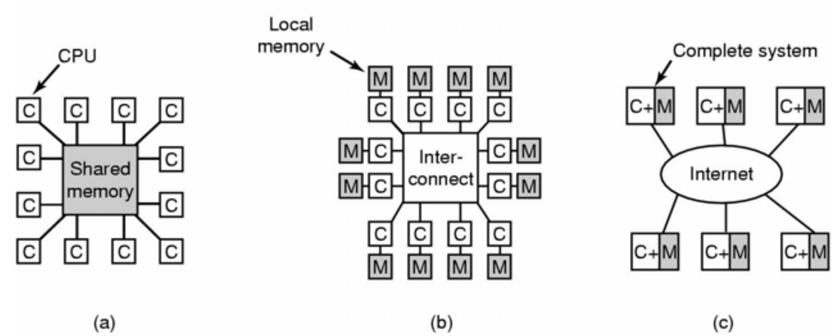
Chapter 8

Multiple Processor Systems

Multiprocessor Systems



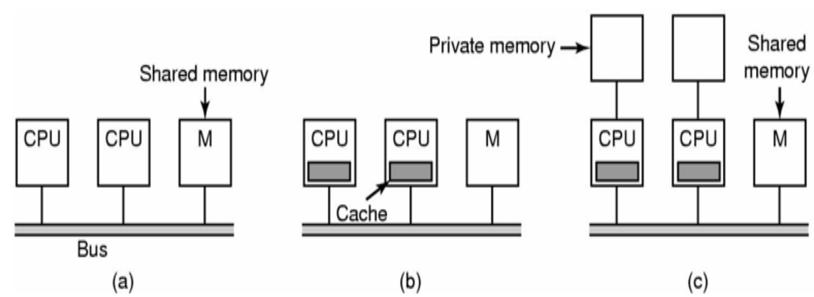
- Continuous need for faster computers
  - shared memory model
  - message passing multiprocessor
  - wide area distributed system

Multiprocessors

Definition:

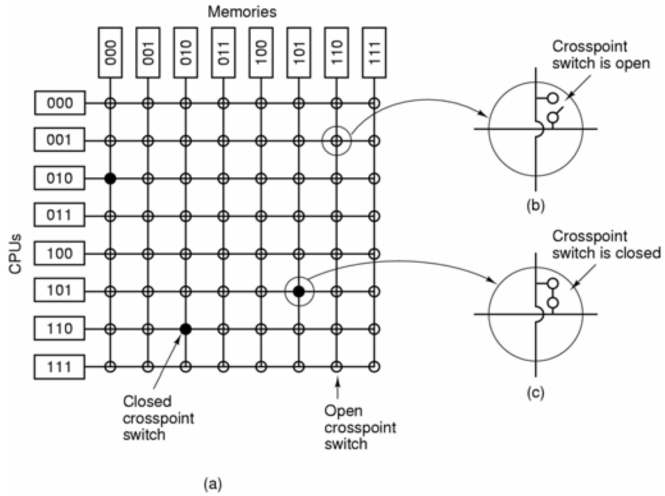
A computer system in which two or more CPUs share full access to a common RAM

Multiprocessor Hardware (1)



Bus-based multiprocessors

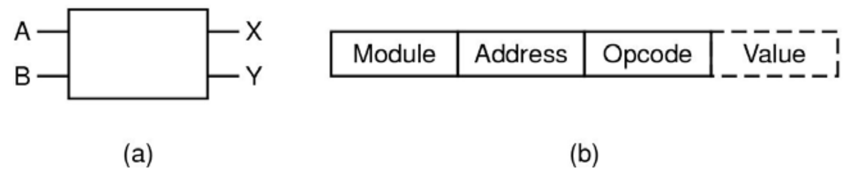
Multiprocessor Hardware (2)



- UMA Multiprocessor using a crossbar switch

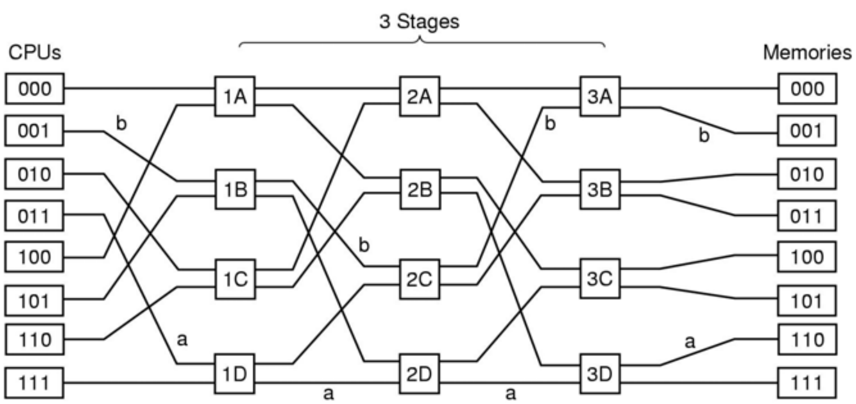
Multiprocessor Hardware (3)

- UMA multiprocessors using multistage switching networks can be built from 2x2 switches



(a) 2x2 switch (b) Message format

Multiprocessor Hardware (4)



