# Operating Systems

# Multiple Processor Systems

dr. Tomasz Jordan Kruk

T.Kruk@ia.pw.edu.pl

Institute of Control & Computation Engineering
Warsaw University of Technology

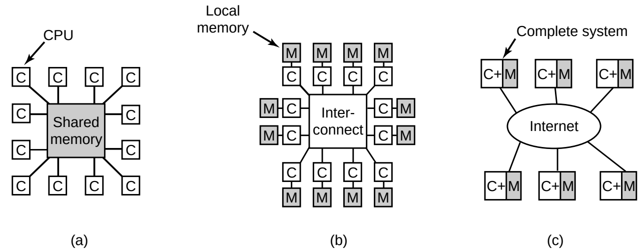


# Multiple Processor Systems

- √ is contemporary processing power huge enough to resolve all research/everyday problems?
- √ how scalable are computer systems?
- √ what is better: connected autonomous systems or many processors with shared memory?



# Organization of Multiple Processors



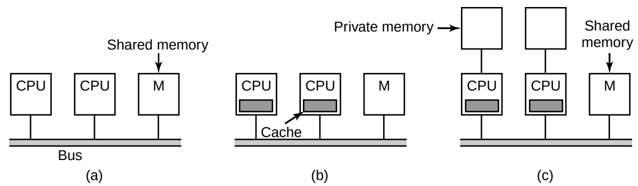

Organization of Multiple Processors

a. **1. A shared-memory multiprocessor**,
b. **2. A message-passing multicomputer** – tightly-coupled systems,
c. **3. A wide area distributed system** – loosely-coupled systems.



# 1. Shared-memory Multiprocessors

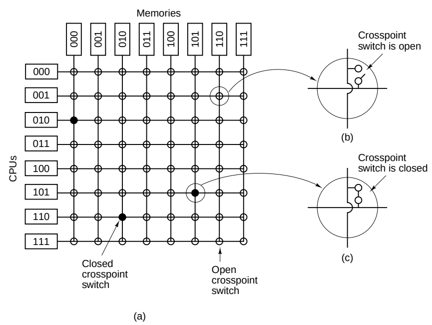
**UMA** (uniform memory access) and **NUMA** (non uniform memory access) systems may be distinguished.




UMA bus-based SMP architectures:

a. without caching,
b. with caching,
c. with caching and private memories.

# UMA Multiprocessors Using Crossbar Switches

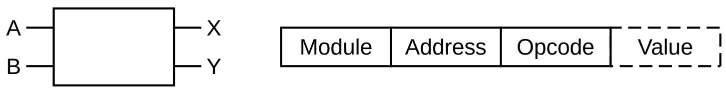



An 8 x 8 crossbar switch.

- √ the biggest advantage: nonblocking crossbar,
- √ the biggest drawback: cost of $n^2$ for $n$ processors.



# Multistage Switching Networks




(a) (b)

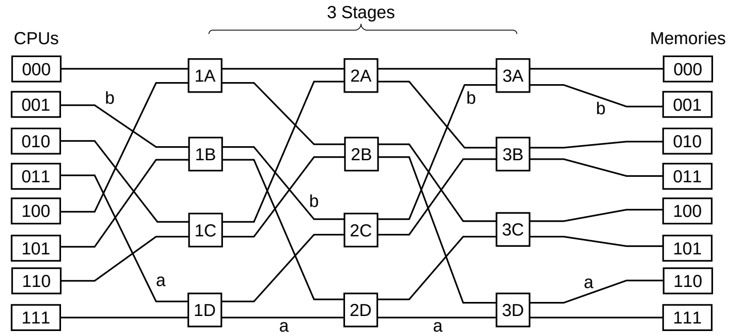
UMA multiprocessors using multistage switching networks.

a. a 2 x 2 switch,
b. a message format.

