EECS 498 – Lecture Notes #1b
Introduction to Distributed Systems

Farnam Jahanian
Department of EECS
University of Michigan

EECS 498 Lecture Notes

http://www.eecs.umich.edu/~farnam

Lectures: Weeks 1-3

- Introduction to Networking: communication issues and network technologies
  Reading list: Tanenbaum text Chapter 2.1, pp 57-68, Layered Protocols
  Handout in class: Chapter 5: -- "Protocols Underlying HTTP" from "Web Protocols and Practices" by Krishnamurthy & Rexford, 2001.

- Introduction to distributed systems, characteristics of distributed systems, design issues, h/s concepts, distributed programming models
  Reading list: Tanenbaum text Chapter 1, pp 1-42.

- Overview of Distributed Computing Paradigms
  Reading List:
  Handout in class: Chapter 3 – from "Distributed Computing: Principles and Applications" text by M.L. Liu (recommended text)

Introduction to Distributed Systems

Distributed Systems

Three Technology Advances:

- Development of powerful microprocessors
- Development of high-speed networks
- Development of denser and cheaper memory/storage

Easy: put together large # of powerful processors connected by a high-speed network.

Hard: SOFTWARE! SOFTWARE! SOFTWARE!

Distributed Systems

What is a distributed system?

"You know you have one when the crash of a computer you've never heard of stops you from getting any work done." Leslie Lamport

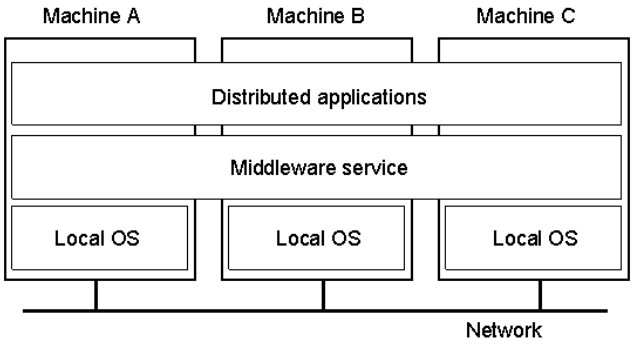
- A collection of (perhaps) heterogeneous nodes connected by one or more interconnection networks which provides access to system-wide shared resources and services.
- A collection of independent computers that appears to its users as a single coherent system.

Examples?

Characteristics of a distributed systems

- **Multiple Computers:**
  More than one physical computer, each consisting of CPUs, local memory, and possibly stable storage, and I/O paths to connect it with the environment.
- **Interconnections:**
  Mechanisms for communicating with other nodes via a network.
- **Shared State:**
  If a subset of nodes cooperate to provide a service, a shared state is maintained by these nodes. The shared state is distributed or replicated among the participants.

An Abstract View Distributed Systems



A distributed system organized as middleware.
Note that the middleware layer extends over multiple machines.

Distributed vs. Centralized Systems

Why distribute?

- Resource sharing
- Device sharing
- Flexibility to spread load
- Incremental growth
- Cost/performance
- Reliability/Availability
- Inherent distribution
- Security?

Why NOT distribute?

- Software
- Network
- Security
- System management

Numerous sources of complexity including:

- Transparent/uniform access to data or resources
- Independent failures of processors (or processes)
- Dynamic membership of a system
- Unreliable/unsecured communication
- Overhead in message communication
- Distribution or replication of data (or meta-data)
- Lack of clean common interfaces

Design Goals & Issues

- Connecting users and resources is the primary goal
- Transparency: hide the fact that processes and resources are physically distributed
- Openness: offer services according to rules and interfaces that describe the syntax and semantics of those services
  – Interoperability and portability
  -- Separating policy from mechanism
- Scalability
- Performance
- Dependability

Transparency in a Distributed System

Transparency	Description
Access	**Hide differences in data representation and how a resource is accessed**
Location	**Hide where a resource is located**
Migration	**Hide that a resource may move to another location**
Relocation	**Hide that a resource may be moved to another location while in use**
Replication	**Hide that a resource may be shared by several competitive users**
Concurrency	**Hide that a resource may be shared by several competitive users**
Failure	**Hide the failure and recovery of a resource**
Persistence	**Hide whether a (software) resource is in memory or on disk**

Different forms of transparency in a distributed system.