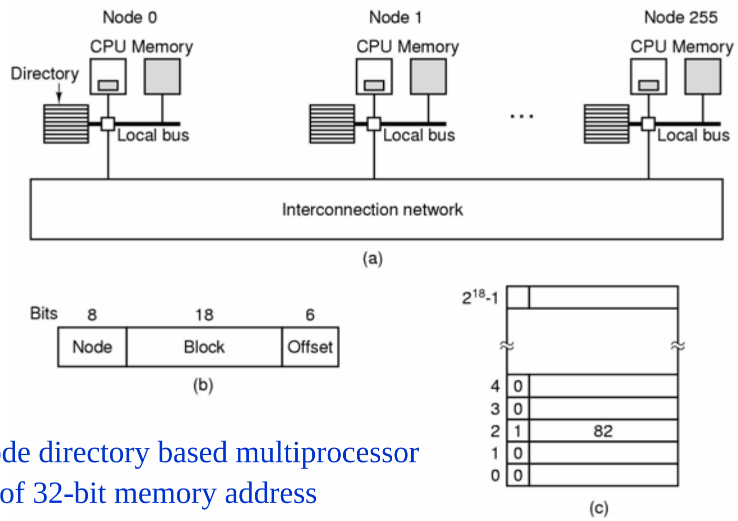
- Omega Switching Network

Multiprocessor Hardware (5)

NUMA Multiprocessor Characteristics

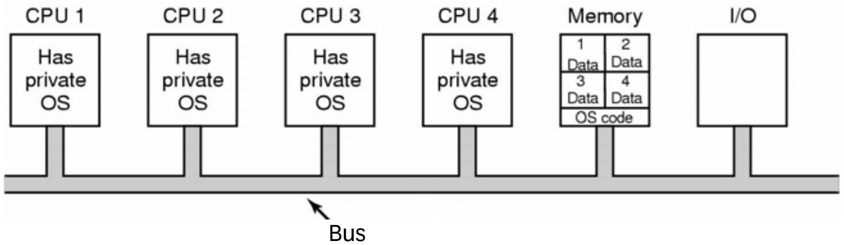
1. Single address space visible to all CPUs
2. Access to remote memory via commands
   - LOAD
   - STORE
3. Access to remote memory slower than to local

Multiprocessor Hardware (6)



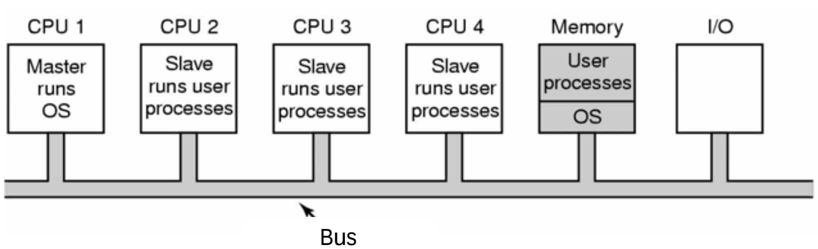
(a) 256-node directory based multiprocessor
(b) Fields of 32-bit memory address
(c) Directory at node 36

Multiprocessor OS Types (1)



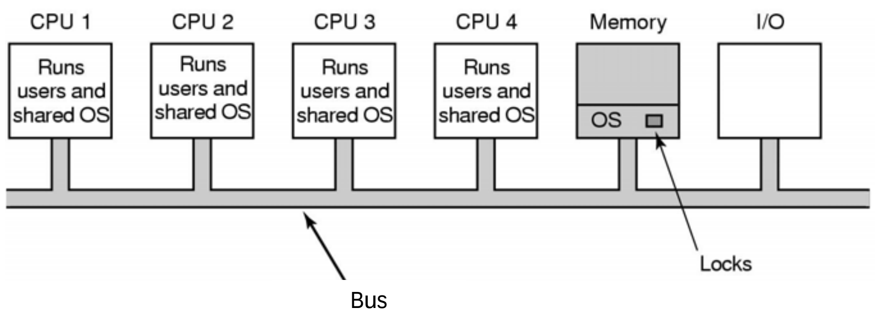
Each CPU has its own operating system

Multiprocessor OS Types (2)



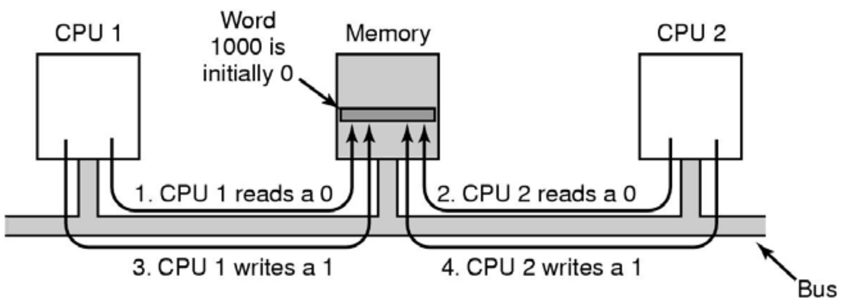
Master-Slave multiprocessors

Multiprocessor OS Types (3)