- √ for $n$ processors and $n$ memory modules $\log_2 n$ stages with $n/2$ switches in each stage is required,
- √ $(n/2)\log_2 n << n^2$



# An Omega Switching Network




- √ routing due to address bits values,
- √ conficts possible forcing retransmission,
- √ interleaved memory system with routing based on low-order bits.



# NUMA Multiprocessors

**Idea**: with cost of different access times to different memory modules it is possible to run unmodified tasks on computers with bigger number of processors.

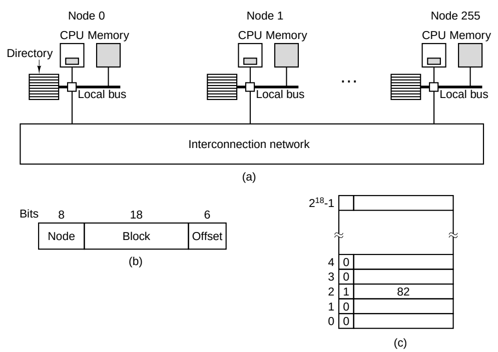
Characteristic features of the NUMA (*Non Uniform Memory Access*) architecture:

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.

**nc-NUMA** when the access time to remote memory is not hidden (because of no caching),

**cc-NUMA** when coherent caches are present (*cache coherent-NUMA*).

# Directory-based NUMA architecture

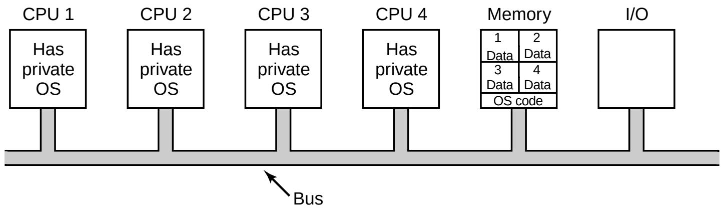



a. a 256-node directory-based multiprocessor,
b. division of a 32-bit memory address into fields,
c. the directory at node 36.



# Multiprocessor Operating System Types

1. Each CPU has its own operating system.

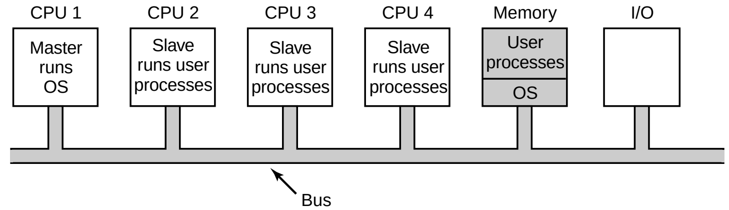



Partitioning multiprocessor memory among four CPUs, but sharing a single copy of the operating system code.



# Master-Slave Multiprocessors

2. A master-slave multiprocessor model



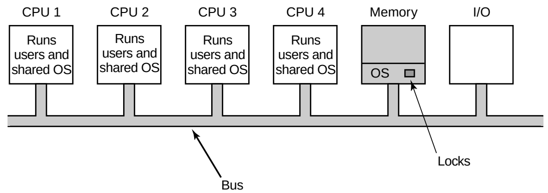

A master-slave multiprocessor model.

- √ single ready processes list,
- √ avoidance of overloading,
- √ the master is a bottleneck, solution not well scalable.



# Symmetric Multiprocessors (SMP)

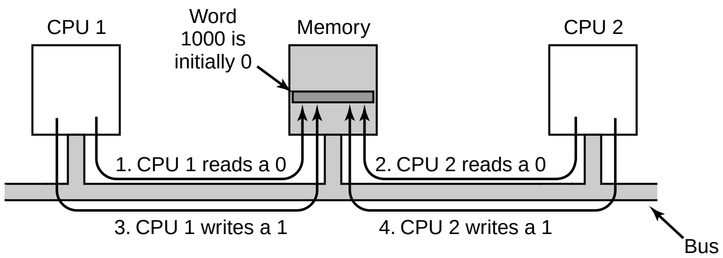
3. Symmetric Multiprocessors




The SMP multiprocessor model.