Transparency

- How to achieve “single-system image”? How to hide distribution from users or programs?
- Is it a good idea?

  Sometimes requires trade off transparency for performance

Scalability

- The challenge is to build distributed systems that scale with the increase in the number of CPUs, users, and processes, larger databases, etc.
- Scalability along several dimensions: size, geography, administrative domains

Scalability Problems

Concept	Example
Centralized services	**A single server for all users**
Centralized data (tables)	**A single on-line telephone book**
Centralized algorithms	**Doing routing based on complete information**

Examples of scalability limitations.

How to scale?

A very simple principle:
Avoid centralized services, centralized tables, and centralized algorithms

Characteristics of decentralized algorithms:

- No machine has complete information about the system state
- Machines make decisions based only on local information
- Failure of one machine does not ruin the algorithm
- There is no implicit assumption about a global clock

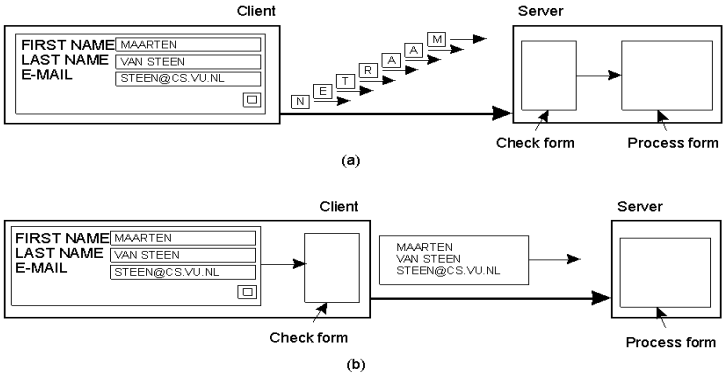
A few lessons from AFS:

- “Clients have cycles to burn.”
- “Cache whenever possible.”
- “Exploit usage properties.”
- “Minimize system-wide knowledge/change.”
- “Batch if possible.”
- Multicast often works!

Scaling Techniques (Tanenbaum's Text)

- Hiding communication latencies
  - Asynchronous communication
  - Function shipping to clients
- Distribution of components
  - DNS name space
- Caching and Replication
  - Maintaining consistency

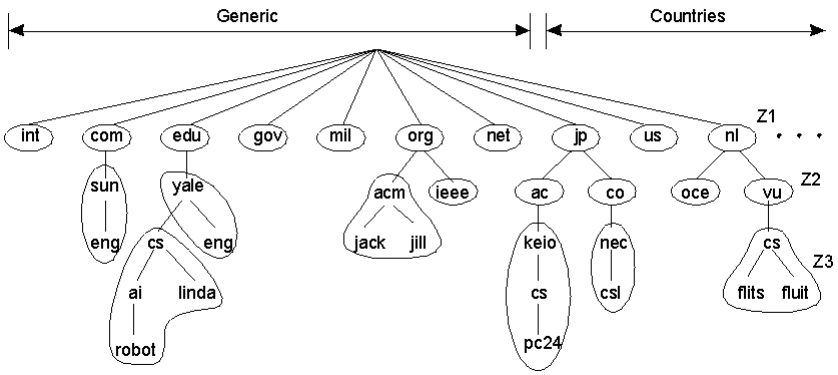
Scaling Techniques (1)



a) a server or

b) a client check forms as they are being filled

Scaling Techniques (2)



An example of dividing the DNS name space into zones.

Performance

Various performance metrics:

- response time
- throughput
- system utilization
- network capacity utilization

Key issue in parallelizing computations in a distributed system?

overhead of message communication

Performance

Trade off:

- More tasks → more parallelism → better performance
- More tasks → more communication → worse performance

Grain size affects # of messages:

fine-grained parallelism	vs.	coarse-grained parallelism
small computations		large computations
high interaction rate		low interaction rate

