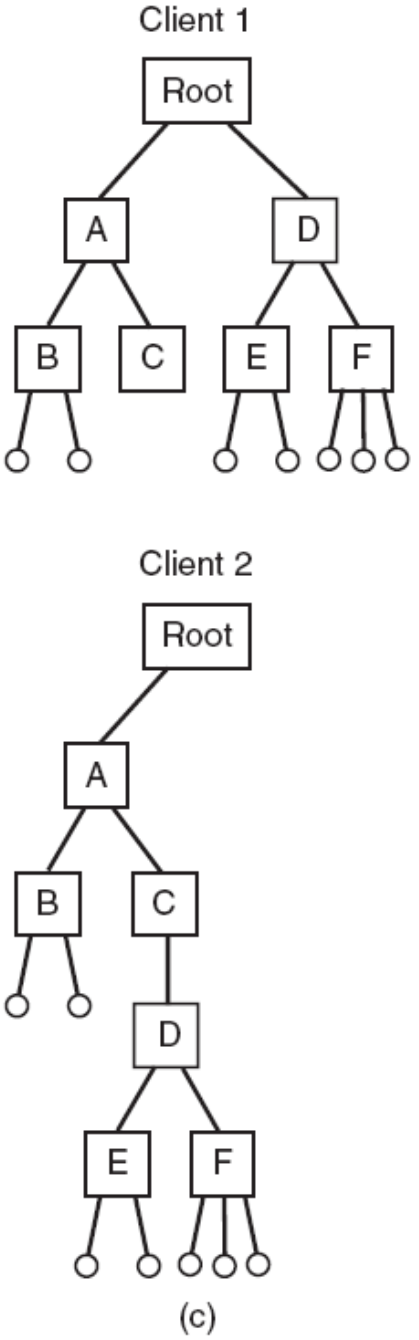


Figure 8-37. (c) A system in which different clients may have different views of the file system.

Naming Transparency

Three common approaches to file and directory naming in a distributed system:

1. Machine + path naming, such as */machine/path* or *machine:path*.
2. Mounting remote file systems onto the local file hierarchy.
3. A single name space that looks the same on all machines.

Semantics of File Sharing(1)

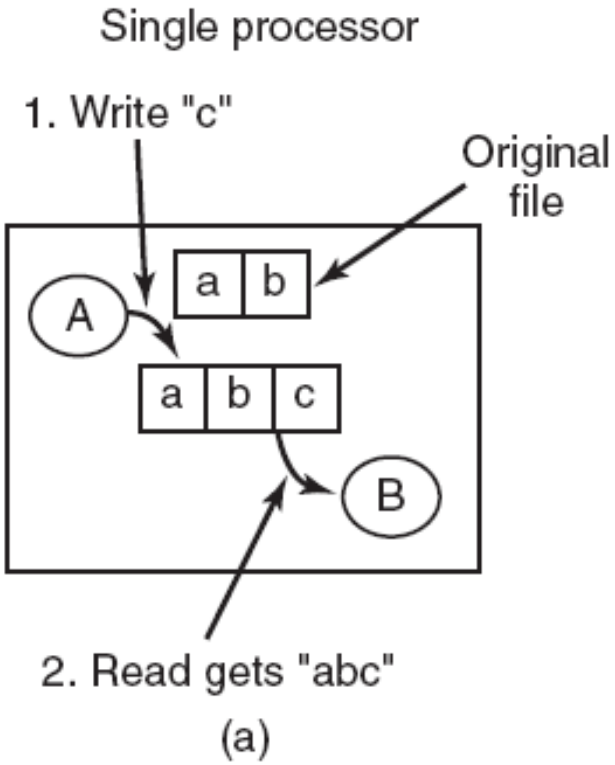


Figure 8-38. (a) Sequential consistency.

Semantics of File Sharing(2)