- √ all processors of equal importance,
- √ one copy of operating system which may be run by each processor,
- √ still some trouble with scalability,
- √ kernel must be divided into smaller critical regions, kernel must be reentrant,
- √ huge costs of synchronization.

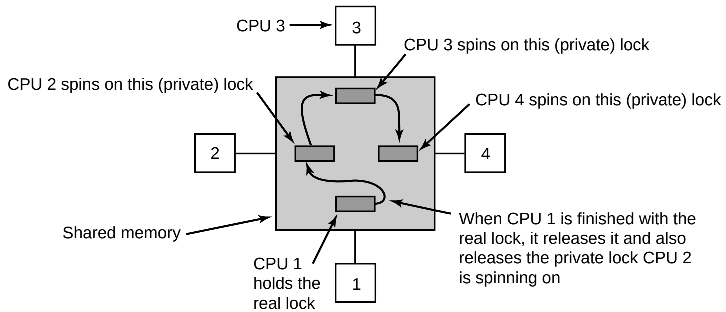
# Multiprocessor Synchronization




Fours steps leading to an error demonstrated. The TSL instruction may fail if the bus blocking fails. Blockiing of bus/crossbar is required.



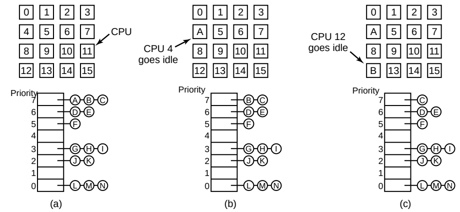
# Cache thrashing




Use of multiple locks to avoid cache thrashing.



# Multiprocessor Scheduling

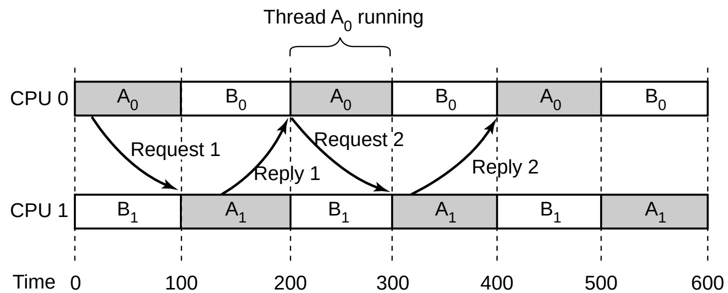



Using of a single data structure for scheduling on a multiprocessor.

- √ **affinity scheduling** - to make a serious effort to have a process run on the same CPU it ran on last time.
- √ **two-level scheduling** - created process assigned to a CPU and run rather on the same CPU. If a CPU has no process to run, it takes one from another one rather than goes idle.



# Gang Scheduling (I)




Communication between two threads belonging to process A that are running out of phase.

# Gang Scheduling (II)

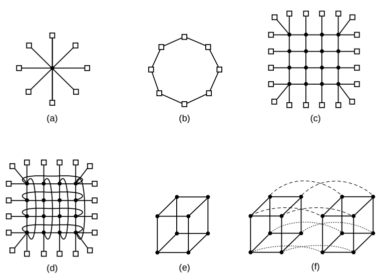
Idea of gang scheduling:

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

| Time slot \ CPU | 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 | $E_1$ | $E_2$ | $E_3$ | $E_4$ | $E_5$ | $E_6$ |