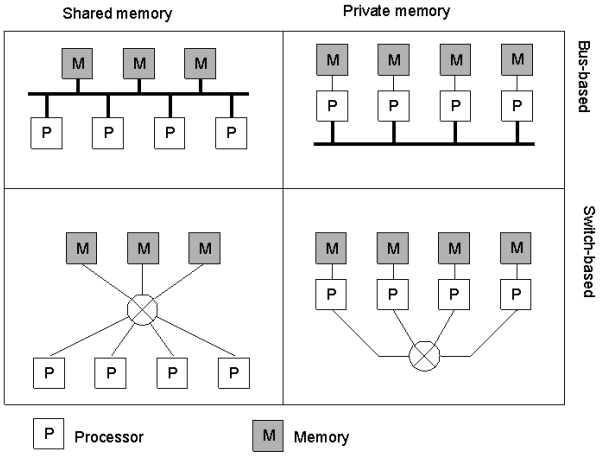
Dependability

- **Reliability: measured by the probability *R(t)* that the system is up (and providing correct service) during the time interval *[0,t]* assuming that it was operational at time t.**
- **Availability: measured by the probability *A(t)* that the system is operational at the instant of time t. As t → ∞, availability expresses the fraction of time that the system is usable.**
- **Timeliness: ability to meet timing constraints imposed on task execution or service delivery.**
- **Integrity: replicated copies of data must be kept *consistent*.**
- **Security: protection from unauthorized usage/access. Why more difficult in distributed systems?**

Distributed Programming Paradigms

- Client/server model
- Remote procedure calls
- Distributed File Systems
- Group communication and multicasts
- Distributed transactions
- Distributed shared memory
- Distributed object-based systems
- Publish-subscribe model
- Peer-to-peer model
- The Web

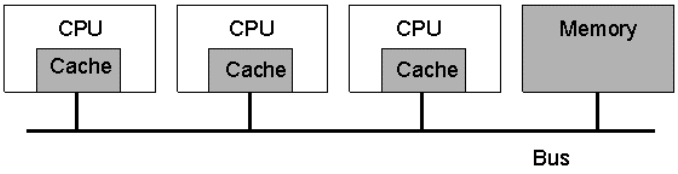
Hardware Concepts



Different basic organizations and memories in distributed computer systems

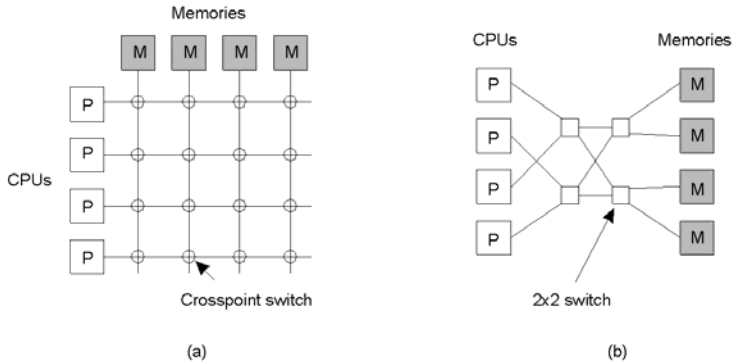
Multiprocessors (1)

- A bus-based multiprocessor.



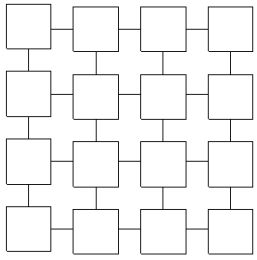
Multiprocessors (2)

a) A crossbar switch
b) An omega switching network

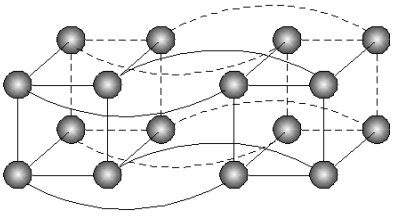


Homogeneous Multicomputer Systems

a) **Grid**
b) **Hypercube**



(a)



(b)

Heterogeneous Multicomputer Systems

- **Most distributed systems today are built on top of heterogeneous multicomputers and interconnection networks**
- **No global system view**
- **Sophisticated software needed to support distributed applications**