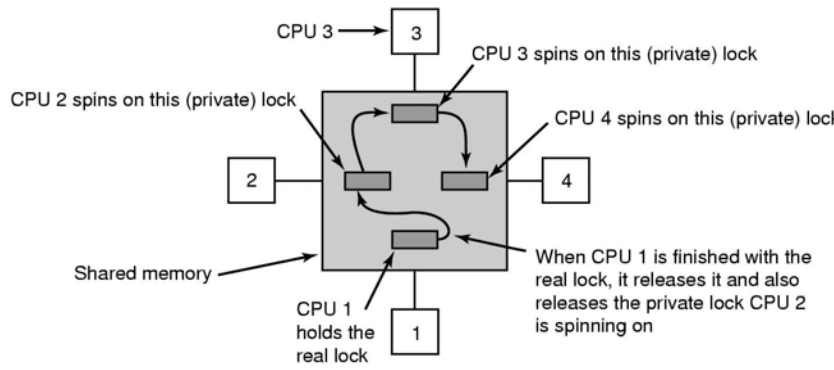
- Symmetric Multiprocessors
  - SMP multiprocessor model

Multiprocessor Synchronization (1)



TSL instruction can fail if bus already locked

Multiprocessor Synchronization (2)



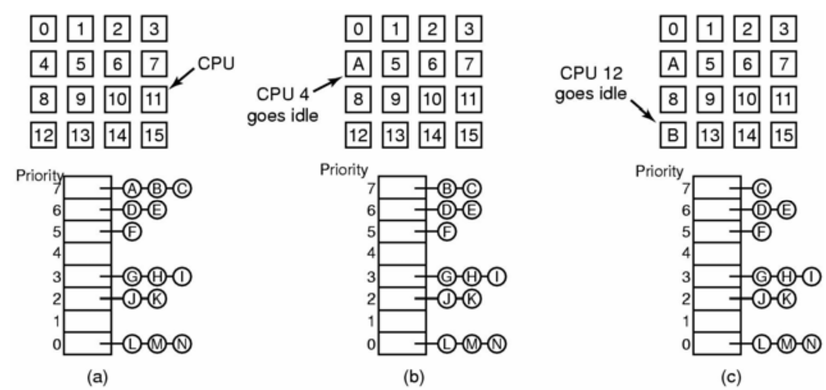
Multiple locks used to avoid cache thrashing

Multiprocessor Synchronization (3)

Spinning versus Switching

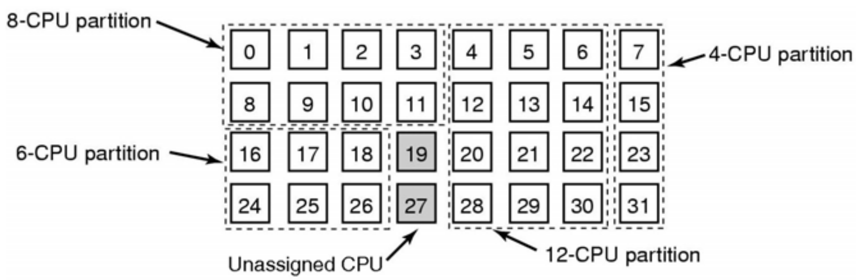
- In some cases CPU must wait
  - waits to acquire ready list
- In other cases a choice exists
  - spinning wastes CPU cycles
  - switching uses up CPU cycles also
  - possible to make separate decision each time locked mutex encountered

Multiprocessor Scheduling (1)



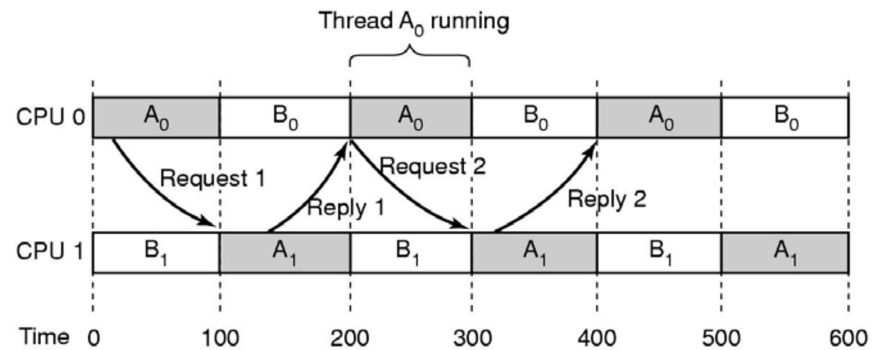
- Timesharing
  - note use of single data structure for scheduling