# 2. Multicomputers

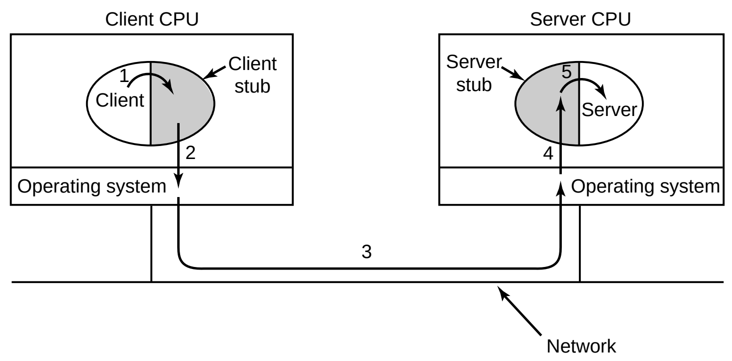



Various interconnect topologies:

a. a single switch,
b. a ring,
c. a grid,
d. a double torus,
e. a cube,
f. a 4D hybercube.



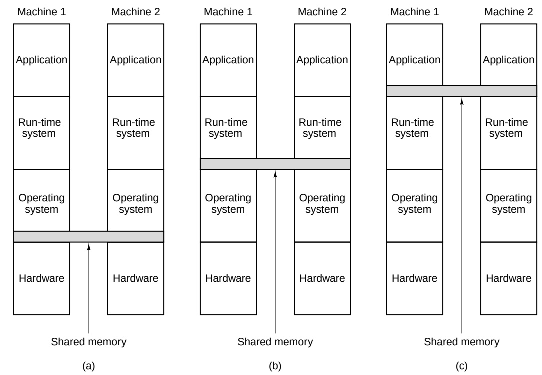
# Remote Procedure Call (RPC)




Steps in making a remote procedure call. The stubs are shaded gray.

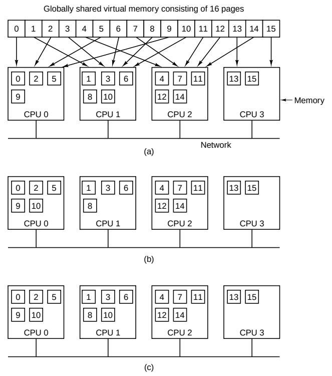


# Distributed Shared Memory (DSM)




a. the hardware,
b. the operating system,
c. user-level software.

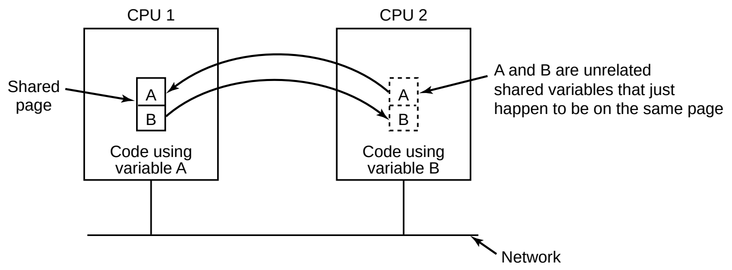
# DSM Memory Distribution




a. pages of the address space distributed among four machines.
b. situation after CPU 1 references page 10.
c. situation if page 10 is read only and replication is used..



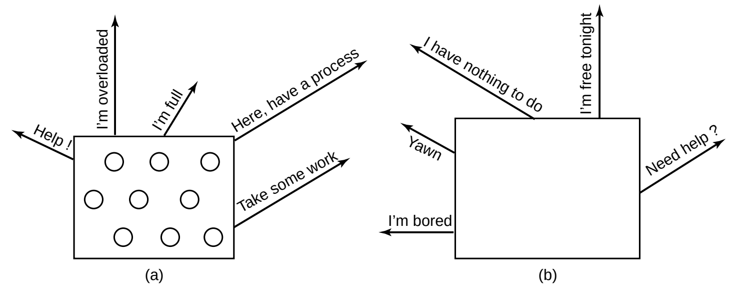
# False Sharing




**False sharing** of a page containing two unrelated variables.



# Load Balancing (I)




Load balancing - heuristic algorithms:

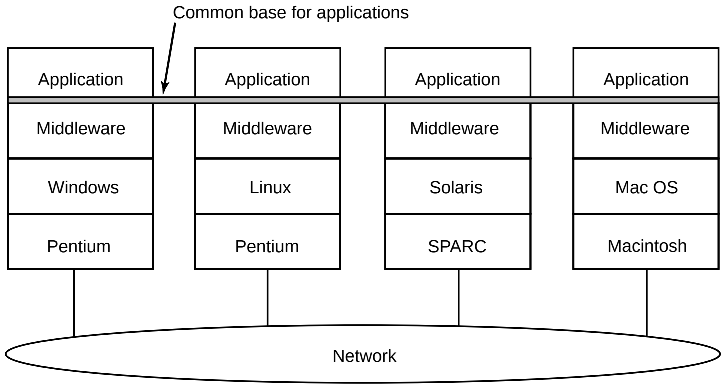
a. an overloaded node looking for a lightly loaded node to hand off process to.
b. an empty node looking for work to do.



# 3. Distributed Systems

| 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 |

# Middleware in Distibuted Systems




Positioning of middleware in a distributed system.

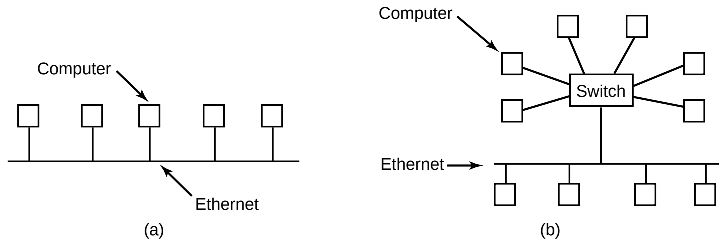


# Different Types of Middleware

1. Document-based middleware,
2. File system-based middleware,
3. Shared object-based middleware,
4. Coordination-based middleware.



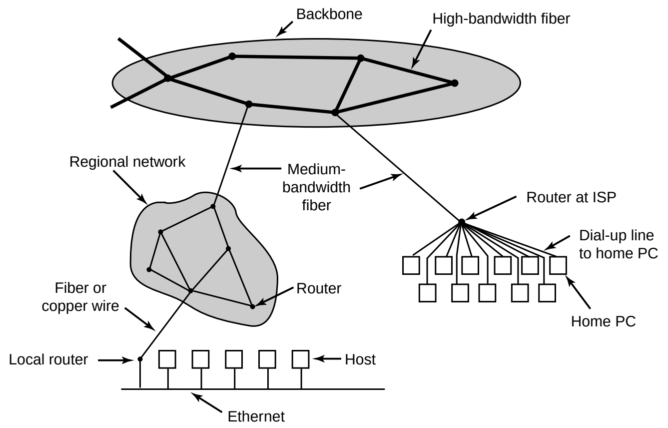
# Network Hardware




a. classic Ethernet,
b. switched Ethernet.



# The Internet



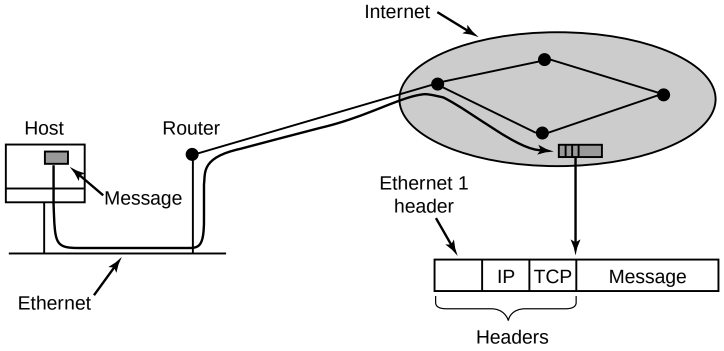
# Network services

| | 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 |

Different types of network services with examples.