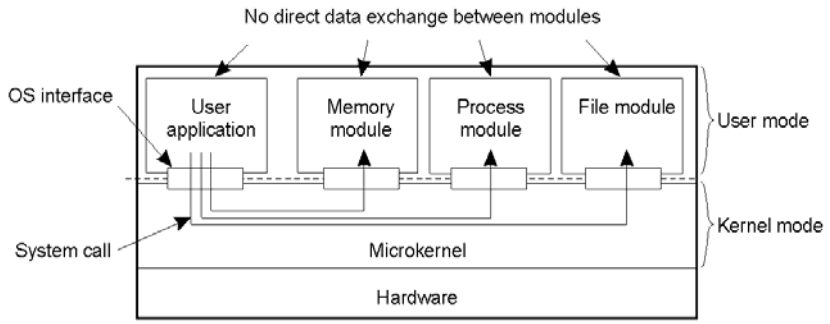
Software Concepts

System	Description	Main Goal
DOS	**Tightly-coupled operating system for multi-processors and homogeneous multicomputers**	**Hide and manage hardware resources**
NOS	**Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)**	**Offer local services to remote clients**
Middleware	**Additional layer atop of NOS implementing general-purpose services**	**Provide distribution transparency**

- **An overview of**
- **DOS (Distributed Operating Systems)**
- **NOS (Network Operating Systems)**
- **Middleware**

Uniprocessor Operating Systems

- **Separating applications from operating system code through a microkernel.**



Multiprocessor Operating Systems (1)

```
monitor Counter {
private:
 int count = 0;
public:
  int value() { return count;}
  void incr () { count = count + 1;}
  void decr() { count = count – 1;}
}
```

- **A monitor to protect an integer against concurrent access.**

Multiprocessor Operating Systems (2)

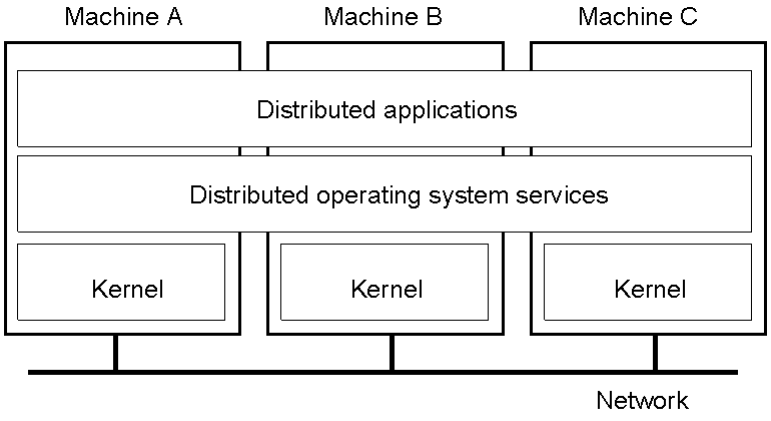
```
monitor Counter {
private:
  int count = 0;
  int blocked_procs = 0;
  condition unblocked;
public:
  int value () { return count;}
  void incr () {
       if (blocked_procs == 0)
         count = count + 1;
       else
         signal (unblocked);
}
```

```
void decr() {
if (count ==0) {
  blocked_procs = blocked_procs + 1;
  wait (unblocked);
  blocked_procs = blocked_procs – 1;
   }
  else
   count = count – 1;
 }
}
```

- **A monitor to protect an integer against concurrent access, but blocking a process.**

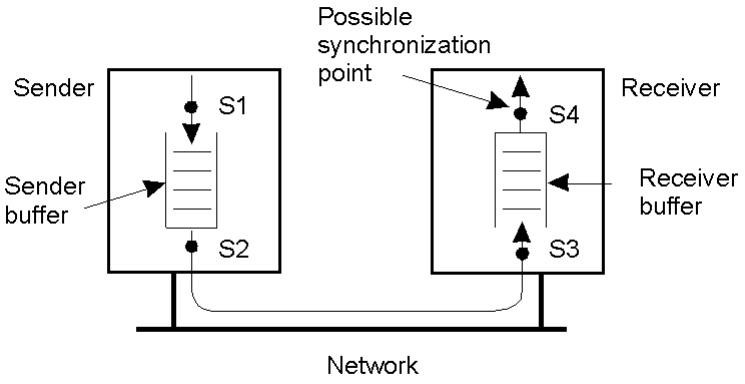
Multicomputer Operating Systems (1)