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

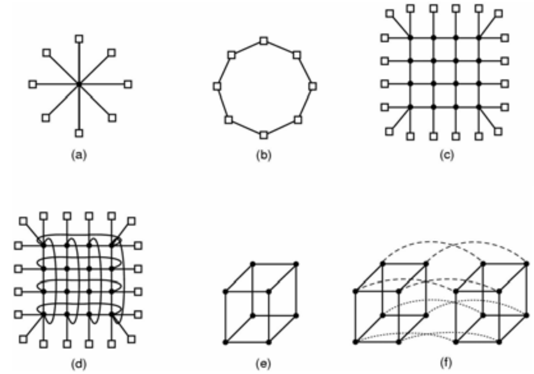
Gang Scheduling

Multicomputers

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

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

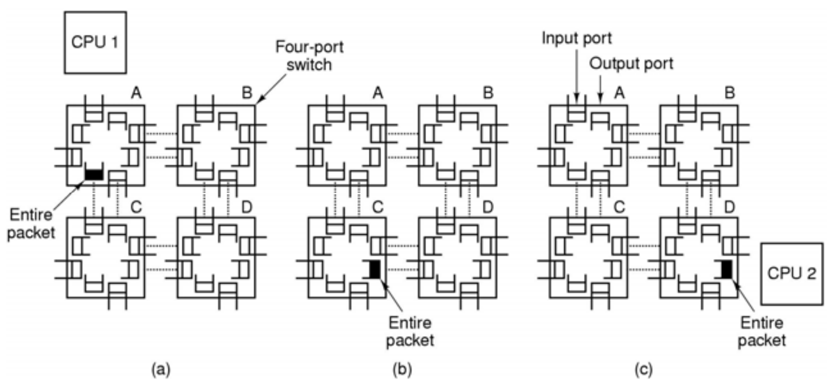
Multicomputer Hardware (1)



- Interconnection topologies

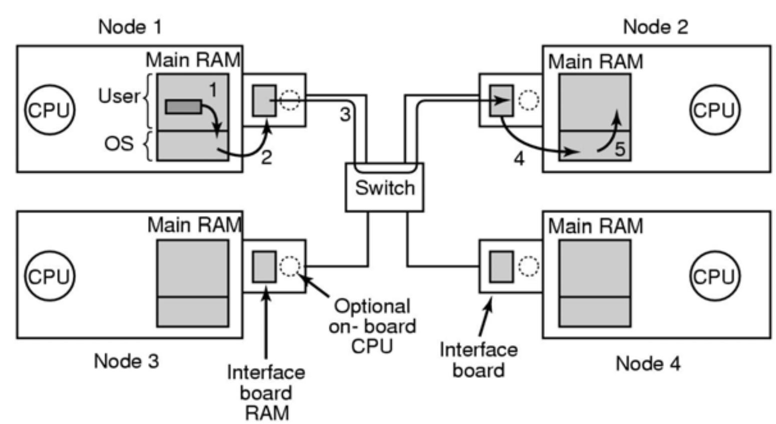
(a) single switch
(b) ring
(c) grid
(d) double torus
(e) cube
(f) hypercube

Multicomputer Hardware (2)



- Switching scheme
  - store-and-forward packet switching

Multicomputer Hardware (3)



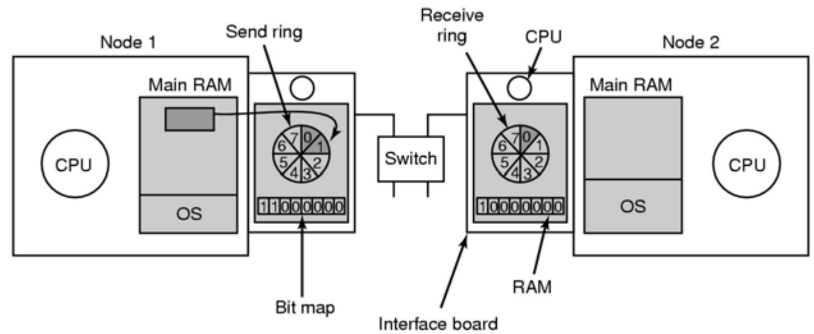
Network interface boards in a multicomputer

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

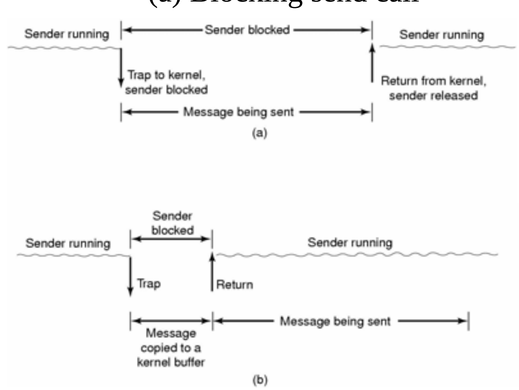
- Use send & receive rings
- coordinates main CPU with on-board CPU

User Level Communication Software

- Minimum services provided
  - send and receive commands

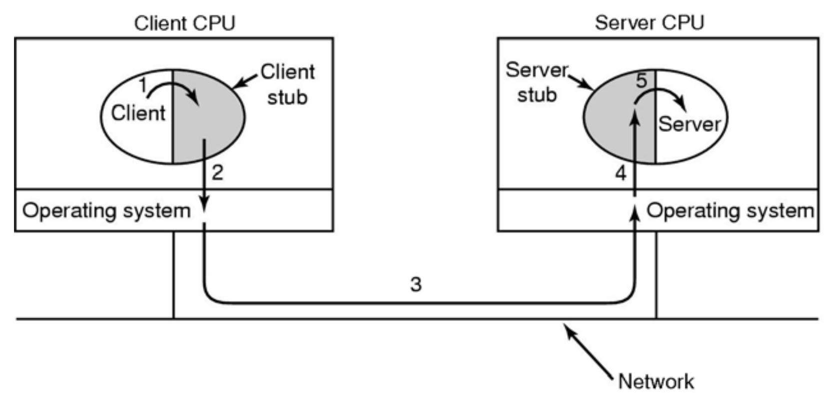
- These are blocking (synchronous) calls