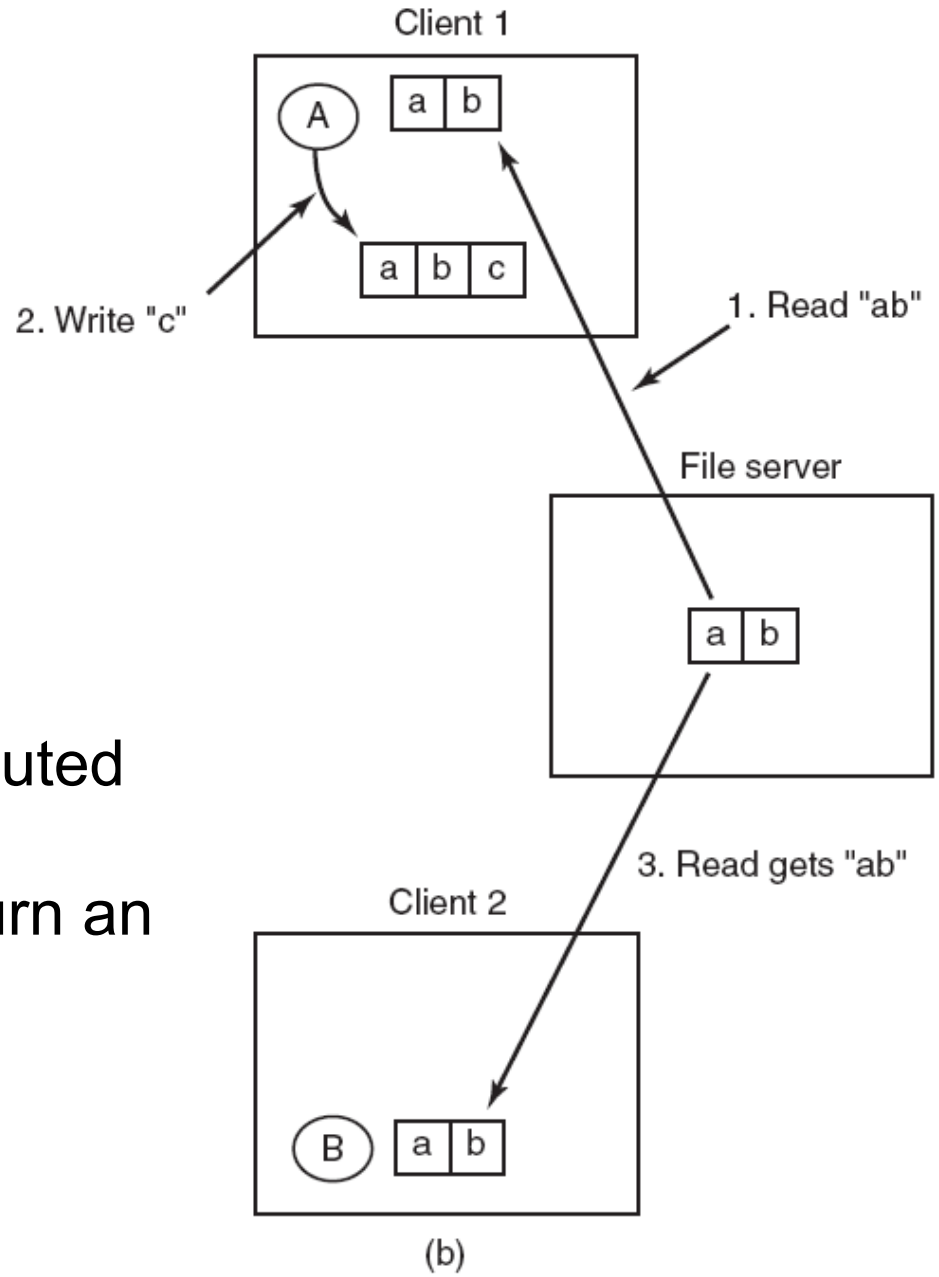


Figure 8-38. (b) In a distributed system with caching, reading a file may return an obsolete value.