- **General structure of a multicomputer operating system**



Multicomputer Operating Systems (2)

- **Alternatives for blocking and buffering in message passing.**



Multicomputer Operating Systems (3)

Synchronization point	Send buffer	Reliable comm. guaranteed?
Block sender until buffer not full	**Yes**	**Not necessary**
Block sender until message sent	**No**	**Not necessary**
Block sender until message received	**No**	**Necessary**
Block sender until message delivered	**No**	**Necessary**

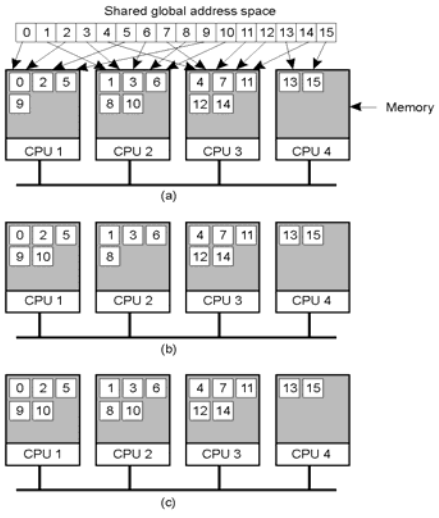
- **Relation between blocking, buffering, and reliable communications.**

Distributed Shared Memory Systems (1)

a) **Pages of address space distributed among four machines**

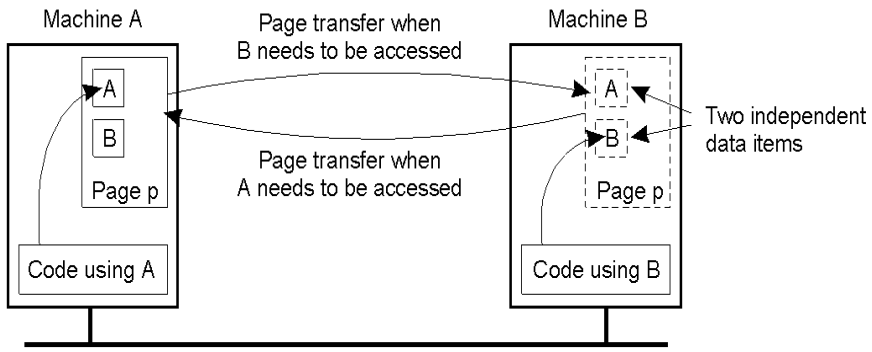
b) **Situation after CPU 1 references page 10**

c) **Situation if page 10 is read only and replication is used**



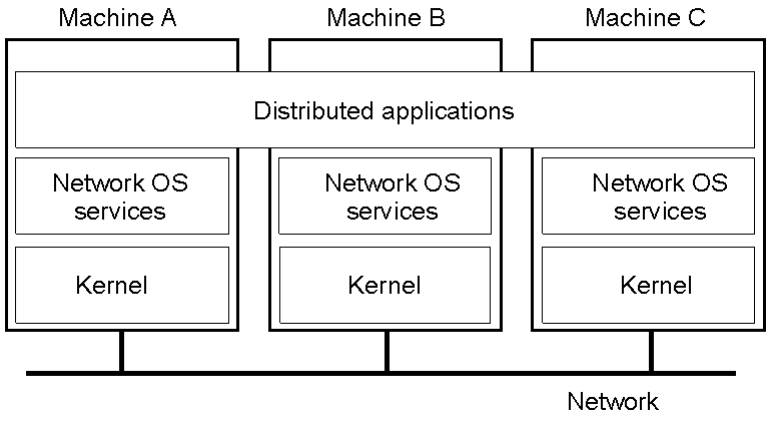
Distributed Shared Memory Systems (2)