(a) Blocking send call



(b) Nonblocking send call

Remote Procedure Call (1)



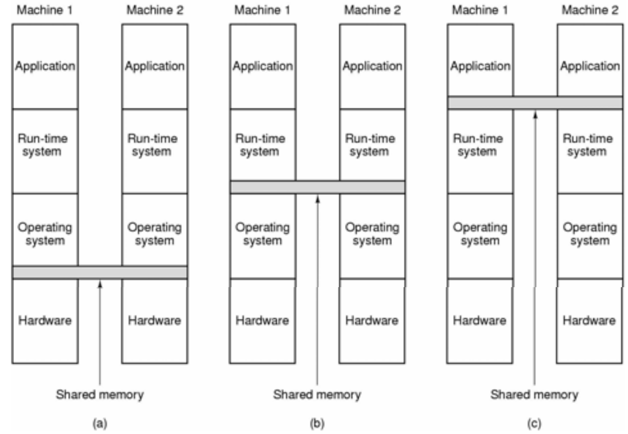
- Steps in making a remote procedure call
  - the stubs are shaded gray

Remote Procedure Call (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
  - user-level software

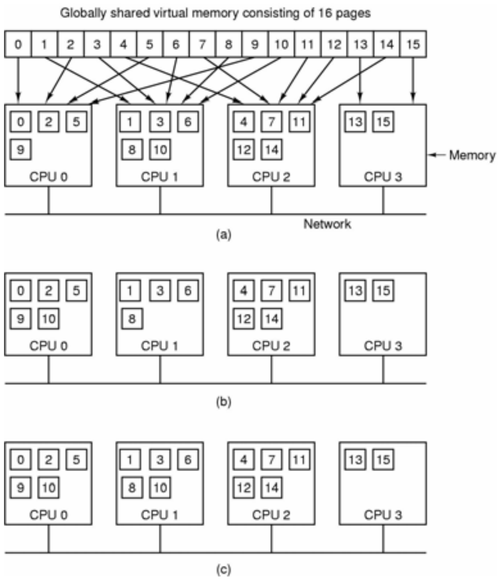
Distributed Shared Memory (2)

Replication

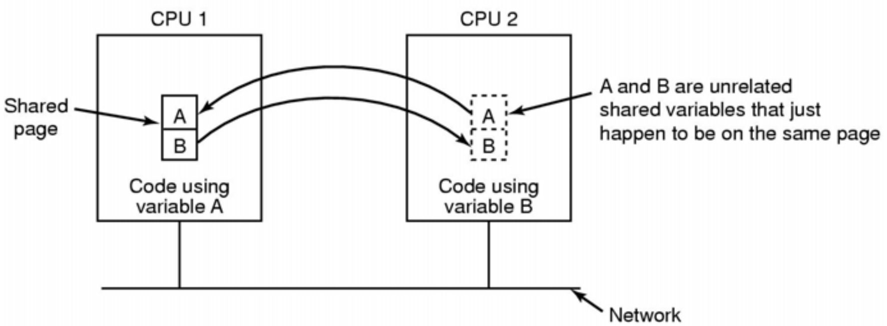
(a) Pages distributed on 4 machines

(b) CPU 0 reads page 10

(c) CPU 1 reads page 10

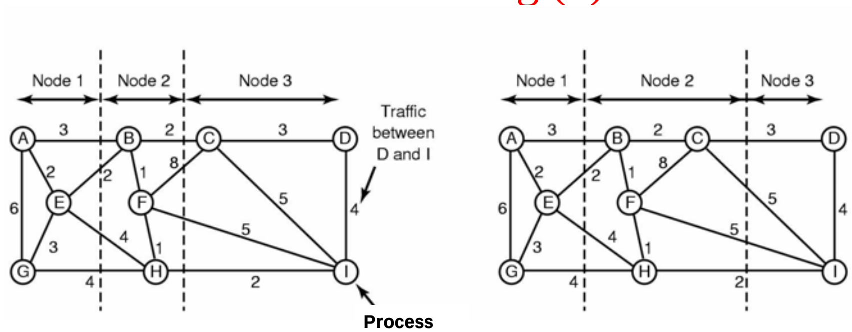


Distributed Shared Memory (3)



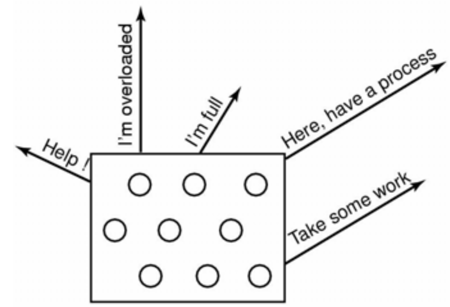
- False Sharing
- Must also achieve sequential consistency

Multicomputer Scheduling
Load Balancing (1)



- Graph-theoretic deterministic algorithm

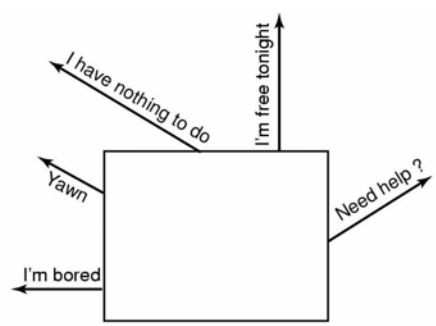
Load Balancing (2)