Object-Based Middleware

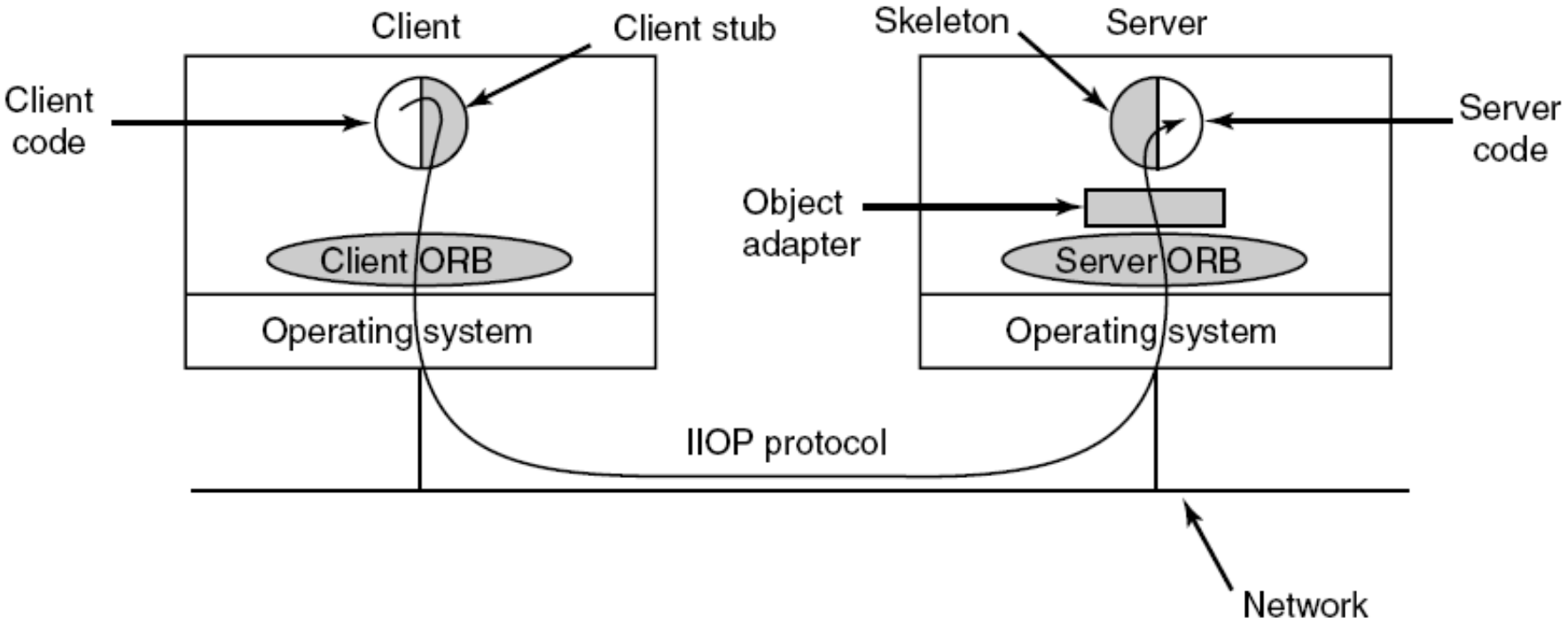


Figure 8-39. The main elements of a distributed system based on CORBA. The CORBA parts are shown in gray.

Coordination-Based Middleware (1)

Linda

- A system for communication and synchronization
- Independent processes communicate via an abstract tuple space
- A tuple is a structure of one or more fields, each of which is a value of some type supported by the base language

```
("abc", 2, 5)
("matrix-1", 1, 6, 3.14)
("family", "is-sister", "Stephany", "Roberta")
```

Figure 8-40. Three Linda tuples.

Matching Tuples

A match occurs if the following three conditions are all met:

1. The template and the tuple have the same number of fields.
2. The types of the corresponding fields are equal.
3. Each constant or variable in the template matches its tuple field.

Publish/Subscribe

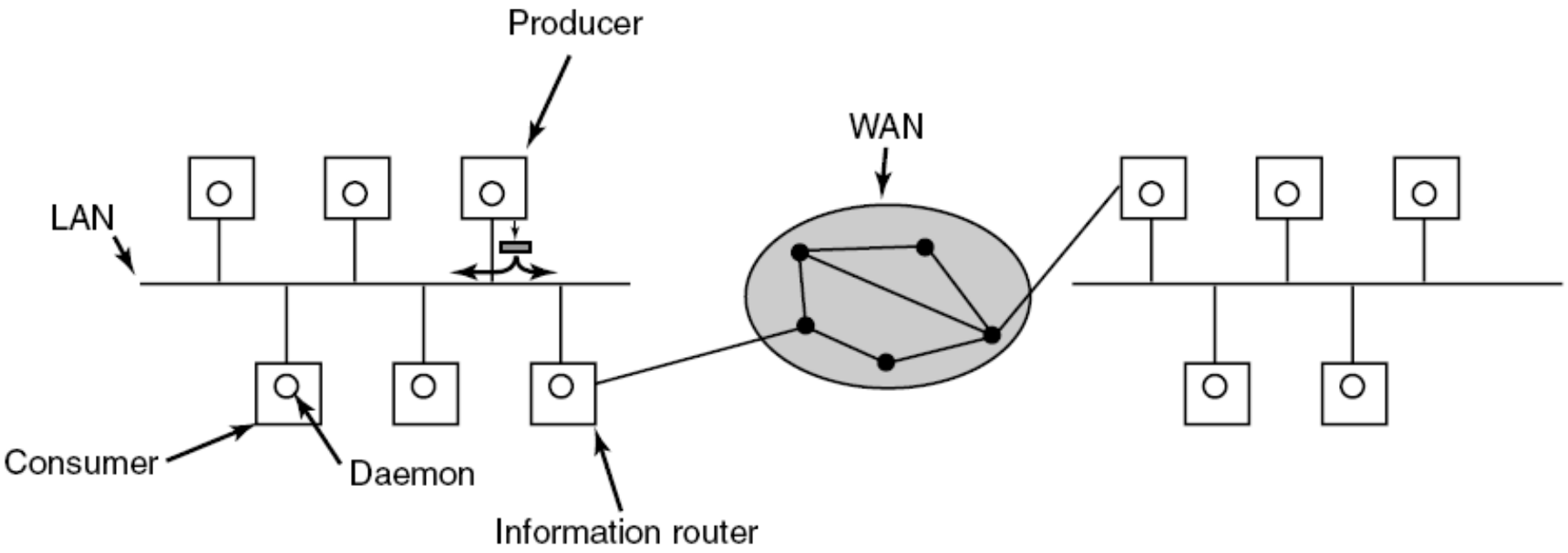


Figure 8-41. The publish/subscribe architecture.

Jini

Jini clients and services communicate and synchronize using JavaSpaces.

Methods defined in a JavaSpace:

1. **Write**: put a new entry into the JavaSpace.
2. **Read**: copy an entry that matches a template out of the JavaSpace.
3. **Take**: copy and remove an entry that matches a template.
4. **Notify**: notify the caller when a matching entry is written.