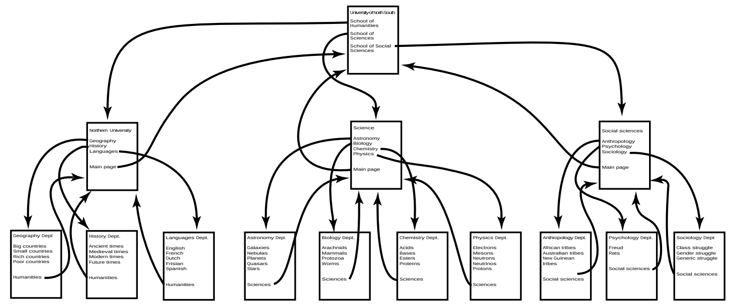
# Packet Headers




Accumulation of packet headers.



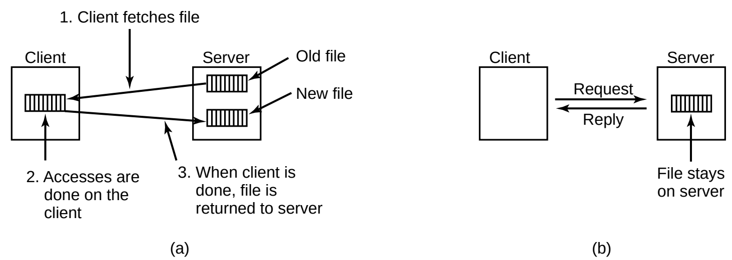
# Document-Based Middleware



WWW pages create a big directed graph of documents.



# File System-Based Middleware

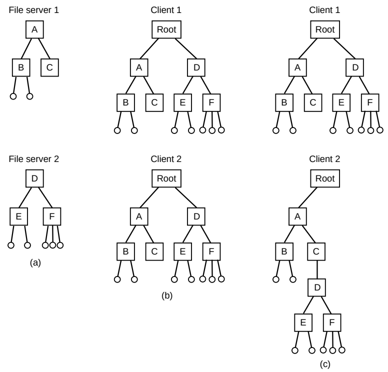



Transfer models:

a. the upload/download model,

b. the remote access model.

# Naming Transparency



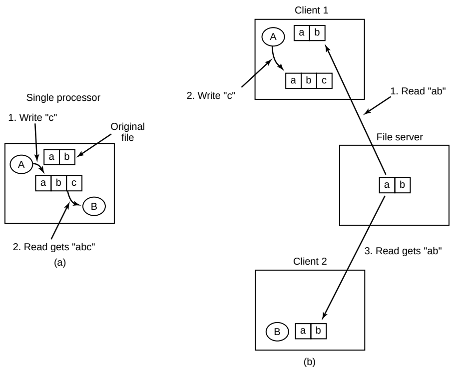

a. two file servers, the squares are directories and the cirles are files,

b. a system in which all clients have the same view of the file system,

c. a system in which different clients may have diffent views of the file system.



# Semantics of File Sharing

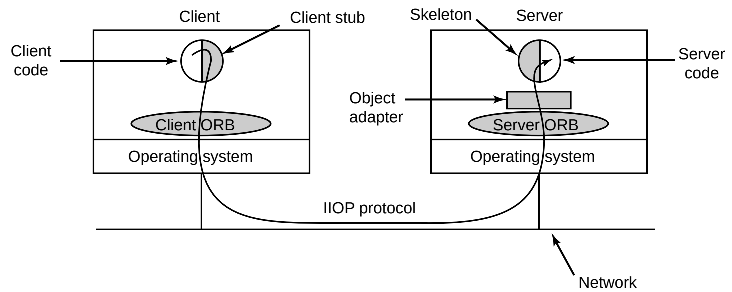



a. sequential consistency,

b. session semantics.



# Shared Object-Based Middleware




The main elements of a distributed system based on Corba. The Corba parts are shown in gray.



# Coordination-Based Middleware

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

Three Linda tuples.

- tuples and tuple spaces,
- communication and synchronization in one mechanism,
- **out**, **in**, **rd**, **eval** operations on tuples,
- solutions: Linda, JavaSpaces.