(a)

- Sender-initiated distributed heuristic algorithm
  – overloaded sender

Load Balancing (3)



(b)

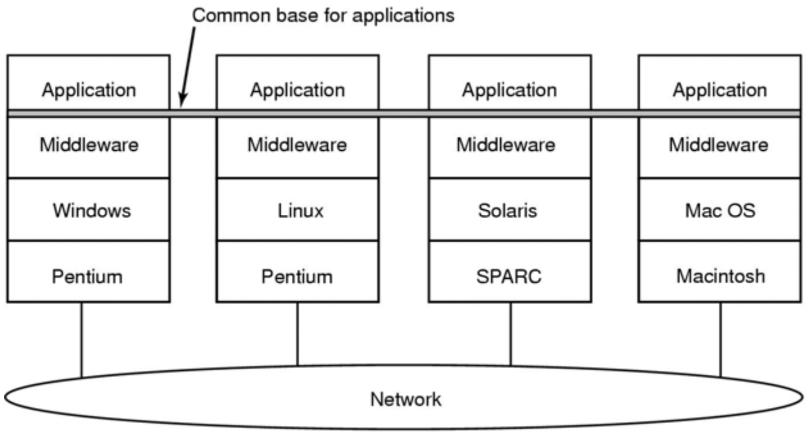
- Receiver-initiated distributed heuristic algorithm
  – under loaded receiver

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

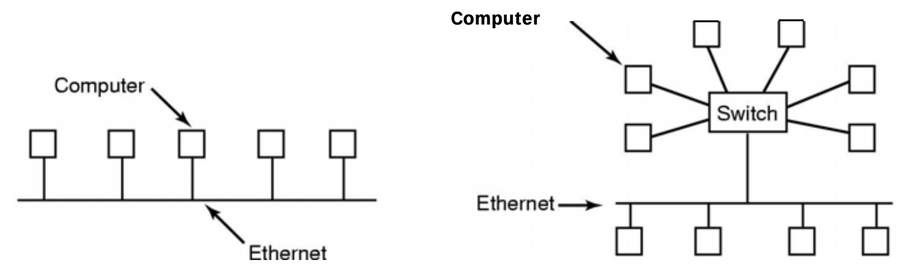
Comparison of three kinds of multiple CPU systems

Distributed Systems (2)



Achieving uniformity with middleware

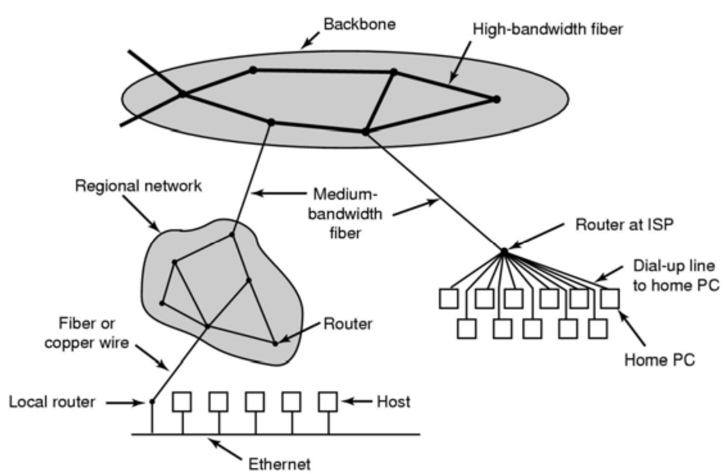
Network Hardware (1)



(a) (b)

- Ethernet
  - (a) classic Ethernet
  - (b) switched Ethernet

Network Hardware (2)



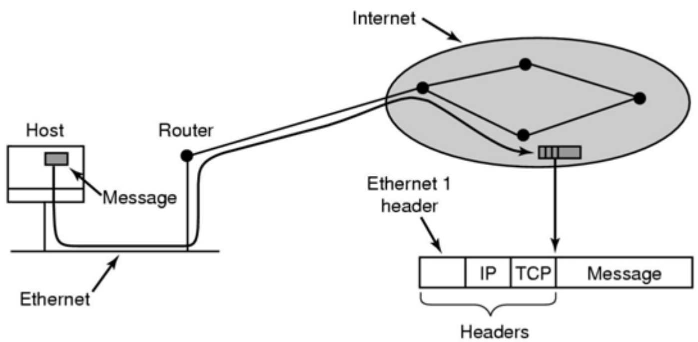
The Internet

Network Services and 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

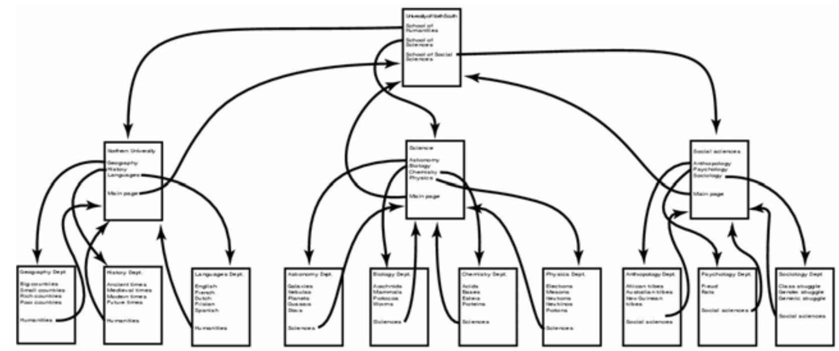
Network Services

Network Services and Protocols (2)