- **False sharing of a page between two independent processes.**



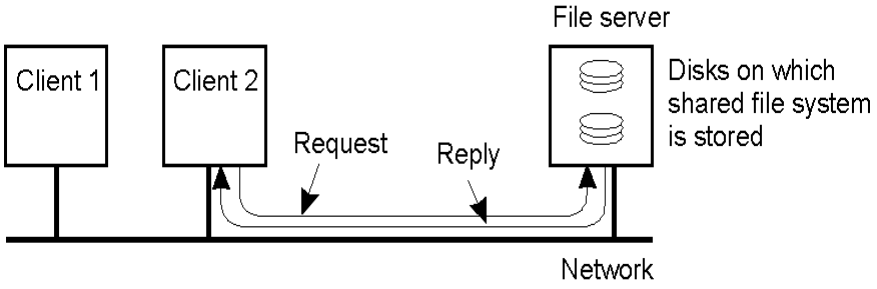
Network Operating System (1)

- **General structure of a network operating system.**



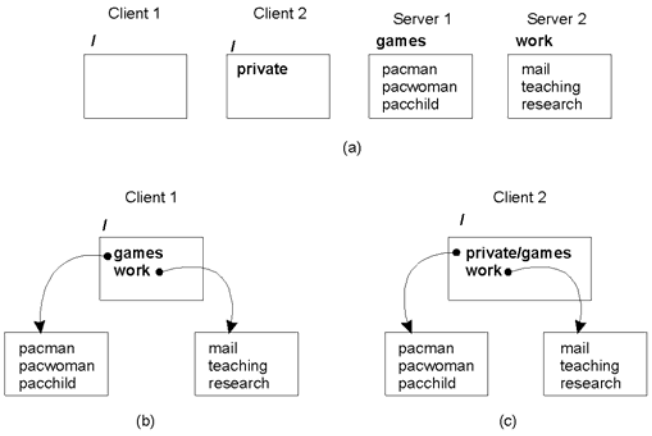
Network Operating System (2)

- **Two clients and a server in a network operating system.**



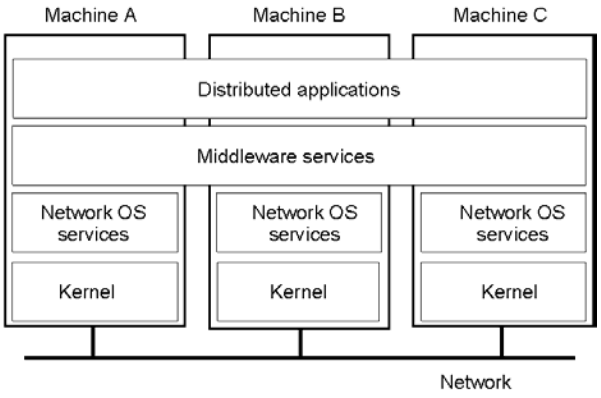
Network Operating System (3)

- **Different clients may mount the servers in different places.**

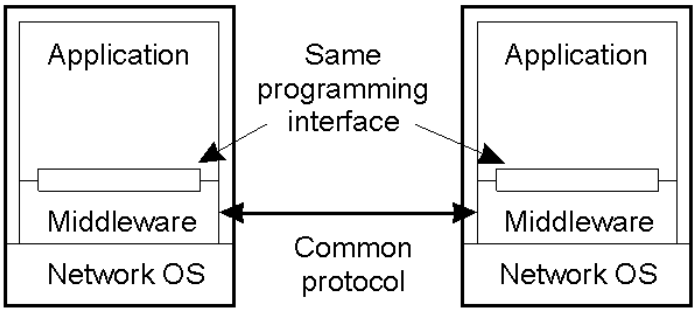


Positioning Middleware

- **General structure of a distributed system as middleware.**



Middleware and Openness



- **In an open middleware-based distributed system, the protocols used by each middleware layer should be the same, as well as the interfaces they offer to applications.**

Comparison between Systems

- A comparison between multiprocessor operating systems, multicomputer operating systems, network operating systems, and middleware based distributed systems.

Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc.	Multicomp.		
Degree of transparency	**Very High**	**High**	**Low**	**High**
Same OS on all nodes	**Yes**	**Yes**	**No**	**No**
Number of copies of OS	**1**	**N**	**N**	**N**
Basis for communication	**Shared memory**	**Messages**	**Files**	**Model specific**
Resource management	**Global, central**	**Global, distributed**	**Per node**	**Per node**
Scalability	**No**	**Moderately**	**Yes**	**Varies**
Openness	**Closed**	**Closed**	**Open**	**Open**