- Internet Protocol
- Transmission Control Protocol
- Interaction of protocols

Document-Based Middleware (1)



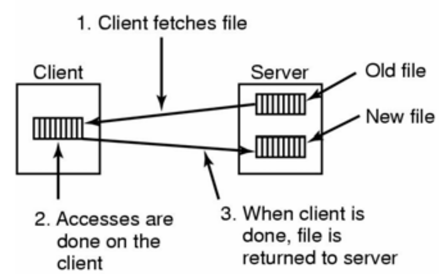
- The Web
  – a big directed graph of documents

Document-Based Middleware (2)

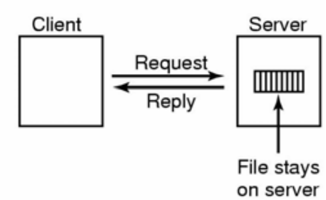
How the browser gets a page

1. Asks DNS for IP address
2. DNS replies with IP address
3. Browser makes connection
4. Sends request for specified page
5. Server sends file
6. TCP connection released
7. Browser displays text
8. Browser fetches, displays images

File System-Based Middleware (1)



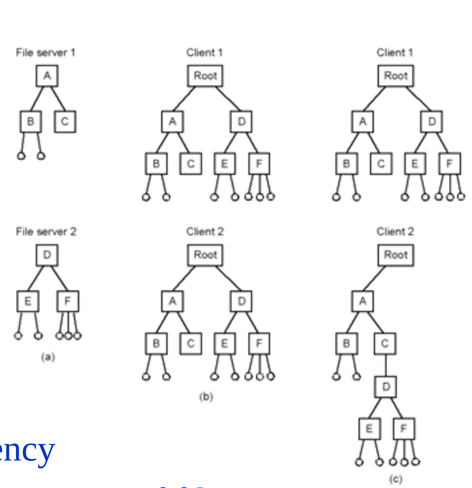
(a)



(b)

- Transfer Models
  (a) upload/download model
  (b) remote access model

File System-Based Middleware (2)

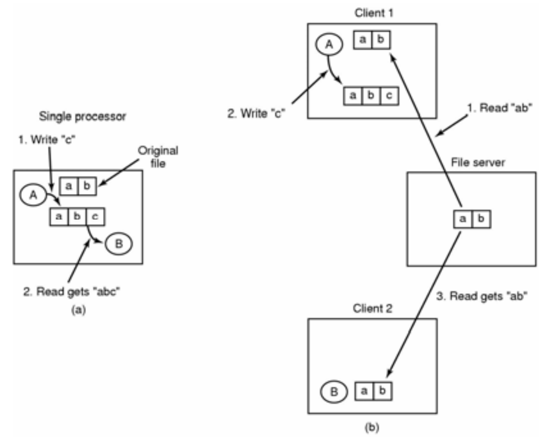


Naming Transparency

(b) Clients have same view of file system

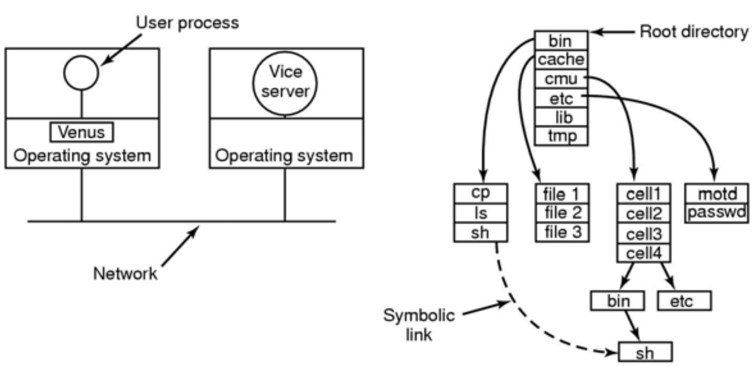
(c) Alternatively, clients with different view

File System-Based Middleware (3)



- Semantics of File sharing
  - (a) single processor gives sequential consistency
  - (b) distributed system may return obsolete value

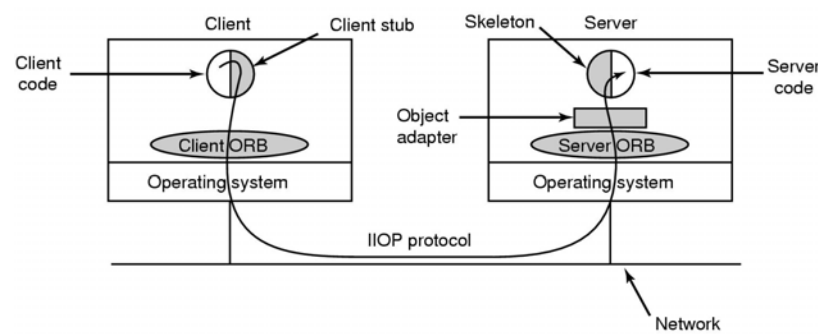
File System-Based Middleware (4)



Client's view

- AFS – Andrew File System
  - workstations grouped into cells
  - note position of venus and vice

Shared Object-Based Middleware (1)

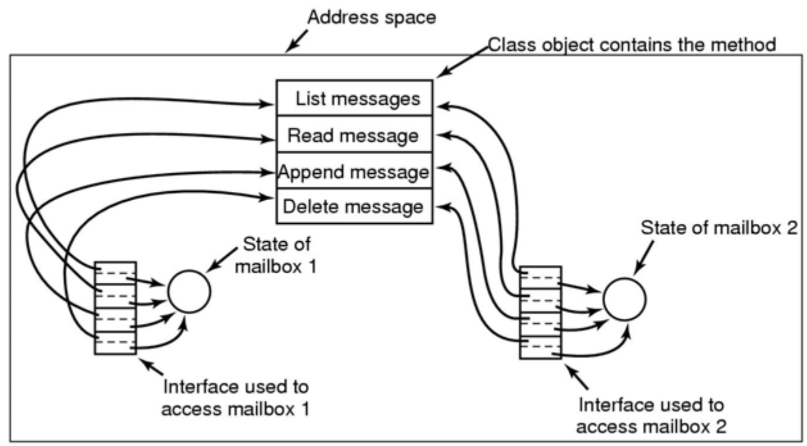


- Main elements of CORBA based system
  - Common Object Request Broker Architecture

Shared Object-Based Middleware (2)

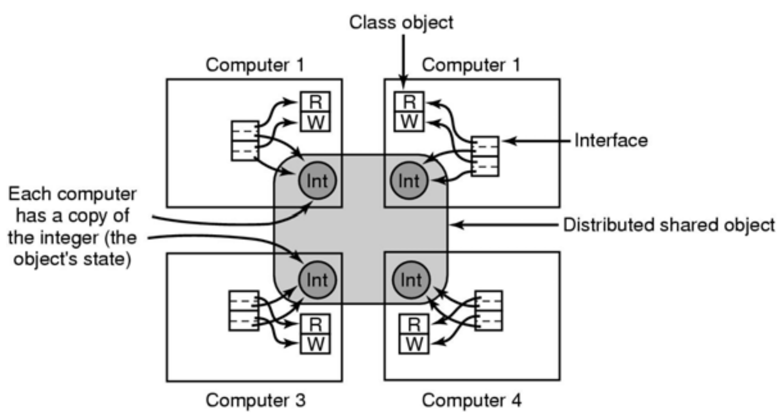
- Scaling to large systems
  – replicated objects
  – flexibility
- Globe
  – designed to scale to a billion users
  – a trillion objects around the world

Shared Object-Based Middleware (3)



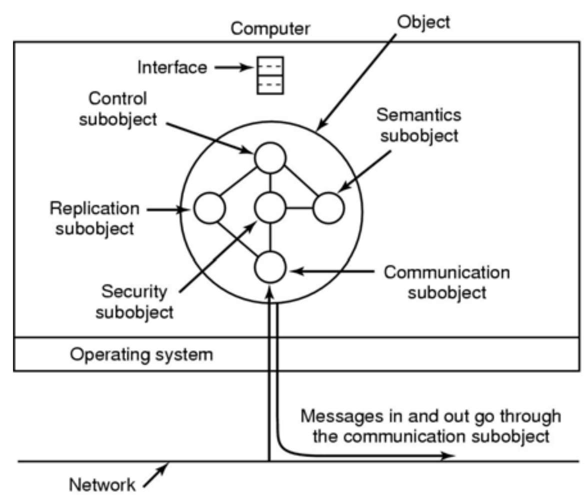
Globe structured object

Shared Object-Based Middleware (4)



- A distributed shared object in Globe
  – can have its state copied on multiple computers at once

Shared Object-Based Middleware (5)



Internal structure of a Globe object

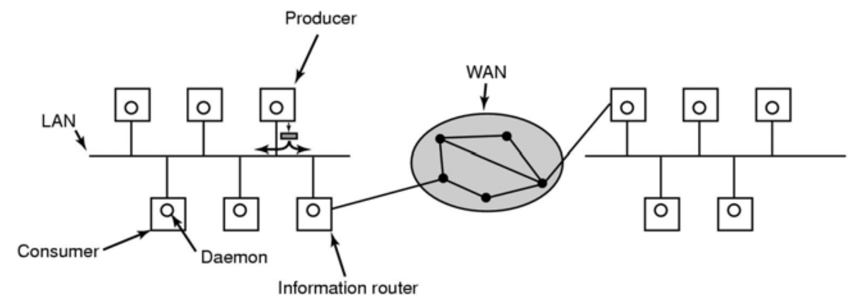
Coordination-Based Middleware (1)

- Linda
  - independent processes
  - communicate via abstract tuple space
- Tuple
  - like a structure in C, record in Pascal

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

1. Operations: out, in, read, eval

Coordination-Based Middleware (2)



Publish-Subscribe architecture

Coordination-Based Middleware (3)

- Jini - based on Linda model
  - devices plugged into a network
  - offer, use services
- Jini Methods
  1. read
  2. write
  3. take
  4. notify